РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ

Регистратор данных CR1000

© 2000–2015
Campbell Scientific, Inc.

Гарантия

Регистратор данных CR1000 поставляется с трехлетней гарантией на следующих условиях.

Гарантия: компания CSI гарантирует отсутствие дефектов в материалах и качестве изготовления своих продуктов в условиях нормальной эксплуатации в течение 36 месяцев с даты поставки, если в соответствующем руководстве по эксплуатации не указано иное. (Руководства по эксплуатации продуктов CSI доступны для ознакомления на вебсайте CSI по ссылке www.campbellsci.com.) Гарантия на изделия сторонних производителей, поставляемые CSI, предоставляется в объеме, аналогичном оригинальной гарантии производителя. Данная гарантия не распространяется на аккумуляторные батареи, термопары из тонкой проволоки, влагопоглотители и прочие расходуемые материалы. Обязательства CSI по настоящей гарантии ограничены ремонтом или заменой (на усмотрение CSI) дефектных изделий. Настоящая гарантия не предусматривает какие-либо иные средства компенсации. Заказчик оплачивает все расходы по демонтажу, пересылке в CSI и повторной установке дефектных изделий. Возврат изделий заказчикам на территории континентальной части США производится наземной почтой с предоплатой. Возврат изделий заказчикам в других регионах производится оптимальным из доступных способов на условиях CIP (Port of entry) Incoterms ® 2010. Настоящая гарантия не распространяется на продукты, подвергавшиеся модификации, использовавшиеся с нарушением правил эксплуатации или обслуживания, а также поврежденные в силу естественных причин или при транспортировке. Настоящая гарантия заменяет все прочие гарантии, как явные, так и подразумеваемые. Гарантия на услуги по установке, оказываемые специалистами CSI, такие как внесение изменений в код ПО для соответствия требованиям заказчика, подключение изделий CSI к электросети или обучение работе с поставляемыми продуктами, включена в общую продуктовую гарантию CSI. CSI БЕЗОГОВОРОЧНО ОТКАЗЫВАЕТСЯ ОТ ЛЮБЫХ ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ ГОДНОСТИ ДЛЯ ПРОДАЖИ И ГОДНОСТИ ДЛЯ КОНКРЕТНЫХ ЗАДАЧ. Во всех установленных законом случаях CSI отказывается от любых (включая явные, подразумеваемые и предусмотренные законом) гарантий и условий по отношению к Продуктам, за исключением изложенных в настоящем документе.

Поддержка

Возврат продуктов возможен только после получения авторизованного разрешения. Ниже приведена контактная информация для заказчиков из США и из стран, прямо обслуживаемых Campbell Scientific, Inc. Аффилированные компании обеспечивают ремонт дефектных продуктов для заказчиков на своей территории. Чтобы узнать, какая компания Campbell Scientific обслуживает вашу страну, пройдите по ссылке *www.campbellsci.com*.

Чтобы получить авторизованное разрешение на возврат материалов (RMA), свяжитесь с Campbell Scientific, Inc. по телефону +1 (435) 227-2342. После того как технические специалисты рассмотрят ваше обращение, вам будет выдано авторизованное разрешение на возврат. Запишите номер разрешения на внешней стороне транспортной тары. Почтовый адрес для возврата дефектных изделий:

CAMPBELL SCIENTIFIC, INC.
Авторизованное разрешение на возврат
#_____
815 Уэст 1800 Норт
США, Юта, г. Логан, 84321-1784

При возврате изделия заказчик заполняет «Заявление об очистке и отсутствии загрязнений» и исполняет все перечисленные в данной форме требования. Форма заявления доступна на вебсайте Campbell по ссылке *www.campbellsci.com/repair*. Заполненная форма пересылается по электронной почте на адрес *repair@campbellsci.com* или факсом на номер +1 (435) 2279579. Мы не принимаем изделия, для которых заказчик не предоставил данное заявление. Если в течение трех дней с даты доставки изделия заявление не было получено или было получено в частично заполненном виде, изделие возвращается заказчику с оплатой почтовых расходов за его счет. Campbell Scientific оставляет за собой право отказаться от обслуживания изделий, подвергавшихся загрязнению, потенциально опасному для здоровья или безопасности наших сотрудников.

Меры предосторожности

ВНИМАНИЕ, ОПАСНОСТЬ! УСТАНОВКА, ЭКСПЛУАТАЦИЯ И ОБСЛУЖИВАНИЕ ТРЕНОЖНЫХ МАЧТ, БАШЕН, А ТАКЖЕ УСТАНАВЛИВАЕМЫХ НА НИХ УСТРОЙСТВ, ТАКИХ КАК ДАТЧИКИ, ЗАЩИТНЫЕ КОРПУСА, КРЕСТОВИНЫ, АНТЕННЫ И Т. Д., ЯВЛЯЕТСЯ ИСТОЧНИКОМ ПОВЫШЕННОЙ ОПАСНОСТИ. НЕПОЛНАЯ ИЛИ НЕНАДЛЕЖАЩАЯ СБОРКА, УСТАНОВКА, ЭКСПЛУАТАЦИЯ ИЛИ ОБСЛУЖИВАНИЕ ТРЕНОЖНЫХ МАЧТ, БАШЕН И УСТАНАВЛИВАЕМЫХ НА НИХ УСТРОЙСТВ, А ТАКЖЕ ИГНОРИРОВАНИЕ ПРЕДУПРЕЖДАЮЩИХ ЗНАКОВ МОЖЕТ ПРИВЕСТИ К СМЕРТИ, НЕСЧАСТНОМУ СЛУЧАЮ, ТЯЖЕЛЫМ ТРАВМАМ, ПОВРЕЖДЕНИЯМ ИМУЩЕСТВА И НЕРАБОТОСПОСОБНОСТИ ПРОДУКТОВ. ПРИНИМАЙТЕ ВСЕ РАЗУМНЫЕ МЕРЫ ПРЕДОСТОРОЖНОСТИ ДЛЯ ПРЕДОТВРАЩЕНИЯ ПОДОБНЫХ СЛУЧАЕВ. ПЕРЕД НАЧАЛОМ РАБОТ СВЯЖИТЕСЬ С ЛИЦОМ, ОТВЕЧАЮЩИМ ЗА ОХРАНУ ТРУДА В ВАШЕЙ ОРГАНИЗАЦИИ, ОЗНАКОМЬТЕСЬ С НОРМАМИ БЕЗОПАСНОСТИ И ПОЛУЧИТЕ ПОЛОЖЕННЫЕ ВАМ СРЕДСТВА РАБОЧЕЙ ЗАЩИТЫ.

Треножные мачты, башни, а также устанавливаемое на них оборудование используйте строго по назначению. Не нарушайте эксплуатационные нормы безопасности. Ознакомьтесь и соблюдайте все требования безопасности, приведенные в руководствах по эксплуатации изделий. Руководства по эксплуатации Вы можете скачать по ссылке www.campbellsci.com или запросить по телефону +1 (435) 227-9000. Вы несете полную ответственность за соблюдение всех рабочих норм и требований, включая требования безопасности, а также за расположение и пригодность строений или участков земли, выбранных для установки треножных мачт, башен и дополнительного оборудования. Установка треножных мачт, башен и дополнительного оборудования должна быть согласована с квалифицированным инженером. По всем вопросам установки, эксплуатации и обслуживания треножных мачт, башен, устанавливаемого на них оборудования и электрических соединений обращайтесь к квалифицированному инженеру или электрику.

Общая информация

- Перед началом работ по установке или эксплуатации оборудования получите все необходимые разрешения. Соблюдайте все нормативные требования вышестоящих организаций.
- Привлекайте к работам по установке, эксплуатации и обслуживанию треножных мачт, башен и дополнительного оборудования только квалифицированный персонал. Настоятельно рекомендуется прибегнуть к услугам лицензированного и квалифицированного подрядчика.
- Перед началом работ внимательно изучите инструкции и ознакомьтесь с порядком работ.
- На треножных мачтах и башнях всегда работайте в каске и защитных очках, а также принимайте прочие разумные меры предосторожности при работе.
- Не поднимайтесь на треножные мачты и башни; следите, чтобы на них не поднимались другие. Примите разумные меры по ограждению территории вокруг треножной мачты или башни от посторонних.
- Используйте только рекомендованные производителем детали, материалы и инструмент.

Инженерные и электрические коммуникации

- Соприкосновение треножной мачты, башни или установленного на них оборудования, а также вашего рабочего инструмента с подземными или воздушными коммуникациями может привести к серьезной травме или смерти.

- Следите за тем, чтобы расстояние от воздушных коммуникаций до треножной мачты, башни, установленного на них оборудования и рабочего инструмента составляло не менее 6 м, 150 % высоты башни, либо установленной нормативами дистанции (в зависимости от того, какая из этих дистанций больше).
- Перед началом установочных или эксплуатационных работ уведомите все обслуживающие компании, отвечающие за проложенные поблизости коммуникации, и разметьте все подземные коммуникации на участке.
- Соблюдайте электротехнические правила безопасности. Установку электрооборудования и систем заземления должен выполнять лицензированный и квалифицированный электрик.

Работа на высоте, погодные условия

- Работая на высоте, будьте предельно осторожны.
- Используйте необходимое защитное оборудование, соблюдайте технику безопасности.
- В ходе установки и обслуживания примите меры по ограждению территории вокруг треножной мачты или башни от неквалифицированного и вспомогательного персонала. Будьте внимательны, чтобы не уронить с высоты рабочий инструмент и прочие предметы.
- Не выполняйте работы в неблагоприятных погодных условиях, включая сильный ветер, дождь, снег, грозу и т. д.

Обслуживание

- Регулярно (не реже раза в год) проверяйте оборудование на износ и повреждения, включая коррозию, вызванные нагрузкой трещины, повреждения изоляции проводов, чрезмерно натянутые провода, плохо закрепленные провода и т. д. При необходимости устраняйте возникшие повреждения.
- Регулярно (не реже раза в год) проверяйте подключение линий заземления.

CAMPBELL SCIENTIFIC ПРИКЛАДЫВАЕТ ВСЕ УСИЛИЯ ДЛЯ ОБЕСПЕЧЕНИЯ ВЫСОЧАЙШЕЙ НАДЕЖНОСТИ ВСЕХ ПРОДУКТОВ КОМПАНИИ. ТЕМ НЕ МЕНЕЕ ПОЛЬЗОВАТЕЛЬ ПРИНИМАЕТ ОТВЕТСТВЕННОСТЬ ЗА ВСЕ РИСКИ НЕСЧАСТНОГО СЛУЧАЯ, СВЯЗАННЫЕ С НЕПРАВИЛЬНОЙ УСТАНОВКОЙ, ЭКСПЛУАТАЦИЕЙ ИЛИ ОБСЛУЖИВАНИЕМ ТРЕНОЖНЫХ МАЧТ, БАШЕН, А ТАКЖЕ УСТАНАВЛИВАЕМЫХ НА НИХ УСТРОЙСТВ, ТАКИХ КАК ДАТЧИКИ, ЗАЩИТНЫЕ КОРПУСА, КРЕСТОВИНЫ, АНТЕННЫ И Т.Д.

Оглавление

1. Введение 31

1.1. Добро пожаловать! 31
1.2. Условные обозначения 31
1.3. Копирование кода CRBasic 32
1.4. Описание текущей версии 32

2. Предупреждения 35

3. Перед началом работы 37

4. Краткое руководство по системе 39

4.1. Системы сбора данных — краткое руководство 39
4.2. Датчики — краткое руководство 40
4.3. Регистратор данных — краткое руководство 41
4.3.1.1. Коммутационная панель — краткое руководство 41
4.4. Источники питания — краткое руководство 42
4.4.1. Внутренняя батарея — краткое руководство 43
4.5. Сбор и обмен данными — краткое руководство 43
4.6. ПО поддержки регистратора данных — краткое руководство 44
4.7. Учебное пособие: считывание сигнала с термопары 44
4.7.1. Необходимое оборудование 44
4.7.2. Подготовка оборудования 45
4.7.2.1. Внешний источник питания 45
4.7.3. Настройка программы PC200W 46
4.7.4. Написание программы CRBasic при помощи Short Cut 48
4.7.4.1. Процедура: (шаги 1–5) 48
4.7.4.2. Процедура: (шаги 6–7) 49
4.7.4.3. Процедура: (шаг 8) 50
4.7.4.4. Процедура: (шаги 9–12) 51
4.7.4.5. Процедура: (шаги 13–14) 52
4.7.5. Отправка программы и сбор данных 53
4.7.5.1. Процедура: (шаг 1) 53
4.7.5.2. Процедура: (шаги 2–4) 53
4.7.5.3. Процедура: (шаг 5) 54
4.7.5.4. Процедура: (шаг 6) 55
4.7.5.5. Процедура: (шаги 7–10) 56
4.7.5.6. Процедура: (шаги 11–12) 57
4.7.5.7. Процедура: (шаги 13–14) 57

5. Общие сведения по системе 59

5.1. Измерения — общие сведения 60
5.1.1. Хронометраж — общие сведения 61
5.1.2. Аналоговые измерения — общие сведения 61
5.1.2.1. Измерение напряжения — общие сведения 61
5.1.2.1.1. Несимметричные измерения — общие сведения 63
5.1.2.1.2. Дифференциальные измерения — общие сведения 64
5.1.2.2. Измерения тока — общие сведения 64
5.1.2.3. Измерения сопротивления — общие сведения 65
5.1.2.3.1. Напряжение возбуждения 65
5.1.2.4. Тензометрические измерения — общие сведения 66
5.1.3. Импульсные измерения — общие сведения 66
5.1.3.1. Измерение импульсов 67
5.1.3.2. Каналы импульсного входа 67

5.1.3.3. Подключение импульсного датчика 68
5.1.4. Среднее значение за период — общие сведения 68
5.1.5. Вибрационные струнные измерения — общие сведения 69
5.1.6. Снятие показаний интеллектуальных датчиков — общие сведения 69
5.1.6.1. Поддержка датчика SDI-12 — общие сведения 70
5.1.6.2. RS-232 — общие сведения 70
5.1.7. Калибровка на месте — общие сведения 71
5.1.8. Эффекты подключения — общие сведения 72
5.1.9. Синхронизация измерений — общие сведения 72
5.2. Управление ПЛК — общие сведения 72
5.3. Регистратор данных — общие сведения 73
5.3.1. Хронометраж — общие сведения 73
5.3.2. Коммутационная панель — общие сведения 74
5.3.2.1. Переключаемое выходное напряжение — общие сведения 76
5.3.2.2. Напряжение возбуждения — общие сведения 77
5.3.2.3. Заземляющие клеммы 78
5.3.2.4. Силовые клеммы 78
5.3.2.4.1. Подача питания 78
5.3.2.4.2. Выходные силовые клеммы 78
5.3.2.5. Коммуникационные порты 79
5.3.2.5.1. Порт CS B/B 79
5.3.2.5.2. Порты RS-232 80
5.3.2.5.3. Периферийный порт 80
5.3.2.5.4. Порты SDI-12 80
5.3.2.5.5. Порт SDM 80
5.3.2.5.6. Порт CPI 80
5.3.2.5.7. Порт Ethernet 81
5.3.3. Экранная клавиатура — общие сведения 81
5.3.3.1. Набор символов 81
5.3.3.2. Пользовательские меню — общие сведения 82
5.3.4. Внешнее оборудование для измерения и контроля — общие сведения 83
5.3.5. Источники питания — общие сведения 83
5.3.6. Конфигурация CR1000 — общие сведения 84
5.3.7. Программирование CRBasic — общие сведения 84
5.3.8. Память — общие сведения 85
5.3.9. Передача и сбор данных — общие сведения 86
5.3.9.1. Связь PakBus® — общие сведения 86
5.3.9.2. Связь 87
5.3.9.3. Устройство памяти большой емкости 87
5.3.9.4. Карта памяти (CRD: диск) — общие сведения 87
5.3.9.5. Форматы файлов данных в памяти CR1000 88
5.3.9.6. Формат данных на компьютере 88
5.3.10. Альтернативная связь — общие сведения 88
5.3.10.1. Modbus 89
5.3.10.2. DNP3 — общие сведения 89
5.3.10.3. TCP/IP — общие сведения 89
5.3.11. Безопасность — общие сведения 90
5.3.12. Обслуживание — общие сведения 91
5.3.12.1. Защита от влаги — общие сведения 91
5.3.12.2. Защита от перепадов напряжения 92
5.3.12.3. Заводская калибровка 92
5.3.12.4. Внутренняя батарея — подробная информация 92
5.4. ПО поддержки регистратора данных — общие сведения 93

6. Технические характеристики 95

7. Установка 97

7.1. Защита от влаги — подробная информация 97
7.2. Диапазон рабочих температур 97

7.3. Защитные корпуса 97
7.4. Источники питания — подробная информация 98
7.4.1. Потребность CR1000 в электроэнергии 99
7.4.2. Расчетная потребляемая мощность 99
7.4.3. Источники питания 99
7.4.3.1. Подключение питания от автомобиля 100
7.4.4. Источник бесперебойного питания (ИБП) 100
7.4.5. Установка внешнего источника питания 101
7.5. Переключаемое выходное напряжение — подробная информация ...101
7.5.1. Переключаемое напряжение возбуждения 102
7.5.2. Непрерывно регулируемый (клемма 5 В) 102
7.5.3. Непрерывное нерегулируемое напряжение (клемма 12 В) ...102
7.5.4. Переключаемое нерегулируемое напряжение (клемма SW12) 103
7.6. Заземление 103
7.6.1. Защита ЭСР 103
7.6.1.1. Грозозащита 105
7.6.2. Несимметричное эталонное измерение 106
7.6.3. Разница потенциала заземления 107
7.6.3.1. Термопара для измерения температуры почвы 107
7.6.3.2. Внешний преобразователь сигнала 107
7.6.4. Контур заземления при ионных измерениях 107
7.7. Конфигурация CR1000 — подробная информация 109
7.7.1.1. Конфигурация с DevConfig 109
7.7.1.2. Планировщик сети 110
7.7.1.2.1. Общие сведения 111
7.7.1.2.2. Основы 112
7.7.1.3. Конфигурация на основе параметров состояния/настроек/ таблиц 112
7.7.1.4. Конфигурация с исполняемыми файлами ЦП 113
7.7.1.4.1. Файл «Default.cr1» 114
7.7.1.4.2. Приоритеты запуска исполняемого файла 114
7.7.2.1. Обновление операционной системы (ОС) 115
7.7.2.1.1. Обновление ОС с помощью вкладки DevConfig для отправки ОС 116
7.7.2.1.2. Обновление ОС с помощью DevConfig 117
7.7.2.1.3. Обновление ОС с помощью DevConfig 117
7.7.2.1.4. Обновление ОС с помощью DevConfig 119
7.7.2.2. Восстановление заводских настроек по умолчанию 120
7.7.2.3. Сохранение и восстановление конфигурации 120
7.8. Программирование CRBasic — подробная информация 120
7.8.1. Структура программы 121
7.8.2. Написание и редактирование программ 123
7.8.2.1. Мастер программирования Short Cut 123
7.8.2.2. Редактор CRBasic 123
7.8.2.2.1. Добавление комментариев 124
7.8.2.2.2. Экономия памяти программы 124
7.8.3. Отправка программ CRBasic 124
7.8.3.1. Сохранение данных при отправке программы 125
7.8.4. Синтаксис программирования 126
7.8.4.1. Операторы программы 126
7.8.4.1.1. Несколько операторов в одной строке 126
7.8.4.1.2. Один оператор на нескольких строках 126
7.8.4.2. Объявления с одним оператором 127
7.8.4.3. Объявление переменных 127
7.8.4.3.1. Объявление типов данных 128

7.8.4.3.2. Размерность числовых переменных ... 132
7.8.4.3.3. Размеры строчных переменных ... 132
7.8.4.3.4. Объявление переменных Flag ... 133
7.8.4.4. Объявление массивов ... 133
7.8.4.5. Объявление локальных и глобальных переменных ... 134
7.8.4.6. Инициализация переменных ... 135
7.8.4.7. Объявление констант ... 135
7.8.4.7.1. Предварительно определенные константы ... 136
7.8.4.8. Объявление псевдонимов и единиц измерения ... 136
7.8.4.9. Числовые форматы ... 137
7.8.4.10. Объявления с несколькими операторами ... 138
7.8.4.10.1. Объявление таблицы данных ... 138
7.8.4.10.2. Объявление подпрограмм ... 145
7.8.4.10.3. «Включаемый» файл ... 145
7.8.4.10.4. Объявление подпрограмм ... 148
7.8.4.10.5. Объявление вспомогательных последовательностей ... 148
7.8.4.11. Выполнение и приоритет задачи ... 149
7.8.4.11.1. Поточный режим ... 150
7.8.4.11.2. Последовательный режим ... 151
7.8.4.12. Время выполнения ... 152
7.8.4.12.1. Команда Scan() / NextScan ... 152
7.8.4.12.2. Команды SlowSequence / EndSequence ... 153
7.8.4.12.3. Команды SubScan() / NextSubScan ... 154
7.8.4.12.4. Приоритеты сканирования в последовательном режиме ... 154
7.8.4.13. Команды программирования ... 156
7.8.4.13.1. Измерение и обработка сохраненных данных ... 156
7.8.4.13.2. Типы аргументов ... 157
7.8.4.13.3. Имена в качестве аргументов ... 157
7.8.4.14. Выражения в качестве аргументов ... 158
7.8.4.15. Типы программируемых выражений ... 158
7.8.4.15.1. Арифметика чисел с плавающей запятой ... 159
7.8.4.15.2. Математические операции ... 159
7.8.4.15.3. Выражения с типами числовых данных ... 160
7.8.4.15.4. Логические выражения ... 162
7.8.4.15.5. Строковые выражения ... 164
7.8.4.16. Программирование доступа к таблицам данных ... 165
7.8.4.17. Программирование использования сигнатур ... 167
7.9. Библиотека программных ресурсов ... 167
7.9.1.1. Считывание событий ... 167
7.9.1.2. Условный вывод ... 168
7.9.1.3. Тестирование насоса для грунтовых вод ... 169
7.9.1.4. Прочие возможности ... 172
7.9.1.5. Команда PulseCountReset ... 175
7.9.1.6. Масштабирование массива ... 175
7.9.1.7. Сигнатуры: примеры программ ... 176
7.9.1.7.1. Текстовая сигнатура ... 176
7.9.1.7.2. Сигнатура времени выполнения ... 176
7.9.1.7.3. Сигнатуры исполняемого кода ... 176
7.9.1.8. Использование многократного сканирования ... 177
7.9.2. Компиляция: условный код ... 178
7.9.3. Отображение данных: пользовательские меню — подробная информация ... 180
7.9.4. Входные данные: загрузка больших наборов данных ... 186
7.9.5. Входные данные: выражение для назначения массива ... 186

7.9.6. Выходные данные: вычисление скользящего среднего 190
7.9.7. Выходные данные: триггеры и исключительная выборка 193
7.9.8. Выходные данные: два интервала в одной таблице данных.195
7.9.9. Выходные данные: использование типа данных Bool8.....196
7.9.10. Выходные данные: использование типа данных NSEC 200
7.9.10.1. Варианты использования NSEC 200
7.9.11. Выходные данные: Запись высокочастотных данных на карты памяти 203
7.9.11.1. Команда TableFile() с параметром Option 64.....204
7.9.11.2. Команда TableFile() с параметром Option 64 как замена команды CardOut().....204
7.9.11.3. Программирование команды TableFile() с параметром Option 64 205
7.9.11.4. Преобразование файлов TOB3 с помощью CardConvert.....205
7.9.11.5. Команда TableFile() с параметром Option 64 — вопросы и ответы 206
7.9.12. Калибровка на месте — описание.....208
7.9.12.1. Калибровка на месте — файлы CAL.....208
7.9.12.2. Калибровка на месте — программирование.....209
7.9.12.3. Калибровка на месте — общее описание мастера калибровки.....209
7.9.12.4. Калибровка на месте — процедуры мониторинга данных.....209
7.9.12.4.1. Одноточечная калибровка (с нулем или со смещением).....210
7.9.12.4.2. Двухточечная калибровка (усиление и смещение) 211
7.9.12.4.3. Калибровка нулевой базовой точки....211
7.9.12.5. Примеры калибровки.....211
7.9.12.5.1. Пример калибровки FieldCal() с нулем или Tare (Opt 0) 212
7.9.12.5.2. Пример калибровки FieldCal() смещения (Opt 1) 214
7.9.12.5.3. Пример калибровки FieldCal() наклона и смещения (Opt 2) 216
7.9.12.5.4. Пример калибровки наклона FieldCal() (вариант 3) 218
7.9.12.5.5. Пример калибровки нулевой точки FieldCal() (вариант 4) — 8 10 30 221
7.9.12.6. Примеры калибровки деформации.....221
7.9.12.6.1. Примеры калибровки деформации 221
7.9.12.6.2. Примеры калибровки деформации 222
7.9.12.6.3. Пример калибровки четвертьмостового шунта FieldCalStrain().....224
7.9.12.6.4. FieldCalStrain() — калибровка четвертьмостового нуля.....225
7.9.13. Измерения: возбуждение, задержка, считывание показаний226
7.9.14. Измерения: повышенные аналоговые частоты 227
7.9.14.1. Измерения от 1 до 100 Гц.....228
7.9.14.2. Частота измерений: от 101 до 600 Гц.....229
7.9.14.2.1. Измерения от 101 до 600 Гц 2 230
7.9.14.3. Частота измерений: от 601 до 2000 Гц.....231
7.9.15. Измерения: ПТС.....232
7.9.15.1. Измерение PT100 (ПТС на 100 Ом) 233
7.9.15.1.1. Самонагрев и разрешающая способность.....233
7.9.15.1.2. Стандарты вычислений для ПТС.....233
7.9.15.2. PT100 в четырехжильном полумостовом подключении 236
7.9.15.2.1. Расчет напряжения возбуждения.....237

7.9.15.2.2. Расчет значения множителя BrHalf4W() 237
7.9.15.2.3. Выбор Rf 238
7.9.15.3. PT100 в трехжильном полумостовом подключении 239
7.9.15.4. PT100 в четырехжильном подключении с полным мостом 240
7.9.16. Контроль ПЛК — подробная информация 242
7.9.17. Последовательный интерфейс ввода/вывода: считывание последовательных данных 243
7.9.17.1. Введение 243
7.9.17.2. Порты ввода/вывода 244
7.9.17.3. Протоколы 245
7.9.17.4. Глоссарий терминов для последовательных интерфейсов ввода/вывода 245
7.9.17.5. Программирование последовательных интерфейсов ввода/вывода CRBasic 247
7.9.17.5.1. Основы программирования последовательных интерфейсов ввода/вывода 248
7.9.17.5.2. Основы программирования последовательных интерфейсов ввода/вывода на входе 249
7.9.17.5.3. Основы программирования последовательных интерфейсов ввода/вывода на выходе 250
7.9.17.5.4. Перевод байтов в последовательных интерфейсах ввода/вывода 251
7.9.17.5.5. Аспекты памяти для последовательных интерфейсов ввода/вывода 251
7.9.17.5.6. Демонстрационная программа 252
7.9.17.6. Проверка приложения последовательного ввода/вывода 254
7.9.17.6.1. Конфигурация HyperTerminal 254
7.9.17.6.2. Создание текстового файла для отправки 256
7.9.17.6.3. Создание файла захвата текста 256
7.9.17.6.4. Последовательный ввод/вывод: пример II 257
7.9.17.7. Последовательный ввод/вывод: вопросы и ответы 262
7.9.18. Последовательный ввод/вывод: поддержка датчика SDI-12 — программные ресурсы 265
7.9.18.1. SDI-12: прозрачный режим 265
7.9.18.1.1. SDI-12: команды прозрачного режима 266
7.9.18.2. SDI-12: режим записи 270
7.9.18.3. SDI-12: режим датчика 277
7.9.18.4. SDI-12: аспекты питания 279
7.9.19. Строковые операции 280
7.9.19.1. Строковые операторы 280
7.9.19.2. Конкатенация строк 281
7.9.19.3. Нулевой символ в строке 283
7.9.19.4. Вставка строковых символов 284
7.9.19.5. Извлечение строковых символов 284
7.9.19.6. Использование ASCII- / ANSII-кодов в строке 285
7.9.19.7. Форматирование строк 285
7.9.19.8. Форматирование строки с шестнадцатеричными переменными 286
7.9.20. Подпрограммы 286
7.9.21. TCP/IP — подробная информация 287

7.9.21.1. PakBus через TCP/IP и обратный вызов 288
7.9.21.2. Веб-сервер HTTP по умолчанию 289
7.9.21.3. Пользовательский веб-сервер HTTP 289
7.9.21.4. Сервер FTP 292
7.9.21.5. FTP-клиент 292
7.9.21.6. Telnet 293
7.9.21.7. SNMP 293
7.9.21.8. Проверка связи (IP) 293
7.9.21.9. Микро-последовательный сервер 293
7.9.21.10. Modbus TCP/IP 293
7.9.21.11. DHCP 293
7.9.21.12. DNS 294
7.9.21.13. SMTP 294
7.9.22. Вектор ветра 294
7.9.22.1. Параметры OutputOpt 294
7.9.22.2. Обработка вектора ветра 295
7.9.22.2.1. Измеренные необработанные данные 296
7.9.22.2.2. Расчеты 296

8. Эксплуатация 301

8.1. Измерения — подробная информация 301
8.1.1. Хронометраж — подробная информация 301
8.1.1.1. Временные метки 301
8.1.2. Аналоговые измерения — подробная информация 303
8.1.2.1. Измерения напряжения — подробная информация 303
8.1.2.1.1. Механика измерения напряжения 303
8.1.2.1.2. Ограничения измерения напряжения . 306
8.1.2.1.3. Качество измерения напряжения 309
8.1.2.2. Измерения термопар — подробная информация .. 325
8.1.2.2.1. Анализ погрешностей термопары 325
8.1.2.2.2. Использование внешнего свободного спая 334
8.1.2.3. Измерения тока — подробная информация 335
8.1.2.4. Измерения сопротивления — подробная информация 335
8.1.2.4.1. Возбуждение переменным током 339
8.1.2.4.2. Измерения сопротивления — погрешность 339
8.1.2.5. Тензометрические измерения — подробная информация 340
8.1.2.6. Автоматическая калибровка — подробная информация 342
8.1.2.6.1. Процесс автоматической калибровки. 342
8.1.3. Импульсные измерения — подробная информация 347
8.1.3.1. Клеммы для импульсного измерения 350
8.1.3.2. Низкоуровневые измерения с переменным током — подробная информация 350
8.1.3.3. Высокочастотные измерения 351
8.1.3.3.1. Разрешение по частоте 351
8.1.3.3.2. Измерения частоты: вопросы и ответы 352
8.1.3.4. Измерения для переключения-замыкания и открытого коллектора 353
8.1.3.5. Время крайних значений 353
8.1.3.6. Подсчет крайних значений 354
8.1.3.7. Импульсные измерения: советы 354
8.1.3.7.1. Условия появления значения NAN при TimerIO() 357
8.1.3.7.2. Входные фильтры и затухание сигнала 357
8.1.4. Среднее значение за период — подробная информация 358

8.1.5. Вибрационные струнные измерения — подробная информация 359
8.1.5.1. Измерения временной области 360
8.1.6. Снятие показаний интеллектуальных датчиков — подробная информация 360
8.1.6.1. RS-232 и TTL 360
8.1.6.2. Поддержка датчика SDI-12 — подробная информация 361
8.1.7. Калибровка на месте — общие сведения 361
8.1.8. Эффекты подключения 362
8.1.8.1. Кабели аналогового датчика 362
8.1.8.2. Импульсные датчики 362
8.1.8.3. Датчики RS-232 362
8.1.8.4. Датчики SDI-12 362
8.1.9. Синхронизация измерений 363
8.2. Внешнее оборудование для измерения и контроля — подробная информация 364
8.2.1. Модули аналогового ввода 364
8.2.2. Модули импульсного ввода 365
8.2.2.1. Модули ввода по низкоуровневому переменному току — общие сведения 365
8.2.3. Модули последовательного ввода/вывода — подробная информация 365
8.2.4. Терминальные модули ввода 365
8.2.5. Вибрационные струнные модули 365
8.2.6. Модули аналогового вывода 365
8.2.7. Модули управления ПЛК — общие сведения 366
8.2.7.1. Клеммы, настроенные на управление 366
8.2.7.2. Реле и драйверы реле 367
8.2.7.3. Компонентные реле 367
8.3. Память 368
8.3.1. Накопитель 368
8.3.1.1. Диски памяти — на плате 372
8.3.1.1.1. SRAM таблицы данных 372
8.3.1.1.2. Диск CPU: 372
8.3.1.1.3. Диск USR: 373
8.3.1.1.4. Диск USB: 373
8.3.1.2. Карта памяти (диск CRD: диск) — подробная информация 374
8.3.2. Форматы файлов данных 375
8.3.3. Сброс CR1000 379
8.3.3.1. Полный сброс памяти 379
8.3.3.2. Сброс отправки программы 379
8.3.3.3. Ручной сброс таблицы данных 380
8.3.3.4. Форматирование дисков 380
8.3.4. Управление файлами 380
8.3.4.1. Атрибуты файлов 381
8.3.4.2. Диспетчер файлов 382
8.3.4.3. Сохранение данных 383
8.3.4.4. Файл Powerup.ini — подробная информация 384
8.3.4.4.1. Создание и редактирование Powerup.ini 385
8.3.4.5. Управление файлами — вопрос-ответ 387
8.3.5. Имена файлов 387
8.3.6. Ошибки файловой системы 387
8.3.7. Память — вопрос-ответ 389
8.4. Сбор и обмен данными — подробная информация 389
8.4.1. Протоколы 390
8.4.2. Сохранение полосы пропускания 390
8.4.3. Инициация связи (обратный вызов) 390
8.5. Связь PakBus® — подробная информация 391
8.5.1. PakBus-адреса 391

8.5.2. Узлы: оконечные узлы и маршрутизаторы....392
8.5.2.1. Конфигурация маршрутизаторов и оконечных узлов....392
8.5.3. Соединение узлов PakBus: обнаружение соседей....394
8.5.3.1. Hello-сообщение....394
8.5.3.2. Сигнал маяка....394
8.5.3.3. Hello-запрос....394
8.5.3.4. Список соседей....394
8.5.3.5. Регулирование соединений....394
8.5.3.6. Поддержка соединений....395
8.5.4. Диагностика сетей PakBus....395
8.5.4.1. Целостность соединения....395
8.5.4.1.1. Автоматическая регулировка размера пакета....395
8.5.4.2. Ping (PakBus)....396
8.5.4.3. Поток трафика....396
8.5.5. Конфигурация карты сети LoggerNet....396
8.5.6. Пример ЛВС на основе протокола PakBus....398
8.5.6.1. Кабельная система ЛВС....398
8.5.6.2. Настройка ЛВС....399
8.5.6.3. Настройка LoggerNet....401
8.5.7. Фильтры маршрутов....403
8.5.8. PakBusRoutes....403
8.5.9. Соседи....404
8.5.10. Шифрование PakBus....404
8.6. Альтернативная связь — подробная информация....405
8.6.1. DNP3 — подробная информация....406
8.6.1.1. DNP3 — введение....406
8.6.1.2. Программирование для DNP3....406
8.6.1.2.1. Объявления (программирование DNP3)....406
8.6.1.2.2. Команды CRBasic (DNP3)....407
8.6.1.2.3. Программирование для DNP3 — сбор данных....408
8.6.2. Modbus — подробная информация....409
8.6.2.1. Терминология Modbus....410
8.6.2.1.1. Глоссарий терминов Modbus....410
8.6.2.2. Программирование для Modbus....411
8.6.2.2.1. Объявления (программирование для Modbus)....411
8.6.2.2.2. Команды CRBasic (Modbus)....412
8.6.2.2.3. Адресация (ModbusAddr)....412
8.6.2.2.4. Поддерживаемые функциональные коды Modbus....413
8.6.2.2.5. Считывание регистров Modbus в обратном формате....413
8.6.2.3. Диагностика (Modbus)....413
8.6.2.4. Modbus over IP....414
8.6.2.5. Modbus — вопрос-ответ....414
8.6.2.6. Преобразование 16-битных длинных чисел Modbus в 32-битные....414
8.6.3. TCP/IP — подробная информация....415
8.6.3.1. PakBus через TCP/IP и обратный вызов....416
8.6.3.2. Веб-сервер HTTP по умолчанию....416
8.6.3.3. Пользовательский веб-сервер HTTP....417
8.6.3.4. Сервер FTP....420
8.6.3.5. FTP-клиент....420
8.6.3.6. Telnet....420
8.6.3.7. SNMP....420
8.6.3.8. Ping (IP)....421
8.6.3.9. Микро-последовательный сервер....421
8.6.3.10. Modbus TCP/IP....421

8.6.3.11. DHCP 421
8.6.3.12. DNS 421
8.6.3.13. SMTP 421
8.6.3.14. Web API 421
8.6.3.14.1. Аутентификация 422
8.6.3.14.2. Синтаксис команды 423
8.6.3.14.3. Синтаксис времени 425
8.6.3.14.4. Управление данными — команда BrowseSymbols 425
8.6.3.14.5. Управление данными — команда DataQuery 429
8.6.3.14.6. Управление — команда SetValueEx 434
8.6.3.14.7. Функции часов — команда ClockSet 437
8.6.3.14.8. Функции часов — команда ClockCheck 438
8.6.3.14.9. Управление файлами — отправка файла в регистратор данных 440
8.6.3.14.10. Управление файлами — команда FileControl 442
8.6.3.14.11. Управление файлами — команда ListFiles 443
8.6.3.14.12. Управление файлами — команда NewestFile 447
8.7. ПО поддержки регистратора данных — подробная информация 448
8.8. Экранная клавиатура — подробная информация 449
8.8.1. Дисплей данных 452
8.8.1.1. Таблицы и графики в реальном масштабе времени 453
8.8.1.2. Real-Time Custom (Пользовательская настройка реального времени) 453
8.8.1.3. Таблицы окончательных данных 455
8.8.2. Run/Stop Program (Запуск/остановка программы) 456
8.8.3. Отображение файлов 457
8.8.3.1. Файл: Edit (Файл: Правка) 457
8.8.4. Отображение PCCard (карт памяти) 459
8.8.5. Порты и состояние 460
8.8.6. Настройки 460
8.8.6.1. Настройка времени/даты 461
8.8.6.2. Настройки PakBus 461
8.8.7. Конфигурация дисплея 461
8.9. Сжатие программных файлов и файлов ОС — вопрос-ответ 461
8.10. Карты памяти и номера записей 464
8.11. Безопасность — подробная информация 465
8.11.1. Уязвимости 466
8.11.2. Блокировка посредством кодов доступа 467
8.11.2.1. Обход блокировки посредством кода доступа 468
8.11.3. Пароли 468
8.11.3.1. .csipasswd 468
8.11.3.2. Команды PakBus 468
8.11.3.3. Команды TCP/IP 469
8.11.3.4. Настройки — пароли 469
8.11.4. Шифрование файла 469
8.11.5. Шифрование связи 469
8.11.6. Скрытие файлов 469
8.11.7. Подписи 470

9. Обслуживание — подробная информация 471

9.1. Защита от влаги — подробная информация 471
9.2. Замена внутренней батареи 471
9.3. Порядок заводской калибровки или ремонта 474

10. Диагностика 477

10.1. Диагностика — средства первой необходимости 477
10.2. Диагностика — основная процедура 477
10.3. Диагностика — источники ошибок....477
10.4. Диагностика — таблица состояния 479
10.5. Программирование 479
10.5.1. Программа не компилируется 479
10.5.2. Программа компилируется / не работает корректно 479
10.5.3. NAN и ±INF 480
10.5.3.1. Измерения и NAN 480
10.5.3.1.1. Измерение напряжения....480
10.5.3.1.2. Измерения SDI-12 480
10.5.3.2. Математика с плавающей запятой, NAN и ±INF ..480
10.5.3.3. Типы данных, NAN и ±INF 481
10.5.3.4. Обработка выхода и NAN 482
10.5.4. Таблица состояния, как средство отладки....483
10.5.4.1. CompileResults....483
10.5.4.2. SkippedScan....485
10.5.4.3. SkippedSlowScan 486
10.5.4.4. SkippedRecord....486
10.5.4.5. ProgErrors....486
10.5.4.6. MemoryFree....486
10.5.4.7. VarOutOfBounds 486
10.5.4.8. Ошибки сторожевых схем....486
10.5.4.8.1. Ошибки сторожевой схемы таблицы состояния 487
10.5.4.8.2. Файл Watchdoginfo.txt....487
10.6. Диагностика — операционные системы....488
10.7. Диагностика — ошибки автоматической калибровки....488
10.8. Связь 488
10.8.1. RS-232....488
10.8.2. Связь с несколькими ПК 489
10.8.3. Ошибки коммуникационной памяти 489
10.8.3.1. CommsMemFree(1)....489
10.8.3.2. CommsMemFree(2)....490
10.8.3.3. CommsMemFree(3)....491
10.9. Диагностика — источники электропитание....492
10.9.1. Диагностика источников электропитания — общие сведения....492
10.9.2. Диагностика источников электропитания — примеры — 8 10 30 492
10.9.3. Диагностика источников электропитания — процедуры 493
10.9.3.1. Тестирование батареи....493
10.9.3.2. Проверка солнечной панели с регулятором зарядки 494
10.9.3.3. Регулятор зарядки с испытанием трансформатора496
10.9.3.4. Настройка зарядного напряжения 497
10.10. Режим терминала 499
10.10.1. Последовательная ретрансляция и просмотр трафика 501
10.11. Журналы 502
10.12. Поиск и устранение неисправностей — восстановление данных 502

11. Глоссарий 505

11.1. Термины....505
11.2. Концепции....531
11.2.1. Погрешность, точность и разрешение 531

12. Заявления о принадлежности 533

Приложения

A. Команды программирования на CRBasic..... 537

A.1 Объявления в программе 537
A.1.1 Объявление и модификаторы переменных 538
A.1.2 Объявление констант 539
A.2 Объявление таблиц данных 540
A.2.1 Модификаторы таблицы данных 540
A.2.2 Адресаты данных 541
A.2.3 Обработка вывода для записи в память окончательных данных 542
A.2.3.1 Один источник данных 542
A.2.3.2 Несколько источников данных 544
A.3 Однократное выполнение при компиляции 544
A.4 Команды управления программой 545
A.4.1 Обычные команды управления программой 545
A.4.2 Дополнительные команды управления программой 548
A.5 Команды измерения 550
A.5.1 Диагностика 550
A.5.2 Напряжение 551
A.5.3 Термопары 551
A.5.4 Измерения с помощью моста сопротивления 551
A.5.5 Возбуждение 552
A.5.6 Импульс и частота 553
A.5.7 Цифровой ввод-вывод 554
A.5.7.1 Управление 554
A.5.7.2 Измерение 555
A.5.8 Поддержка датчика SDI-12 — Команды 555
A.5.9 Специализированные датчики 556
A.5.9.1 Сеть беспроводных датчиков 558
A.5.10 Поддержка периферийных устройств 559
A.6 Управление ПЛК — Команды 562
A.7 Команды обработки и математические команды 563
A.7.1 Математические операторы 563
A.7.2 Арифметические операторы 563
A.7.3 Побитовые операции 564
A.7.4 Составные операторы присваивания 565
A.7.5 Логические операторы 565
A.7.6 Тригонометрические функции 566
A.7.6.1 Встроенные тригонометрические функции 566
A.7.6.2 Производные тригонометрические функции 568
A.7.7 Арифметические функции 568
A.7.8 Интегрированная обработка данных 570
A.7.9 Пространственная обработка данных 571
A.7.10 Другие функции 572
A.7.10.1 Гистограммы 573
A.8 Строковые функции 574
A.8.1 Строковые операции 574
A.8.2 Строковые команды 575
A.9 Регистрация времени — команды 578
A.10 Команды модема с поддержкой голосовых сообщений 580
A.11 Пользовательские меню — команды 581
A.12 Последовательный ввод/вывод 583
A.13 Одноранговая сеть PakBus® 584
A.14 Управление переменными 589

A.15 Управление файлами 589
A.16 Доступ и управление таблицами данных 592
A.17 Команды управления TCP/IP 593
A.18 Управление модемом 597
A.19 SCADA 597
A.20 Функции калибровки 598
A.21 Спутниковые системы 599
A.21.1 Спутниковая система Argos 599
A.21.2 Спутниковая система GOES 600
A.21.3 Спутниковая система OMNISAT 601
A.21.4 Спутниковая система INMARSAT-C 601
A.22 Пользовательские функции 602

B. Информация о состоянии, настройках и таблицах данных (Status/Settings/DTI) 603
B.1 Каталоги параметров состояния/настроек/таблиц 604
B.2 Описание параметров состояния/настроек/таблиц (в алфавитном порядке) 611

C. Схемы выводов последовательных портов 633
C.1 Порт CS I/O 633
C.2 Порт RS-232 633
C.2.1 Схемы выводов 633
C.2.2 Режимы питания 634

D. Таблица кодов ASCII / ANSI 637

E. Формат данных FP2 641

F. Порядок следования байтов 643

G. Перечни продуктов для поддержки 645
G.1 Регистраторы данных — список 645
G.2 Внешнее оборудование для измерения и контроля — списки 645
G.3 Списки входных модулей датчиков 646
G.3.1 Список модулей аналогового ввода 646
G.3.2 Список модулей импульсного входа 646
G.3.3 Список модулей последовательного ввода/вывода 646
G.3.4 Список модулей входа струнных датчиков 647
G.3.5 Список пассивных преобразователей сигнала 647
G.3.5.1 Список модулей резистивного моста TIM 647
G.3.5.2 Список модулей делителей напряжения 647
G.3.5.3 Список модулей токовых шунтов 647
G.3.5.4 Список ограничителей переходного напряжения 648
G.3.6 Список крышек клеммных колодок 648
G.4 Модули управления ПЛК — списки 648
G.4.1 Список модулей цифрового ввода/вывода 648
G.4.2 Список модулей непрерывного аналогового вывода (Continuous Analog Output, CAO) 649
G.4.3 Управляющие цепи реле — список 649
G.4.4 Список модулей возбуждения 649
G.5 Датчики — списки 649
G.5.1 Список типов проводных датчиков 650
G.5.2 Список беспроводных сетевых датчиков 650

G.6 Внешнее оборудование для сбора и обмена данными — списки 651
G.6.1 Дисплеи с клавиатурой — список 651
G.6.2 Список кабельных коммуникационных устройств с одиночным подключением 652
G.6.3 Список кабельных сетевых устройств 652
G.6.4 Каналы TCP/IP — список 652
G.6.5 Список телефонных модемов 652
G.6.6 Список радиомодулей для частных сетей 653
G.6.7 Список спутниковых приемопередатчиков 653
G.7 Накопители данных — список 653
G.8 ПО поддержки регистратора данных — списки 654
G.8.1 Список ПО начального уровня 654
G.8.2 ПО поддержки регистратора данных — списки 654
G.8.2.1 Пакет LoggerNet — список 655
G.8.3 Список программных средств 656
G.8.4 Список пакетов для разработки ПО 656
G.9 Источники питания — продукты 657
G.9.1 Список совмещенных батарей/регуляторов 657
G.9.2 Список батарей 658
G.9.3 Список регуляторов 658
G.9.4 Список первичных источников питания 658
G.9.5 Список комплектов источников питания 24 В пост. тока 659
G.10 Корпуса — продукты 659
G.11 Списки штативов, башен и креплений 659
G.12 Список корпусов 660

Алфавитный указатель 661

Список рисунков.

Рисунок 1. Компоненты системы сбора данных 42
Рисунок 2. Коммутационная панель 44
Рисунок 3. Соединения питания и последовательной передачи данных 48
Рисунок 4. Главное окно PC200W 49
Рисунок 5. Список датчиков температуры в Short Cut 51
Рисунок 6. Подключение термопары в Short Cut (необходим новый рисунок для CR6: 1H = U1, 1L = U2 .....)52
Рисунок 7. Экран вывода данных Short Cut 53
Рисунок 8. Экран вывода данных Short Cut 54
Рисунок 9. Подтверждение компиляции в Short Cut 54
Рисунок 10. Главное окно PC200W 55
Рисунок 11. Экран мониторинга данных PC200W — таблица Public 56
Рисунок 12. Экран мониторинга данных PC200W — таблицы Public и OneMin 56
Рисунок 13. Экран сбора данных PC200W 57
Рисунок 14. Утилита просмотра данных PC200W 58
Рисунок 15. Таблица просмотра данных PC200W 58
Рисунок 16. График просмотра данных PC200W 60
Рисунок 17. Система сбора данных — общие сведения 62
Рисунок 18. Подключение аналогового датчика к несимметричному каналу 1 64
Рисунок 19. Подключение аналогового датчика к дифференциальному каналу 1 64
Рисунок 20. Упрощенная последовательность дифференциального измерения напряжения 66
Рисунок 21. Пример полу-мостового подключения — потенциометр флюгера 67

Рисунок 22. Пример полного мостового подключения — измерительный преобразователь давления 68
Рисунок 23. Типы выходных сигналов импульсного датчика 69
Рисунок 24. Пример импульсного входа — анемометр 70
Рисунок 25. Настраиваемые клеммы для ввода RS-232 73
Рисунок 26. Использование RS-232 и цифрового ввода/вывода при чтении устройств RS-232 73
Рисунок 27. Коммутационная панель 76
Рисунок 28. Контроль и управление клеммами C 79
Рисунок 29. Дисплей с клавиатурой CR1000KD 83
Рисунок 30. Пример пользовательского меню 84
Рисунок 31. Защитный корпус 100
Рисунок 32. Подключение к источнику питания автомобиля 102
Рисунок 33. Схема заземления 107
Рисунок 34. Схема грозозащиты 108
Рисунок 35. Модель контура заземления с резистивным датчиком 110
Рисунок 36. Служебная программа конфигурации устройства (DevConfig) 112
Рисунок 37. Настройка планировщика сети 113
Рисунок 38. Общая информация о конфигурации CR1000 122
Рисунок 39. Окно редактора CRBasic для отправки контрольного файла 127
Рисунок 40. Параметры «включаемого файла» в DevConfig 149
Рисунок 41. Параметры «включаемого файла» в PakBusGraph 149
Рисунок 42. Функциональная схема приоритета сканирования в последовательном режиме 158
Рисунок 43. Пример пользовательского меню — главный экран 183
Рисунок 44. Пример пользовательского меню — окно просмотра данных 183
Рисунок 45. Пример пользовательского меню — подменю создания примечаний 184
Рисунок 46. Пример пользовательского меню — список выбора предварительно определенных примечаний 184
Рисунок 47. Пример пользовательского меню — окно для свободного ввода примечаний 184
Рисунок 48. Пример пользовательского меню — окно подтверждения/удаления примечаний 184
Рисунок 49. Пример пользовательского меню — подменю управления 185
Рисунок 50. Пример пользовательского меню — список выбора контрольных светодиодов 185
Рисунок 51. Пример пользовательского меню — список выбора логических контрольных светодиодов 185
Рисунок 52. Скользящее среднее частотной характеристики 194
Рисунок 53. Затухание скользящего среднего значения сигнала 195
Рисунок 54. Данные программы TrigVar 196
Рисунок 55. Пример переключения аварийных сигналов с битовым сдвигом 199
Рисунок 56. Данные типа Bool8 из примера битового сдвига (цифровой монитор) 199
Рисунок 57. Данные типа Bool8 из примера битового сдвига (файл данных ПК) 200
Рисунок 58. Четвертьмостовой тензодатчик с шунтом на базе RC-резистора 225
Рисунок 59. Начало калибровки шунта тензодатчика 226
Рисунок 60. Окончание калибровки шунта тензодатчика 227
Рисунок 61. Начало нулевой процедуры 227
Рисунок 62. Окончание нулевой процедуры 227
Рисунок 63. PT100 в четырехжильном полумостовом подключении 240
Рисунок 64. PT100 в трехжильном полумостовом подключении 242
Рисунок 65. PT100 в четырехжильном подключении с полным мостом 244
Рисунок 66. HyperTerminal: описание нового подключения 256
Рисунок 67. HyperTerminal: настройки подключения 257
Рисунок 68. HyperTerminal: вкладка настроек COM-порта 257
Рисунок 69. HyperTerminal: настройка ASCII 258

Рисунок 70. HyperTerminal: пример текстового файла для отправки ... 258
Рисунок 71. HyperTerminal: пример файла захвата текста ... 259
Рисунок 72. Вход в прозрачный режим SDI-12 ... 268
Рисунок 73. Предварительно сконфигурированная домашняя страница HTML ... 291
Рисунок 74. Домашняя страница, созданная с помощью команды WebPageBegin() ... 292
Рисунок 75. Пользовательская веб-страница отслеживания данных ... 293
Рисунок 76. Входящие выборочные векторы ... 298
Рисунок 77. Графическое изображение среднего вектора ветра ... 299
Рисунок 78. Стандартное отклонение направления ... 300
Рисунок 79. Упрощенная последовательность измерения напряжения ... 306
Рисунок 80. Программируемый входной усилитель (PGIA) ... 306
Рисунок 81. PGIA с разложением входящего сигнала ... 311
Рисунок 82. Пример диапазона погрешности измерения напряжения, включая влияние процента от показаний и смещения, для дифференциального измерения с инверсией входящего сигнала при температуре от 0 до 40 °C ... 314
Рисунок 83. Методы подавления помех в шине питания переменного тока ... 316
Рисунок 84. Входное напряжение: возрастание и затухание перехода ... 318
Рисунок 85. Время успокоения для датчика давления ... 321
Рисунок 86. Сводка по погрешностям температуры панели ... 329
Рисунок 87. Градиенты температуры панели (по возрастанию температуры) ... 329
Рисунок 88. Градиенты температуры панели (по убыванию температуры) ... 330
Рисунок 89. Расчет погрешности на входе ... 332
Рисунок 90. Схема устройства термопары ... 337
Рисунок 91. Типы выходных сигналов импульсного датчика ... 350
Рисунок 92. Импульсный датчик переключения-замыкания ... 350
Рисунок 93. Настраиваемые клеммы для импульсного входа ... 351
Рисунок 94. Сокращение амплитуды волны подсчета импульсов (до и после применения фильтра постоянной времени 1 мкс) ... 360
Рисунок 95. Входной контур преобразования для поиска среднего значения за период ... 361
Рисунок 96. Струнный датчик ... 362
Рисунок 97. Контур ограничения входящего сигнала RS-232 на клемме C до 5 В пост. тока ... 363
Рисунок 98. Резистор для ограничения тока в контуре дождемера ... 364
Рисунок 99. Подача тока с клемм C, настроенных на управление ... 369
Рисунок 100. Цепь драйвера реле и реле ... 370
Рисунок 101. Переключение электропитания без реле ... 370
Рисунок 102. Сетевая адресация PakBus ... 394
Рисунок 103. Схема одноуровневого подчинения ... 399
Рисунок 104. Схема многоуровневого подчинения ... 399
Рисунок 105. Конфигурация и кабельная система ЛВС PakBus ... 400
Рисунок 106. DevConfig — закладка Deployment ... 401
Рисунок 107. DevConfig — закладка Deployment | ComPorts Settings (Настройки COM-портов) ... 402
Рисунок 108. DevConfig — закладка Deployment | Advanced (Расширенные настройки) ... 402
Рисунок 109. Настройка карты сети LoggerNet: COM-порт ... 403
Рисунок 110. Настройка карты сети LoggerNet: PakBusPort ... 404
Рисунок 111. Настройка карты сети LoggerNet: Регистраторы данных ... 404
Рисунок 112. Предварительно сконфигурированная домашняя страница HTML ... 419
Рисунок 113. Домашняя страница, созданная с помощью команды WebPageBegin() ... 420
Рисунок 114. Пользовательская веб-страница числового отслеживания ... 420
Рисунок 115. Использование клавиатуры / дисплея ... 453

Рисунок 116. Отображение данных с помощью клавиатуры / дисплея 454
Рисунок 117. Таблицы и графики в реальном масштабе времени 455
Рисунок 118. Пользовательская настройка реального времени 456
Рисунок 119. Таблицы окончательных данных 457
Рисунок 120. Запуск/остановка программы 458
Рисунок 121. Просмотр файла 459
Рисунок 122. Файл: редактирование 460
Рисунок 123. Отображение PCCard (карт CF) 461
Рисунок 124. Состояние клемм (портов) C 462
Рисунок 125. Настройки 462
Рисунок 126. Конфигурация дисплея 463
Рисунок 127. Ослабьте винты крепления 474
Рисунок 128. Отведите край от панели 475
Рисунок 129. Отверните гайки, чтобы разобрать коробку 475
Рисунок 130. Снимите и замените батарею 476
Рисунок 131. Потенциометр R3 в зарядном устройстве/регуляторе PS100 или CH100 501
Рисунок 132. Закладка Terminal утилиты DevConfig 503
Рисунок 133. Взаимосвязь погрешности, точности и разрешения 534

Список таблиц

Таблица 1. Пример выбора настроек в мастере EZSetup программы PC200W 49
Таблица 2. Дифференциальные и несимметричные входные клеммы ... 65
Таблица 3. Клеммы импульсного входа и измерения 69
Таблица 4. Описание клемм коммутационной панели CR1000 77
Таблица 5. Источник тока и ограничения при снижении 103
Таблица 6. Параметры состояния/настроек/таблиц: Узлы доступа 115
Таблица 7. Общие действия и инструменты для конфигурации 117
Таблица 8. Структура программы CRBasic 123
Таблица 9. Варианты отправки программы, при которых происходит сброс памяти* 127
Таблица 10 Структуры таблицы данных 128
Таблица 11. Типы данных в памяти переменных 130
Таблица 12. Типы данных в памяти окончательных данных 131
Таблица 13. Форматы ввода чисел в CRBasic 139
Таблица 14. Типовая таблица данных 141
Таблица 15. Строка среды TOA5 141
Таблица 16. Варианты параметров пропуска в команде DataInterval() . 145
Таблица 17. Задачи программы 152
Таблица 18. Приоритеты задач в поточном режиме 153
Таблица 19. Команды, управляющие скоростью выполнения программы 154
Таблица 20. Правила для имен 159
Таблица 21. Двоичные условия TRUE (ИСТИНА) и FALSE (ЛОЖЬ) . 165
Таблица 22. Примеры логических выражений 165
Таблица 23. Сокращения, используемые при обработке данных 168
Таблица 24. Пример CRBasic. Выражение для назначения массива: суммирование столбцов и строк 190
Таблица 25. Пример CRBasic. Выражение для назначения массива: преобразование массива 190
Таблица 26. Пример CRBasic. Выражение для назначения массива: сравнение / логическая оценка 191
Таблица 27. Пример CRBasic. Выражение для назначения массива: заполнение массива 192
Таблица 28. Коды FieldCal() 212
Таблица 29. Отчет о калибровке датчика относительной влажности ... 214
Таблица 30. Отчет о калибровке датчика содержания соли 216
Таблица 31 Отчет о калибровке гидрометра 218
Таблица 32. Отчет о калибровке датчика состава воды 221
Таблица 33. Частота аналоговых измерений напряжения — краткое описание 230

Таблица 34. Параметры для аналогового пакетного режима (от 601 до 2000 Гц) 234
Таблица 35. PRTCalc(): датчик с кодом типа 1 236
Таблица 36. PRTCalc(): датчик с кодом типа 2 237
Таблица 37. PRTCalc(): датчик с кодом типа 3 237
Таблица 38. PRTCalc(): датчик с кодом типа 4 237
Таблица 39. PRTCalc(): датчик с кодом типа 5 238
Таблица 40. PRTCalc(): датчик с кодом типа 6 238
Таблица 41. Эквиваленты ASCII / ANSI 245
Таблица 42. Последовательные порты CR1000 247
Таблица 43. Команды SDI-12 для прозрачного режима 269
Таблица 44. Пример CRBasic: настройка датчика SDI-12 — результаты 281
Таблица 45. Пример профиля энергопотребления для сети измерительных приборов SDI-12 282
Таблица 46. Строковые операторы 283
Таблица 47. Примеры конкатенации строк 284
Таблица 48. Примеры нулевого символа в строке 285
Таблица 49. Извлечение строковых символов 286
Таблица 50. Примеры использования ASCII- / ANSII-кодов 287
Таблица 51. Примеры форматирования строк 287
Таблица 52. Форматирование шестнадцатеричных переменных — примеры 288
Таблица 53. WindVector(): опции OutputOpt 296
Таблица 54. Параметры CRBasic, варьирующие последовательность и временные рамки измерений 307
Таблица 55. Аналоговое напряжение: входные диапазоны и опции 309
Таблица 56 Погрешность измерения аналогового напряжения1 313
Таблица 57. Смещения измерения аналогового напряжения 313
Таблица 58. Разрешение измерения аналогового напряжения 313
Таблица 59. Интеграция аналогового измерения 316
Таблица 60. Подавление помех переменного тока для слабых сигналов1 317
Таблица 61. Подавление помех переменного тока для сильных сигналов1 317
Таблица 62. Время успокоения для измерения CRBasic 318
Таблица 63. Первые шесть значений данных времени успокоения 321
Таблица 64. Опция кода диапазона: C, перенапряжение 322
Таблица 65. Опции компенсации напряжения смещения 325
Таблица 66. Пределы погрешности для провода термопары (свободный спай при 0 °C) 331
Таблица 67. Диапазон напряжения для максимального разрешения термопары 331
Таблица 68. Пределы погрешности для полиномов термопары в CR1000 334
Таблица 69. Диапазон компенсации опорной температуры и погрешность 335
Таблица 70. Примеры погрешности термопары 336
Таблица 71. Контуры моста сопротивления с напряжением возбуждения 339
Таблица 72. Погрешность при логометрическом измерении сопротивления 342
Таблица 73. Команда StrainCalc(): уравнения 343
Таблица 74. Усиления и смещения автокалибровки 346
Таблица 75. Команда Calibrate(): результаты 347
Таблица 76. Импульсные измерения: клеммы и программирование 351
Таблица 77. Пример. E для входящего сигнала частотой 10 Гц 354
Таблица 78. Сравнение разрешений по частоте 354
Таблица 79. Переключения-замыкания и открытые коллекторы для P-клемм 357
Таблица 80. Переключения-замыкания и открытые коллекторы для C-клемм 357
Таблица 81. Три различия в технических характеристиках между клеммами P и C 358

Таблица 82. Постоянные времени (τ) 359
Таблица 83. Амплитуда низкоуровневого переменного тока и максимальная измеряемая частота 360
Таблица 84. Выделение памяти в CR1000 371
Таблица 85. Основная память CR1000 373
Таблица 86. Диски памяти 374
Таблица 87. Состояния карты памяти 377
Таблица 88. Команда TableFile() – форматы файлов данных 378
Таблица 89. Функции управления файлами 382
Таблица 90. Атрибуты файлов CR1000 383
Таблица 91. Опции сохранения данных 386
Таблица 92. Команды и приложения файла Powerup.ini 388
Таблица 93 Пример файла Powerup.ini. Формат кода и синтаксические правила 388
Таблица 94. Пример файла Powerup.ini. Выполнение программы при включении электропитания 388
Таблица 95. Пример файла Powerup.ini. Форматировать диск USR 389
Таблица 96. Пример файла Powerup.ini. Обновление ОС при включении электропитания 389
Таблица 97. Пример файла Powerup.ini. Выполнить программу с диска USB 389
Таблица 98. Пример файла Powerup.ini. Выполнять программу всегда, стереть данные 389
Таблица 99. Пример файла Powerup.ini. Выполнить программу сейчас, стереть данные 389
Таблица 100. Коды ошибок файловой системы 390
Таблица 101 Конфигурация маршрутизаторов и оконечных узлов PakBus 395
Таблица 102. Шкала пропускной способности соединения в сети PakBus 398
Таблица 103. Пример ЛВС PakBus — настройки связи регистратора данных 403
Таблица 104. Номера портов маршрутизатора 405
Таблица 105. Реализация DNP3 — типы данных, подлежащих хранению в таблицах общего пользования для групп объектов 409
Таблица 106. Эквивалентные термины в протоколах Modbus и Campbell Scientific 412
Таблица 107. Порты, флаги и переменные CRBasic и регистры Modbus 414
Таблица 108. Поддерживаемые функциональные коды Modbus 415
Таблица 109. Команды, параметры и аргументы API 425
Таблица 110. Параметры команды BrowseSymbols API 427
Таблица 111. Ответ на команду BrowseSymbols API 428
Таблица 112. Параметры команды DataQuery API 431
Таблица 113. Параметры команды SetValueEx API 437
Таблица 114. Ответ на команду SetValue API 437
Таблица 115. Параметры команды ClockSet API 439
Таблица 116. Ответ на команду ClockSet API 439
Таблица 117. Параметры команды ClockCheck API 440
Таблица 118. Ответ на команду ClockCheck API 441
Таблица 119. Параметры запроса Curl HTTPPut 442
Таблица 120. Параметры API команды FileControl 444
Таблица 121. Ответ на команду FileControl API 445
Таблица 122. Параметры API команды ListFiles 446
Таблица 123. Ответ на команду ListFiles API 446
Таблица 124. Параметры API команды NewestFile 450
Таблица 125. Специальные функции клавиш электронной клавиатуры 451
Таблица 126 Типовые результаты сжатия файлов в формате Gzip 465
Таблица 127. Технические характеристики внутренней литиевой батареи 474
Таблица 128. Математические выражения и результаты CRBasic 483
Таблица 129. Типы данных переменных и конечной памяти с NAN и ±INF 484
Таблица 130. Примеры предупреждающих сообщений 486

Таблица 131. Пример значений по умолчанию CommsMemFree(l) и их использование, TLS не активен 492
Таблица 132. Пример значений по умолчанию CommsMemFree(l) и их использование, TLS не активен 492
Таблица 133. Команды терминала CR1000 502
Таблица 134. Местоположения журналов 504
Таблица 135. Команда Program Send 524
Таблица 136. Арифметические операторы 563
Таблица 137. Составные операторы присваивания 565
Таблица 138. Производные тригонометрические функции 568
Таблица 139. Строковые операции 574
Таблица 140. Скорости передачи данных для асинхронных портов 588
Таблица 141. Параметры состояния/настроек/таблиц: Узлы доступа 603
Таблица 142. Параметры состояния/настроек/таблиц: каталоги 604
Таблица 143. Параметры состояния/настроек/таблиц: часто используемые 604
Таблица 144. Параметры состояния/настроек/таблиц: список ключевых слов в алфавитном порядке 605
Таблица 145. Параметры состояния/настроек/таблиц: элементы таблицы состояния на дисплее с клавиатурой CR1000KD 606
Таблица 146. Параметры состояния/настроек/таблиц: настройки (общие) на дисплее с клавиатурой CR1000KD 606
Таблица 147. Параметры состояния/настроек/таблиц: настройки (последовательного порта) на дисплее с клавиатурой CR1000KD 607
Таблица 148. Параметры состояния/настроек/таблиц: настройки (TCP/IP) на дисплее с клавиатурой CR1000KD 607
Таблица 149. Параметры состояния/настроек/таблиц: настройки только в редакторе настроек 607
Таблица 150. Параметры состояния/настроек/таблиц: ключевые слова таблицы информации о таблицах данных (DTI) 607
Таблица 151. Параметры состояния/настроек/таблиц: автоматическая калибровка 608
Таблица 152. Параметры состояния/настроек/таблиц: связь, общая информация 608
Таблица 153. Параметры состояния/настроек/таблиц: связь, протокол PakBus 608
Таблица 154. Параметры состояния/настроек/таблиц: связь, протокол TCP_IP I 608
Таблица 155. Параметры состояния/настроек/таблиц: связь, протокол TCP_IP II 608
Таблица 156. Параметры состояния/настроек/таблиц: связь, протокол TCP_IP III 609
Таблица 157. Параметры состояния/настроек/таблиц: программа I на CRBasic 609
Таблица 158. Параметры состояния/настроек/таблиц: программа II на CRBasic 609
Таблица 159. Параметры состояния/настроек/таблиц: данные 609
Таблица 160. Параметры состояния/настроек/таблиц: память 609
Таблица 161. Параметры состояния/настроек/таблиц: прочее 609
Таблица 162. Параметры состояния/настроек/таблиц: устаревшие 609
Таблица 163. Параметры состояния/настроек/таблиц: операционная система и контроль версий аппаратного обеспечения 610
Таблица 164. Параметры состояния/настроек/таблиц: показатели электропитания 610
Таблица 165. Параметры состояния/настроек/таблиц: безопасность 610
Таблица 166. Параметры состояния/настроек/таблиц: сигнатуры 610
Таблица 167. Параметры состояния/настроек/таблиц: B 611
Таблица 168. массив Baudrate(), ключевые слова и настройки по умолчанию 611
Таблица 169. Массив Beacon(), ключевые слова и настройки по умолчанию 612
Таблица 170. Параметры состояния/настроек/таблиц: C 612
Таблица 171. Параметры состояния/настроек/таблиц: D 615
Таблица 172. Параметры состояния/настроек/таблиц: E 616
Таблица 173. Параметры состояния/настроек/таблиц: F 616

Таблица 174 Параметры состояния/настроек/таблиц: H 617
Таблица 175. Параметры состояния/настроек/таблиц: I 617
Таблица 176. Параметры состояния/настроек/таблиц: L 619
Таблица 177. Параметры состояния/настроек/таблиц: M 620
Таблица 178. Параметры состояния/настроек/таблиц: N 622
Таблица 179. Параметры состояния/настроек/таблиц: O 622
Таблица 180. Параметры состояния/настроек/таблиц: P 623
Таблица 181. Параметры состояния/настроек/таблиц: R 626
Таблица 182. Параметры состояния/настроек/таблиц: S 627
Таблица 183. Параметры состояния/настроек/таблиц: T 629
Таблица 184. Параметры состояния/настроек/таблиц: U 631
Таблица 185. Параметры состояния/настроек/таблиц: V 631
Таблица 186. Параметры состояния/настроек/таблиц: W 632
Таблица 187. Название выводов CS I/O 633
Таблица 188. Схема расположения выводов RS-232 системы CR1000 634
Таблица 189. Стандартный нульмодемный кабель или подключение контактов через адаптер 635
Таблица 190. Десятичные и шестнадцатеричные коды и символы, используемые приборами CR1000 637
Таблица 191. Описание битов в формате данных FP2 641
Таблица 192. Биты указателей разрядов десятичного числа в формате FP2 641
Таблица 193. Порядок следования байтов в приборах компании Campbell Scientific 643
Таблица 194. Регистраторы данных 645
Таблица 195. Модули аналогового ввода 646
Таблица 196. Модули импульсного ввода 646
Таблица 197. Список модулей последовательного ввода/вывода 646
Таблица 198. Модули ввода информации струнных датчиков 647
Таблица 199. Модули резистивного моста TIM 647
Таблица 200. Модули делителей напряжения 647
Таблица 201. Модули токовых шунтов 647
Таблица 202. Ограничители переходного напряжения 648
Таблица 203. Крышки клеммных колодок 648
Таблица 204 Модули цифрового ввода/вывода 648
Таблица 205. Модули непрерывного аналогового вывода (CAO) 649
Таблица 206. Управляющие цепи реле — продукты 649
Таблица 207. Модули возбуждения током 649
Таблица 208. Типы проводных датчиков 650
Таблица 209. Модули беспроводных датчиков 650
Таблица 210. Типы датчиков, доступные для подключения к CWS900 651
Таблица 211. Регистратор данных / доступность и совместимость дисплея с клавиатурой1 651
Таблица 212. Кабельные коммуникационные устройства с одиночным подключением 652
Таблица 213. Кабельные сетевые устройства 652
Таблица 214. Каналы TCP/IP 652
Таблица 215. Телефонные модемы 652
Таблица 216. Радио для частных сетей 653
Таблица 217. Спутниковые приемопередатчики 653
Таблица 218. Накопители данных большой емкости 653
Таблица 219. Модуль хранения данных на базе карты CF 653
Таблица 220. ПО начального уровня 654
Таблица 221. ПО поддержки регистратора данных 655
Таблица 222. Комплекс LoggerNet 1,2 655
Таблица 223. Программные средства 656
Таблица 224. Пакеты для разработки ПО 656
Таблица 225. Комбинации батарея / регулятор 657
Таблица 226. Батареи 658
Таблица 227. Регуляторы 658
Таблица 228. Первичные источники питания 658
Таблица 229 Комплекты источников питания 24 В пост. тока 659
Таблица 230. Корпуса — продукты 659
Таблица 231. Корпуса с предварительной прокладкой кабелей 659

Таблица 232. Штативы, башни и крепления 659
Таблица 233. Защита от влаги — продукты 660

Список примеров программ CRBasic

CRBasic: пример 1. Простой файл Default.cr1 для управления клеммой SW12 116
CRBasic: пример 2. Добавление комментариев 126
CRBasic: пример 3. Объявление типа данных 133
CRBasic: пример 4. Использование индексов размера массива переменных 134
CRBasic: пример 5. Объявление и использование флага 135
CRBasic: пример 6. Использование массива переменных в расчетах 136
CRBasic: пример 7. Инициализация переменных 137
CRBasic: пример 8. Использование объявления констант 138
CRBasic: пример 9. Загрузка двоичных данных в переменную 139
CRBasic: пример 10. Объявление и использование таблицы данных 142
CRBasic: пример 11. Использование переменной Disable 146
CRBasic: пример 12. Использование «включаемого» файла 149
CRBasic: пример 13. «Включаемый» файл для управления клеммой SW12 150
CRBasic: пример 14. Синтаксис команд BeginProg /Scan() /NextScan /EndProg 155
CRBasic: пример 15. Синтаксис команды снятия показаний 159
CRBasic: пример 16. Использование команды Move() для экономии кода 162
CRBasic: пример 17. Использование массивов переменных для экономии кода 162
CRBasic: пример 18. Преобразование переменной FLOAT/LONG в логическое значение 162
CRBasic: пример 19. Оценка целых чисел 163
CRBasic: пример 20. Преобразование констант в LONG или FLOAT 163
CRBasic: пример 21. Конкатенация строк и переменных 166
CRBasic: пример 22. Синтаксис команд BeginProg /Scan() /NextScan /EndProg 169
CRBasic: пример 23. Условный вывод 170
CRBasic: пример 24. Тестирование насоса для грунтовых вод 172
CRBasic: пример 25. Прочие возможности программы 174
CRBasic: пример 26. Масштабирование массива 177
CRBasic: пример 27. Сигнатуры программ 178
CRBasic: пример 28. Использование многократного сканирования 179
CRBasic: пример 29. Условный код 181
CRBasic: пример 30. Пользовательские меню 185
CRBasic: пример 31. Загрузка больших наборов данных 188
CRBasic: пример 32. Использование TrigVar для триггера сохранения данных 196
CRBasic: пример 33. Два интервала выходных данных в одной таблице данных 197
CRBasic: пример 34. Программирование с помощью Bool8 и оператора битового сдвига 200
CRBasic: пример 35. NSEC — массив времени с одним элементом 203
CRBasic: пример 36. NSEC — массив времени с двумя элементами 203
CRBasic: пример 37. NSEC— массивы времени с семью и девятью элементами 204
CRBasic: пример 38. NSEC — преобразование метки времени в универсальное время 205
CRBasic: пример 39. Использование команды TableFile() с параметром Option 64 с картой CF 207
CRBasic: пример 40. FieldCal() — нулевое значение 215
CRBasic: пример 41. FieldCal() — смещение 217
CRBasic: пример 42. FieldCal() — двухточечная калибровка наклона и смещения 219
CRBasic: пример 43. FieldCal() — множитель 221
CRBasic: пример 44. FieldCalStrain() — калибровка 225

CRBasic: пример 45. Считывание сигнала датчика с возбуждением и паузой....228
CRBasic: пример 46. Измерение VoltSE() с частотой 1 Гц....230
CRBasic: пример 47. Измерение VoltSE() с частотой 100 Гц....231
CRBasic: пример 48. Измерение VoltSE() с частотой 200 Гц....231
CRBasic: пример 49. Измерение VoltSE() с частотой 2000 Гц....233
CRBasic: пример 50. PT1GG в четырехжильном полумостовом подключении....240
CRBasic: пример 51. PT1GG в трехжильном полумостовом подключении....242
CRBasic: пример 52. PT1GG в четырехжильном подключении с полным мостом....244
CRBasic: пример 53. Получение строки RS-232....255
CRBasic: пример 54. Измерение показаний датчиков / Отправка данных RS-232....260
CRBasic: пример 55. Использование SDI12Sensor() для тестирования команды Cv....276
CRBasic: пример 56. Использование альтернативной параллельной команды (aC)....277
CRBasic: пример 57. Использование расширенной команды SDI-12....279
CRBasic: пример 58. Настройка датчика SDI-12....280
CRBasic: пример 59. Конкатенация чисел и строк....284
CRBasic: пример 60. Форматирование строк....287
CRBasic: пример 61. Подпрограмма с глобальными и локальными переменными....289
CRBasic: пример 62. Пользовательская веб-страница HTML....293
CRBasic: пример 63. Присвоение временных меток с системным временем....304
CRBasic: пример 64. Измерение времени успокоения....320
CRBasic: пример 65. Измерение и обработка четырехжильного подключения с полным мостом....341
CRBasic: пример 66. Реализация DNP3....410
CRBasic: пример 67. Связывание длинных переменных Modbus....416
CRBasic: пример 68. Пользовательская веб-страница HTML....421
CRBasic: пример 69. Использование значений NAN для фильтрации данных....485
CRBasic: пример 70. Использование операторов побитового сдвига....565

1. Введение

1.1. Добро пожаловать!

В ледяном холоде Антарктики, в раскаленном воздухе Долины смерти, на соленом ветру Тихого океана, в космической невесомости или в суровом климате вашего офиса — в самых разных точках планеты трудятся регистраторы данных Campbell Scientific. Наши заказчики работают с огромным ассортиментом приложений, от самых простых до невообразимо сложных. Регистратор данных CR1000 готов справиться с любой задачей, поставленной заказчиком. Цель данного руководства — ознакомить вас с возможностями этого регистратора для наиболее эффективного использования в работе.

Чтобы быстро ознакомиться с возможностями CR1000 по измерению аналоговых и цифровых сигналов, прочитайте разделы *Краткое руководство (стр. 41)* и *Общие сведения по системе (стр. 61). Это займет всего несколько минут.* Если вам необходима более детальная информация, к вашим услугам дальнейшие разделы руководства и другие материалы от Campbell Scientific. Если же в ваши задачи входит программирование на CRBasic, вам понадобится изучить большой объем документации по программе *CRBasic Editor*. Кроме того, Campbell Scientific предлагает дополнительные обучающие курсы по работе с CR1000.

Материалы в данном руководстве подобраны по принципу «от простого к сложному». Для решения рабочих задач зачастую достаточно ознакомиться с разделами *Краткое руководство (стр. 41)* и *Общие сведения по системе (стр. 61)*. *Краткое руководство (стр. 41)* содержит общее описание возможностей CR1000 по сбору данных, а также предлагает обучающий пример по считыванию сигнала с датчика. Раздел *Общие сведения по системе (стр. 61)* содержит общие сведения об основных функциях CR1000, более подробно описанных в последующих разделах и приложениях.

Содержание поможет вам изучить общую структуру данного руководства. При навигации по документу пользуйтесь алфавитным указателем и функцией поиска в электронной PDF версии.

Более подробную техническую информацию вы можете найти в других публикациях Campbell Scientific, свободно доступных для скачивания по ссылке *www.campbellsci.com*. Если какая-либо функция регистратора данных CR1000 требует подключения периферийного оборудования, рекомендуется также изучить руководство по этому оборудованию.

Если вы испытываете сложности с поиском нужной информации, желаете приобрести продукты Campbell Scientific или просто хотели бы задать вопрос одному из наших специалистов по продуктам, звоните на нашу горячую линию +1 (435) 227-9100 или пишите на адрес электронной почты *support@campbellsci.com*. Ранние модели регистраторов данных Campbell Scientific приветствовали пользователя дружелюбным сообщением ДОБРО ПОЖАЛОВАТЬ. Интерфейс современных регистраторов данных, таких как CR1000, изменился и стал несравнимо сложнее, но Campbell Scientific по-прежнему приветствует своих пользователей — ДОБРО ПОЖАЛОВАТЬ!

1.2. Условные обозначения

В данном руководстве по CR1000 используются условные обозначения. Обратите внимание, что в печатной версии руководства все цветные шрифты отображаются черным цветом.

- Подчеркивание — подчеркнутый текст содержит непроверенную информацию. Обычно встречается в черновых и предварительных версиях документов.
- Заглавные буквы — с заглавной буквы пишутся новые предложения, собственные имена и модели оборудования Campbell Scientific.

- **Жирный шрифт** — жирным шрифтом в тексте выделяются команды CRBasic, команды ввода, сообщения вывода, команды пользовательского интерфейса, текст на этикетках продуктов и названия таблиц данных.
- ***Номера страниц*** — в PDF-версии руководства номера страниц служат гиперссылками, ведущими на соответствующую страницу.
- *Курсив* — курсивом выделяются словарные статьи, названия публикаций, программ, разделов, таблиц, рисунков и примеров.
- ***Курсив и жирный шрифт*** — так в тексте обозначаются параметры и аргументы команд CRBasic.
- Синий цвет — синим отмечены команды CRBasic, вынесенные на отдельную строку.
- *Бирюзовый цвет, курсив* — комментарии к программам CRBasic.
- Команды CRBasic, команды ввода и сообщения вывода, вынесенные на отдельную строку или приведенные в тексте примеров программ, выделены шрифтом:
  Lucida Sans Typewriter

1.3. Копирование кода CRBasic

В данном руководстве содержатся десятки примеров кода CRBasic. Мы стараемся сделать использование этих примеров максимально простым и удобным. При копировании кода из примеров в *CRBasic Editor* помните:

Если пример кода напечатан на нескольких страницах, выделяйте и копируйте по одной странице. Иначе вместе с текстом примера вы захватите нумерацию страниц, заголовки и скрытые символы.

1.4. Описание текущей версии

Предварительная версия от 26 марта 2015 г. для операционной системы версии 28:

Рецензентам

По возможности просим вас дождаться выхода следующей предварительной версии данного руководства (ориентировочно в июне 2015 г.) и писать рецензии на ее материале.

Читателям

Это предварительная версия руководства, в которой могут быть не полностью отражены особенности операционной системы версии 28. Если вы считаете, что в данной версии отсутствует важная информация, свяжитесь с техническими специалистами Campbell Scientific.

Основные изменения по сравнению с версией 5/13: добавлен раздел *Меры предосторожности* *(стр. 7)*, в приложение *Состояние, настройки и информация о таблицах данных* *(стр. 603)* внесены практически все существенные изменения параметров состояния, настроек и информации о таблицах данных, добавленные в ОС версии 28.

Обновления остальных разделов, от раздела *Установка* *(стр. 99)* до приложения *Перечни продуктов для поддержки* *(стр. 645)*, находятся в процессе подготовки. Помимо прочего, ожидаются обновления/дополнения по следующим темам:

Аналоговые измерения
Массивы
CDM
Таблица констант
Типы данных
DNP3 (полная переработка материала)
Команда Function()
Дисплей с клавиатурой
Modbus

Команда NewFile()
Управление ОС
Среднее значение за период
Погрешность переменных
Программирование
Команда Route()
Безопасность
Пропущенные записи
Подпрограммы
Клеммы SW12 и 12V
Устройство определения последовательности
Режим терминала
Часы
Диагностика
Сброс сторожевого таймера

2. Предупреждения

- ВНИМАНИЕ, ОПАСНОСТЬ! Риск пожара, взрыва и сильного ожога. Неправильное использование или установка внутренней литиевой батареи может привести к серьезным травмам. Запрещается перезаряжать, разбирать, нагревать до температуры свыше 100 °C, припаивать к ячейке, сжигать или погружать содержимое в воду. Утилизация отработанных литиевых батарей должна выполняться в соответствии со всеми местными требованиями.
- ВНИМАНИЕ!

  - Избегайте перенапряжения
  - Не допускайте контакта с водой
  - Обеспечьте защиту от *электростатических разрядов* *(стр. 105)*

- ВНИМАНИЕ! Долгий простой ускоряет деградацию встроенного аккумулятора, питающего регистратор данных. В режиме хранения встроенный аккумулятор выходит из строя через 3 года или ранее. В режиме регулярного использования встроенный аккумулятор может прослужить до 10 лет и более. Подробную информацию см. в разделе *Внутренняя батарея — подробная информация* *(стр. 94)***.**
- ВАЖНАЯ ИНФОРМАЦИЯ! Регулярно выполняйте калибровку оборудования в соответствии с выполняемыми работами. Campbell Scientific рекомендует обновлять заводскую калибровку CR1000 каждые три года.

3. Перед началом работы

- Сверьтесь с документом **Комплект поставки** по ссылке http://www.campbellsci.com/CR1000 и убедитесь, что все детали комплекта в наличии. Помимо прочего, в комплект поставки входят:
  - Отвертка для подсоединения проводов к клеммам
  - Термопара типа T для использования в обучающем примере в разделе *Краткое руководство* *(стр. 41)*
  - Программа по измерению напряжения источника питания и температуры коммутационной панели, предустановленная на регистратор данных CR1000
  - Кабель для последовательной связи CR1000 с ПК
  - DVD диск, содержащий руководства по продуктам и следующее базовое ПО:
    - *Short Cut*
    - *PC200W*
    - *DevConfig*
- Сразу после поставки CR1000 убедитесь, что упаковка и содержимое не пострадали. Если на упаковке или содержимом присутствуют следы повреждений, обратитесь с жалобой в транспортную компанию, ответственную за доставку
- Осмотрите содержимое упаковки. Тщательно проверьте упаковку на наличие деталей комплекта, не замеченных ранее. Сверьте серийные номера продуктов и деталей, а также описания продуктов с накладной. Серийные номера нанесены на каждый продукт. Если устройство предусматривает проводное подключение, серийный номер часто расположен на соединительном кабеле, со стороны устройства. Серийные номера по каталогу Campbell Scientific могут отличаться от серийных номеров, которыми маркированы датчики согласно каталогу производителя. Убедитесь, что длина всех кабелей соответствует заявленной. В случае каких-либо расхождений с официальным комплектом поставки немедленно обратитесь к представителю Campbell Scientific.
- Проверьте версию ОС, установленной в CR1000. При необходимости обновите ее согласно инструкциям раздела *Обновление операционной системы (ОС)* *(стр. 117)*.

4. Краткое руководство по системе

Подробнее:
- *Краткое руководство (стр. 41)*
- *Технические характеристики (стр. 97)*
- *Установка (стр. 99)*
- *Эксплуатация (стр. 303)*

Данный раздел содержит общее описание возможностей CR1000 по сбору данных, а также предлагает обучающий пример по считыванию сигнала с датчика.

4.1. Системы сбора данных — краткое руководство

Связанные темы:
- *Системы сбора данных — краткое руководство (стр. 41)*
- *Система сбора данных — общие сведения (стр. 62)*

Сбор данных с помощью регистратора данных Campbell Scientific — это четко регламентированная процедура, предусматривающая использование электронных датчиков, регистратора данных CR1000, устройства связи и *ПО поддержки регистратора данных (см. стр. 512).*

Регистратор данных CR1000 — лишь один из компонентов системы сбора данных. Чтобы повысить качество данных, используйте сертифицированные датчики и придерживайтесь проверенных процедур сбора данных. Сбой в любом из компонентов системы может привести к потере или искажению собираемых данных. Типовая система сбора данных показана на рис. *Компоненты системы сбора данных (стр. 42)*. Типовая система содержит следующие компоненты:

- *Датчики (стр. 42)* — электронные средства измерения физических величин, вырабатывающие сигнал измерительной информации.
- *Регистратор данных (стр. 43)* — регистратор данных CR1000 считывает электрические сигналы датчиков или цифровые данные. CR1000 конвертирует собранные данные в технические единицы измерения, выполняет вычисления и сводит данные к статистическим значениям. Данные сохраняются в памяти устройства, после чего переносятся на персональный компьютер (на внешнем накопителе или по каналу связи).
- *Сбор и обмен данными (стр. 45)* — копирование данных на ПК или на другие устройства производится при помощи ПО поддержки регистратора данных. При копировании данные не удаляются из памяти регистратора. Большая часть из вариантов обмена данными является двунаправленной связью, поэтому программы и настройки можно отправлять в CR1000.
- *ПО поддержки регистратора данных (стр. 46)* — программное обеспечение для получения и отправки программ и настроек. Программное обеспечение управляет каналом связи и предусматривает опции для отображения данных.
- *Программируемое логическое управление (стр. 74)* — некоторым системам сбора данных требуется управление со стороны внешних устройств для снятия показаний или управления устройством на основе показаний. CR1000 — эксперт в программируемом логическом управлении. К сожалению, эти возможности освещены достаточно скупо в данном руководстве. Для получения дополнительной информации обратитесь к *CRBasic Editor Help (стр. 125)* или научным специалистам Campbell Scientific.
- *Внешнее оборудование для измерения и контроля (стр. 85)* — некоторым системам требуются возможности, превышающие стандартные входы и выходы, предусмотренные в регистраторе CR1000. Дополнительные входные и выходные расширительные модули могут удовлетворить большинству этих требований.

Рисунок 1. Компоненты системы сбора данных

4.2. Датчики — краткое руководство

Связанные темы:

- *Датчики — краткое руководство* *(стр. 42)*
- *Измерения — общие сведения* *(стр. 62)*
- *Измерения — подробная информация* *(стр. 303)*
- *Датчики — списки* *(стр. 649)*

Датчики преобразуют явления в измеряемую электрическую форму, модулируя сигналы по напряжению, току, сопротивлению, состоянию или импульсные выходные сигналы. Соответствующие целевому применению датчики осуществляют это *точно и безошибочно* *(стр. 533)***.** Интеллектуальные датчики оснащены внутренними компонентами для измерения и обработки и просто выводят цифровое значение в двоичном, шестнадцатеричном формате или формате ASCII. Регистратор данных CR1000, иногда в сочетании с различными периферийными устройствами, может измерить или считать практически все типы выходных сигналов датчиков.

Поддерживаемые типы датчиков:

- Аналоговые
  - Напряжение
  - Ток
  - Термопары
  - Мосты сопротивления
- Импульсные
  - Высокая частота
  - Переключение-замыкание
  - Низкоуровневый переменный ток, Гц
- Среднее значение за период
- Вибрационные струнные
- Интеллектуальные датчики
  - SDI-12
  - RS-232

- Modbus
- DNP3
- RS-485

В приложении *Датчики — списки (стр. 649)* приводится список узкоспециальных датчиков производства Campbell Scientific. Для получения информации по работе с различными измерительными датчиками можно обратиться к библиотеке руководств по датчикам и замечаний по их эксплуатации на сайте *www.campbellsci.com*. Вышеприведенный список поддерживаемых датчиков не является исчерпывающим. При работе с датчиками незнакомого типа обратитесь за помощью к техническому специалисту Campbell Scientific.

4.3. Регистратор данных — краткое руководство

Связанные темы:
- *Регистратор данных — краткое руководство (стр. 43)*
- *Регистратор данных — общие сведения (стр. 75)*
- *Регистратор данных — список (стр. 645)*

CR1000 может получать измерения практически всех датчиков с электрическим выходом. CR1000 измеряет электрические сигналы и конвертирует результат измерения в технические единицы измерения, выполняет вычисления и сводит данные к статистическим значениям. Большинству случаев применения не требуется хранение каждого измерения по отдельности, тогда как требуется сочетание с другими измерениями в виде статических или расчетных сводных таблиц. CR1000 хранит данные в памяти в ожидании передачи в ПК при помощи внешних запоминающих устройств или средств связи.

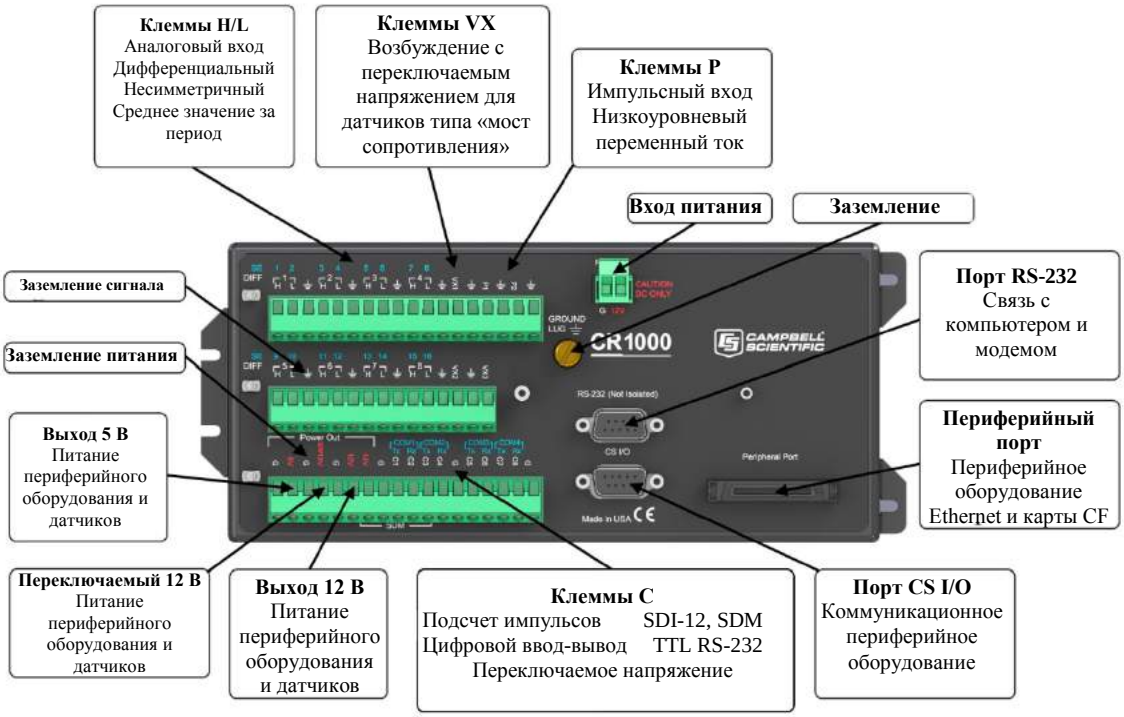
Электронные средства регистратора CR1000 находятся в защищенном герметичном корпусе из нержавеющей стали. Такая конструкция обеспечивает экономичность, компактность и надежность регистратора CR1000.

4.3.1.1. Коммутационная панель — краткое руководство

Связанные темы
- *Коммутационная панель — краткое руководство (стр. 43)*
- *Коммутационная панель — общие сведения (стр. 76)*
- *Внешнее оборудование для измерения и контроля (стр. 366)*

Как показано на рисунке *Коммутационная панель (стр. 44)*, коммутационная панель CR1000 предусматривает клеммы для подключения датчиков, электропитания и коммуникационных устройств. Большинство соединителей коммутационной панели оснащается защитой от перенапряжения.

Рисунок 2. Коммутационная панель



4.4. Источники питания — краткое руководство

Связанные темы:
- Источники питания — технические характеристики
- *Источники питания — краткое руководство* *(стр. 44)*
- *Источники питания — общие сведения* *(стр. 85)*
- *Источники питания — общие сведения* *(стр. 100)*
- *Источники питания — продукты* *(стр. 657)*
- *Источники питания* *(стр. 101)*
- *Диагностика — источники питания* *(стр. 494)*

Для работы CR1000 требуется источник питания. Убедитесь, что все компоненты источника питания соответствуют техническим характеристикам устройства, к которому они подключаются. При подключении питания необходимо выключить подачу питания, выполнить подсоединение и только после этого включить подачу питания.

Для работы CR1000 требуется питание напряжением от 9,6 до 16 В пост. тока, подаваемое на клеммы **POWER IN** зеленого соединителя на лицевой части коммутационной панели.

Внешнее питание подключается через зеленый соединитель **POWER IN** на лицевой панели CR1000. Положительный вывод питания подключается к клемме **12V**. Отрицательный вывод — к клемме **G**. Соединение имеет внутреннюю защиту от обратной полярности.

CR1000 внутренне защищен от случайного изменения полярности питания.

4.4.1. Внутренняя батарея — краткое руководство

Связанные темы:
- *Внутренняя батарея — краткое руководство (стр. 45)*
- *Внутренняя батарея — общие сведения (стр. 94)*

Предупреждение! Неправильное использование или установка внутренней литиевой батареи может привести к серьезным травмам, пожару, взрыву и тяжелым ожогам. Запрещается перезаряжать, разбирать, нагревать до температуры свыше 100 °C, припаивать к ячейке, сжигать или погружать содержимое в воду. Утилизация отработанных литиевых батарей должна выполняться в соответствии со всеми местными требованиями.

Литиевая батарея обеспечивает резервное питание для часов, программ и памяти регистратора CR1000.

4.5. Сбор и обмен данными — краткое руководство

Связанные темы:
- *Сбор и обмен данными — краткое руководство (стр. 45)*
- *Сбор и обмен данными — общие сведения (стр. 88)*
- *Сбор и обмен данными — подробная информация (стр. 391)*
- *Периферийное оборудование для сбора и обмена данными — списки (стр. 651)*

Если регистратор данных CR1000 находится в непосредственной близости от ПК, наилучшим решением является прямое последовательное соединение. В полевых условиях можно использовать запоминающее устройство с прямым последовательным соединением. Возможность удаленной связи (или сочетание вариантов) обеспечивает сбор данных с ПК на большом удалении и возможность раннего обнаружения проблем.

Технический специалист Campbell Scientific может помочь с выбором следующих вариантов связи:

- Стандартный
  - Последовательный RS-232
- Варианты
  - Ethernet
  - CompactFlash, запоминающее устройство
  - Мобильная телефонная связь
  - iOS, Android
  - Планшет
  - Многоточечные линии, оптоволокно
  - Радио-, спутниковая связь

Можно совмещать несколько вариантов связи.

4.6. ПО поддержки регистратора данных — краткое руководство

Подробнее:
- *ПО поддержки регистратора данных — краткое руководство (стр. 46)*
- *ПО поддержки регистратора данных — общие сведения (стр. 95)*
- *ПО поддержки регистратора данных — подробная информация (стр. 450)*
- *ПО поддержки регистратора данных — списки (стр. 654)*

ПО поддержки регистратора данных Campbell Scientific для компьютера с Windows или Linux ускоряет связь между компьютером и CR1000. На рынке имеется огромный массив программного обеспечения, но в данном разделе рассматриваются следующие приложения:

- *Short Cut* Генератор программ для Windows (SCWin)
- *PC200W* ПО начального уровня регистратора данных для Windows
- *LoggerLink* Мобильное ПО начального уровня регистратора данных для ОС iOS и Android

Программа CRBasic должна быть загружена в регистратор CR1000, чтобы данное приложение могло производить измерения, считывать показания датчиков и сохранять данные. *Short Cut* используется для написания простых программ CRBasic и не требует изучения языка программирования CRBasic. *Short Cut* представляет собой простой в использовании мастер, который поможет вам подготовить программу.

После того как программа CRBasic написана, она загружается в CR1000. Затем, по прошествии достаточного времени для измерений и сохранения данных, данные выгружаются в компьютер. Эти функции также поддерживаются в *PC200W* и *LoggerLink Mobile*.

Short Cut и *PC200W* можно бесплатно скачать на сайте *www.campbellsci.com/downloads (http://www.campbellsci.com/downloads)*.

Примечание. Подробную информацию о ПО Campbell Scientific можно найти на сайте *www.campbellsci.com http://www.campbellsci.com*. Для получения рекомендации по выбору ПО, подходящего для вашей сферы деятельности, обратитесь к техническому специалисту Campbell Scientific.

4.7. Учебное пособие: считывание сигнала с термопары

В данном руководстве представлены основные функции регистратора CR1000. В упражнении выделяются следующие аспекты:

- привязка датчика к регистратору данных;
- создание программы для регистратора CR1000 для измерения датчика;
- выполнение простых измерений;
- сохранение данных измерений;
- сбор данных из регистратора CR1000 при помощи ПК;
- просмотр архивных данных и данных в PMB, полученных от CR1000.

4.7.1. Необходимое оборудование

В данном упражнении используются следующие изделия. При отсутствии данных изделий можно использовать аналогичное. При возникновении вопросов по совместимым источникам питания или кабелям последовательной передачи данных просим обратиться к техническому специалисту Campbell Scientific.

- Регистратор данных CR1000;
- источник питания с выходным напряжением от 10 до 16 В пост. тока;
- термопара длиной 10–15 см из комплекта поставки регистратора CR1000;
- персональный компьютер (ПК) с последовательным 9-контактным портом RS-232 или с USB-портом и переходником USB-на-RS-232;
- кабель с 9-контактным гнездовым и 9-контактным штыревым соединителями RS-232 из комплекта поставки CR1000;
- программное обеспечение *PC200W* находится либо на диске DVD или флеш-накопителе Campbell Scientific, либо на сайте *www.campbellsci.com*.

Примечание. Если регистратор данных CR1000 подключается к ПК в ходе нормальной эксплуатации, используйте интерфейс Campbell Scientific SC32B, чтобы обеспечить оптическую изоляцию через порт **CS I/O**. В этом случае обеспечивается защита низкоуровневых измерений от помех заземления.

4.7.2. Подготовка оборудования

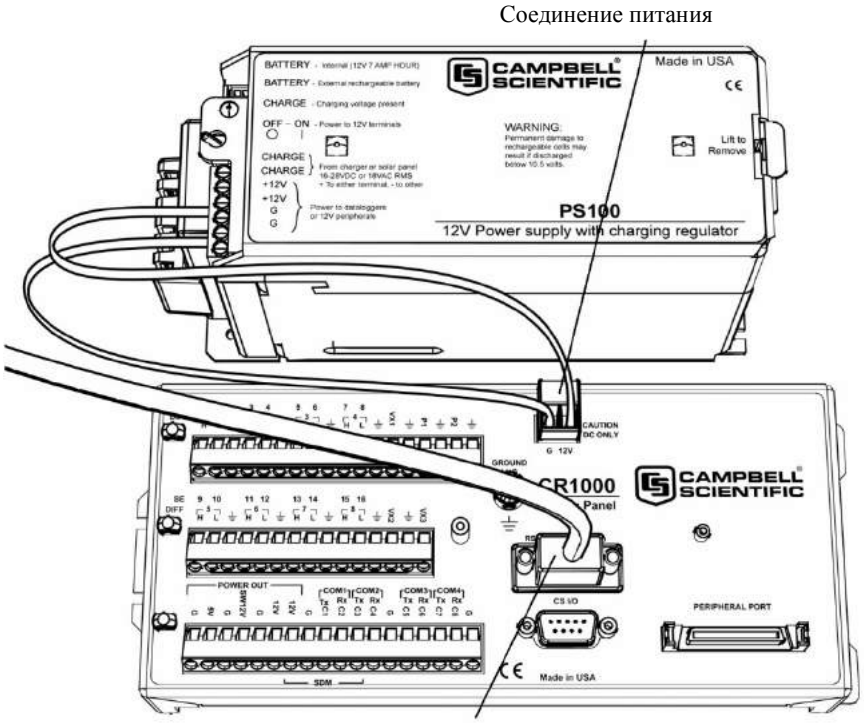
Примечание. Термопара подключается к CR1000 далее в этом упражнении.

4.7.2.1. Внешний источник питания

Руководствуясь рисунком *Соединения питания и последовательной передачи данных* *(стр. 48)*, действуйте следующим образом:

1. Отсоедините зеленый соединитель питания от коммутационной панели CR1000.
2. Отключите источник питания.
3. Подсоедините положительный вывод источника питания к клемме **12 V** зеленого соединителя питания. Подсоедините отрицательный вывод источника питания к клемме **G** зеленого соединителя.
4. Проверив полярность соединений источника питания, вставьте зеленый соединитель питания в соответствующее гнездо на коммутационной панели CR1000.
5. Подсоедините кабель последовательной передачи данных к порту **RS-232** на регистраторе CR1000 и к порту RS-232 на ПК.
6. Включите источник питания.

Рисунок 3. Соединения питания и последовательной передачи данных



4.7.3. Настройка программы PC200W

1. Установите программное обеспечение *PC200W* на ПК. Следуйте инструкциям на экране в процессе установки. Используйте каталоги по умолчанию.
2. Откройте *PC200W*. На экране ПК должно отображаться окно, как показано на рисунке *Главное окно PC200W* *(стр. 49)*. При первом запуске PC200W в новом окне автоматически откроется мастер настройки *EZSetup Wizard*. Он предназначен для настройки конфигурации программного обеспечения для связи с регистратором данных CR1000. В таблице *Пример выбора настроек в мастере EZSetup программы PC200W* *(стр. 49)* указывается, какую информацию вводить на каждом экране мастера. Нажмите **Next** (Далее) в нижней части окна, чтобы развернуть его.

Смотреть ролик! Обучающий видеоролик доступен по следующей ссылке: *www.youtube.com/playlist?list=PL9E364A63D4A3520A&feature=plcp*. Другие обучающие видеоролики доступны по ссылке *www.campbellsci.com/videos*.

После выхода из мастера отобразится главное окно *PC200W*. В данном окне имеется несколько вкладок. На вкладке **Clock/Program** (Часы/Программа) отображается информация, выбранная на текущий момент в CR1000 с функциями часов и программы. Также доступны вкладки **Monitor Data** (Мониторинг данных) и **Collect Data** (Сбор данных). С помощью пиктограмм в верхней части окна можно вызвать дополнительные функции.

Рисунок 4. Главное окно PC200W

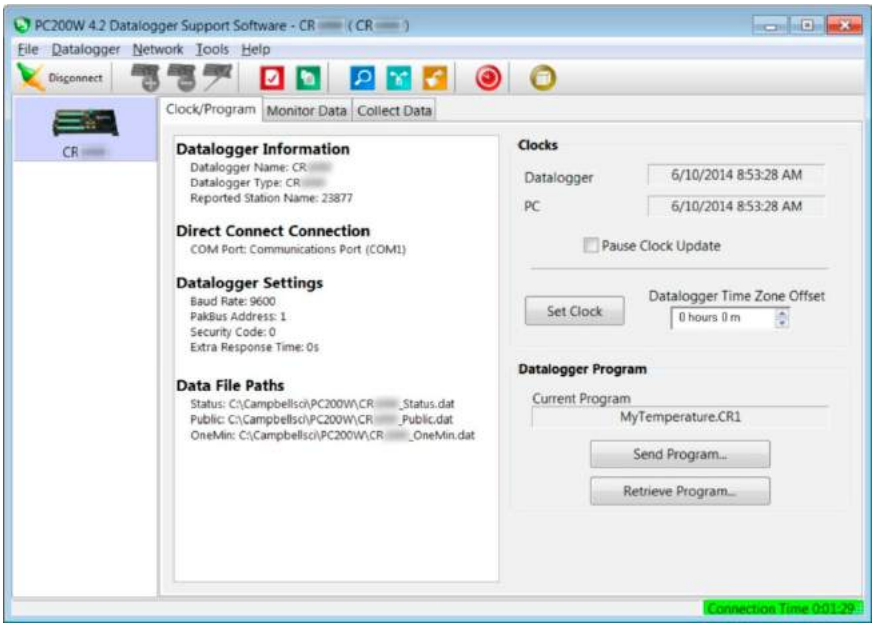


Таблица 1. Пример выбора настроек в мастере EZSetup программы PC200W	
Запустите мастер, чтобы получить информацию по записям в таблице.	
Название экрана	***Необходимая информация***
Введение	Предоставляет вводную информацию по *EZSetup Wizard* наряду с командами по переходам в рамках мастера.
Datalogger Type and Name (Тип и название регистратора данных)	Выберите CR1000 из ниспадающего списка. Примите название по умолчанию «CR1000».
COM Port Selection (Выбор COM-порта)	Выберите соответствующий COM-порт ПК для последовательной передачи данных. Обычно это COM1. Также можно использовать порты COM с другими номерами, особенно при использовании USB-кабеля. Оставить в поле **COM Port Communication Delay (Задержка связи COM-порта)** значение **00** секунд. **Примечание.** При использовании USB-кабелей для последовательной передачи данных номер COM-порта можно изменить, если кабель подключается к другому USB-порту. Таким образом, исключается возможность передачи данных между программным обеспечением и CR1000. В этом случае просто подключите кабель в исходный порт. Если сделать это невозможно, закройте и повторно откройте программное обеспечение *PC200W*, чтобы обновить доступные COM-порты. Нажмите на **Edit Datalogger Setup (Редактировать настройки регистратора данных)** и измените номер COM-порта на новый.
Datalogger Settings (Настройки регистратора данных)	Здесь содержатся настройки параметров связи между CR1000 и ПК. В целях данного упражнения примите настройки по умолчанию.
Datalogger Settings – Security (Настройки регистратора данных – безопасность)	В целях данного упражнения установите для параметра **Security Code (Код безопасности)** значение **0** и не заполняйте параметр **PakBus Encryption Key (Код шифрования PakBus)**.
Communication Setup Summary (Сводная информация по настройкам связи)	Предоставляет сводную информацию по настройкам в предыдущих экранах. В данном упражнении вносить изменения не требуется. Нажмите **Finish** (Готово) для выхода из мастера.

4.7.4. Написание программы CRBasic при помощи Short Cut

Назначение *Short Cut*:

- Создать программу для измерения напряжения источника питания CR1000, температуры коммутационной панели CR1000 и температуры окружающего воздуха с помощью термопары.
- Когда программа загружается в CR1000, она выполняет измерения с частотой один раз в секунду и сохраняет средние значения этих параметров с интервалом в 1 минуту.

Смотреть ролик. Обучающий видеоролик доступен по следующей ссылке: *www.youtube.com/playlist?list=PLCD0CAFEAD0390434&feature=plcp http://www.youtube.com/playlist?list=PLCD0CAFEAD0390434&feature=plcp.* Другие обучающие видеоролики доступны по ссылке *www.campbellsci.com/videos.*

4.7.4.1. Процедура: (шаги 1–5)

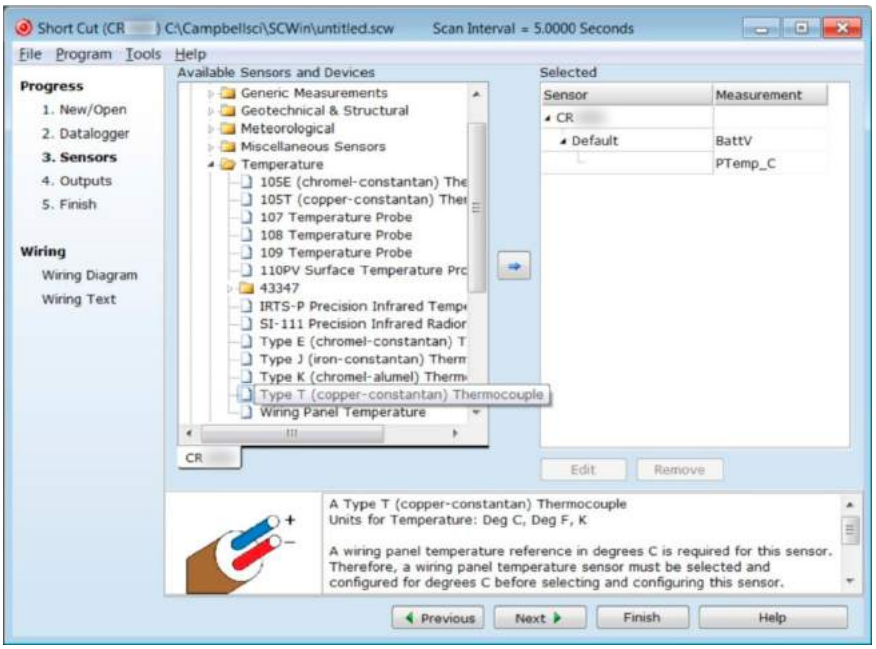
1. Нажмите на пиктограмму *Short Cut* в верхнем правом углу окна *PC200W*. На пиктограмме изображены часы.
2. Откроется окно *Short Cut*. Нажмите **New Program** (Новая программа).
3. В ниспадающем списке **Datalogger Model** (Модель регистратора данных) выберите **CR1000**.
4. В поле **Scan Interval** (Сканировать интервал) введите **1** и выберите **Seconds** (Секунды) в ниспадающем списке. Нажмите **Next** (Далее).

Примечание. При первом запуске программы *Short Cut* появится диалоговое окно выбора подавления шумов переменного тока. Выберите значение **60 Hz**, если оборудование используется в США или в странах, где используется напряжение переменного тока частотой 60 Гц. Выберите значение **50 Hz**, если оборудование используется в странах Европы или в странах, где используется напряжение переменного тока частотой 50 Гц.

В следующем диалоговом окне содержится список поддерживаемых датчиков. **Campbell Scientific, Inc. (US)** является, вероятно, наилучшим вариантом, если оборудование используется за пределами стран Европы.

5. В следующем окне отображаются **Available Sensors and Devices** (Доступные датчики и устройства). Разверните каталог **Sensors** (Датчики), нажав на символ ▷, чтобы отобразить несколько подкаталогов. Разверните каталог **Temperature** (Температура) для просмотра доступных датчиков. Учитывайте, что температура коммутационной панели (**PTemp_C** в столбце **Selected** (Выбрано)) выбирается по умолчанию.

Рисунок 5. Папка датчиков температуры в программе Short Cut



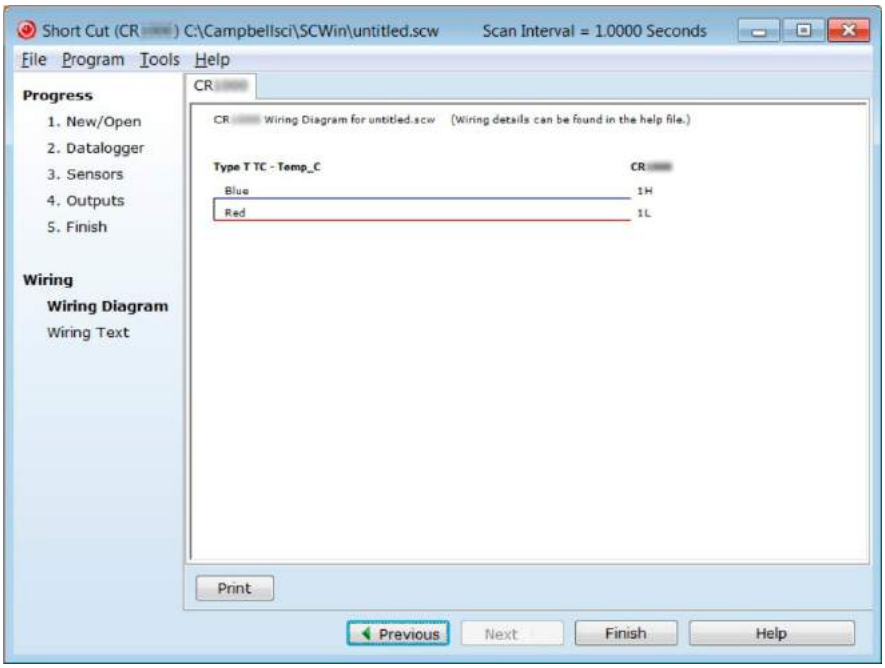
4.7.4.2. Процедура: (шаги 6–7)

6. Дважды нажмите на **Type T (copper-constantan) Thermocouple** (Термопара типа T (медь-константан)), чтобы добавить ее в столбец **Selected**. В диалоговом окне представлено несколько полей. Нажимая **OK**, вы принимаете значения по умолчанию, в том числе: **1** датчик и **PTemp_C** в качестве эталонного измерения температуры.

Примечание. **BattV** (напряжение батареи) и **PTempC** (температура коммутационной панели) являются измерениями по умолчанию. Во время работы состояние батареи и температура должны регистрироваться не реже одного раза в день, чтобы отслеживать состояние системы.

7. В левой части главного окна *Short Cut* нажмите **Wiring Diagram** (Схема соединений). Подсоедините термопару физического типа T к CR1000, как показано на схеме. Нажмите на **3. Sensors** (Датчики) в левой части окна, чтобы вернуться к экрану выбора датчиков.

Рисунок 6. Подключение термопары в программе Short Cut



4.7.4.3. Процедура: (шаг 8)

Историческая справка. Во времена космической гонки измерение термопар в полевых условиях представляло собой сложный и трудоемкий процесс, для которого требовалась термопара с тремя спаями, микровольтметр, термос с ледяной шугой и толстенный справочник. Один спай подключался к микровольтметру. Другой опускался в термос, в котором поддерживалась эталонная температура 0 °C. Третий вставлялся в место, в котором необходимо было измерить температуру. После стабилизации измерений микровольтметра показания последнего регистрировались вручную. Данное значение в последующем необходимо было найти в соответствующей таблице справочника, чтобы найти эквивалентную температуру.

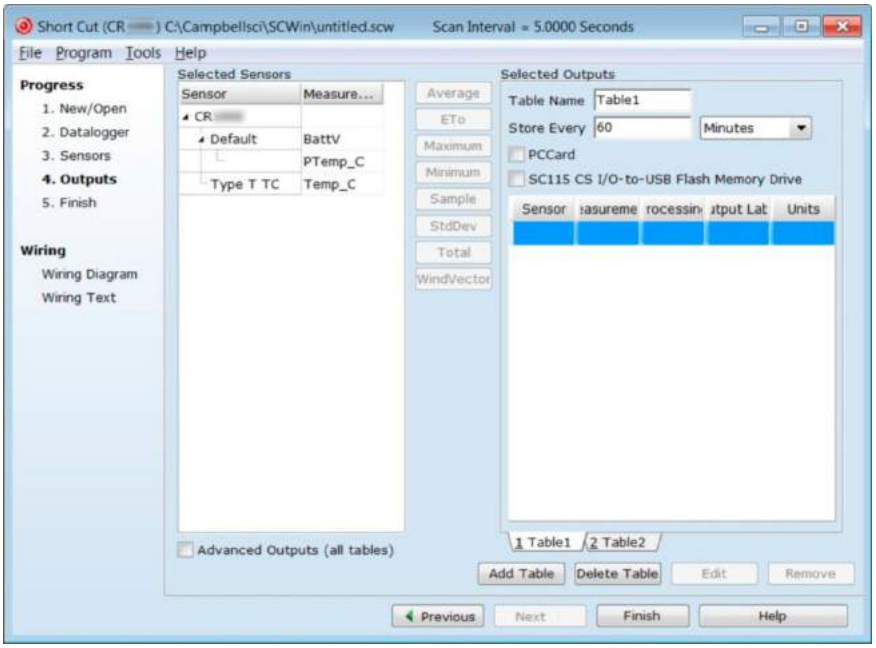
Затем на рынок пришли Эрик и Эван Кэмпбелл. Компания Campbell Scientific разработала первый регистратор данных CR7, чтобы регистрацию температуры можно было вести без термосов, справочников или трех спаев. И вот, идея!

В наши дни, термопара может состоять всего из двух проводов из различного металла, таких как медь и константан, соединенные на одном конце. Соединенный конец является измерительным соединением; соединение, создаваемое при подключении двух проводов к аналоговым входным клеммам CR1000, является эталонным спаем.

Если температура обоих спаев различается, в проводах возникает напряжение, прямо пропорциональное этой разнице температур. Для измерения термопары требуется только температура эталонного спая для расчета температуры измерительного спая на основе алгоритмов собственной разработки в операционной системе CR1000.

8. Нажмите **Next** (Далее), чтобы развернуть вкладку **Outputs** (Выходы), на которой слева отображается список **Selected Sensors** (Выбранные датчики), а справа — таблицы хранения данных в столбце **Selected Outputs** (Выбранные выходы).

Рисунок 7. Вкладка Outputs (Выходы) программы Short Cut



4.7.4.4. Процедура: (шаги 9–12)

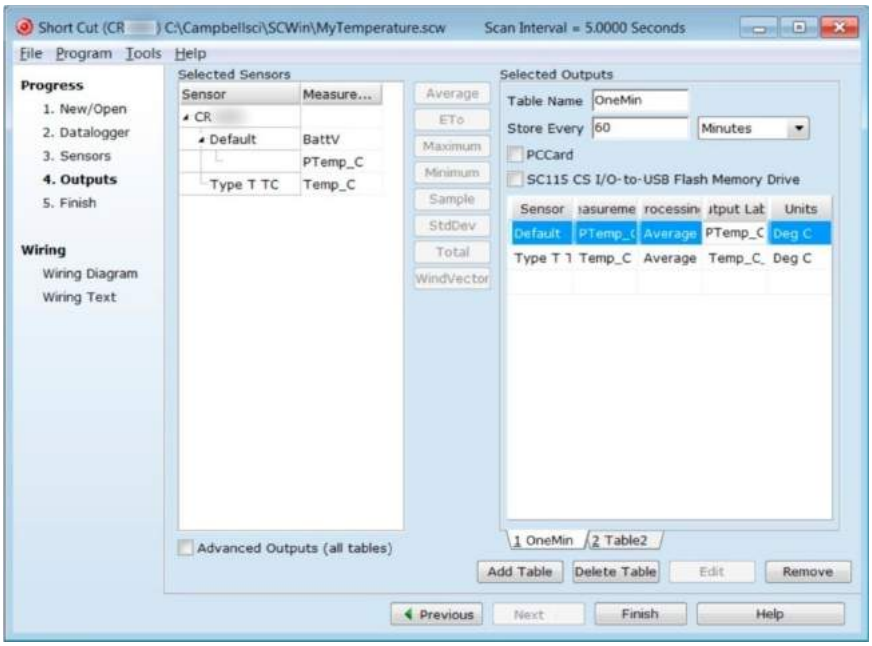
9. Изначально доступны две таблицы выходов (вкладки **1 Table1** и **2 Table2**). Обе таблицы имеют поле **Store Every** (Сохранять каждые) и ниспадающий список, в котором можно выбрать единицы времени. Они используются для определения временных интервалов при сохранении данных.

10. В целях данного упражнения нам понадобится только одна таблица, поэтому таблицу 2 можно удалить. Нажмите **2 Table2**, а затем — **Delete Table (Удалить таблицу).**

11. Измените имя оставшейся таблицы с **Table1** на **OneMin**, а затем измените интервал сохранения Store Every до **1 Minutes**.

12. Добавьте измерения в таблицу, выбрав значение **BattV** в поле **Selected Sensors** (Выбранные датчики), а затем нажмите **Average** (Среднее значение) в центральном столбце кнопок. Выполните эти же действия для **PTemp_C** и **Temp_C**.

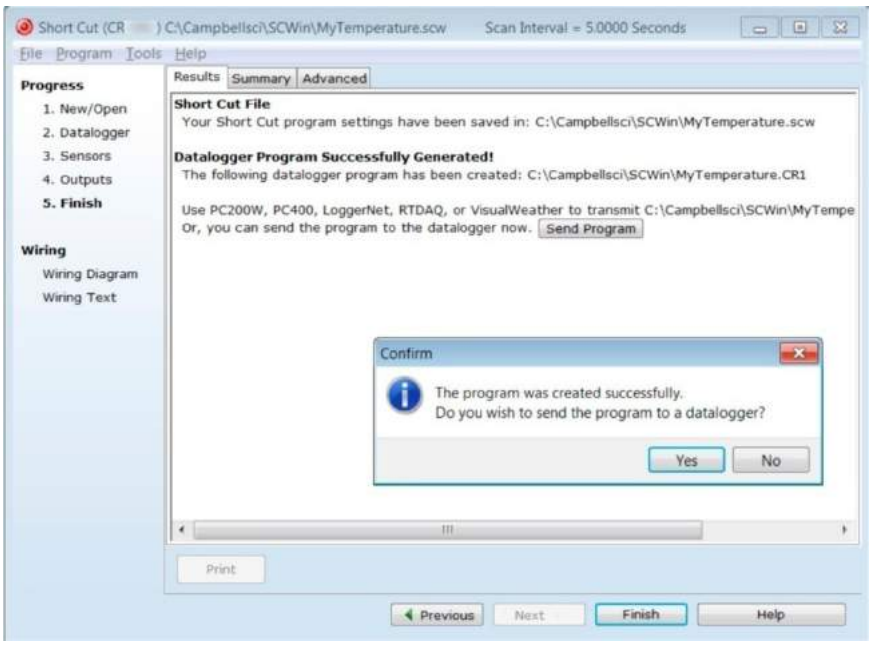
*Рисунок 8. Вкладка **Outputs** (Выходы) программы Short Cut*



4.7.4.5. Процедура: (шаги 13–14)

13. Нажмите **Finish** (Готово), чтобы скомпилировать программу. Присвойте программе имя **MyTemperature**. Появится экран сводной информации с отображением результатов компилятора. Все ошибки, выявленные во время компиляции, будут отображены.

Рисунок 9. Подтверждение компиляции в Short Cut



14. Закройте данное окно, нажав на X в правом верхнем углу.

4.7.5. Отправка программы и сбор данных

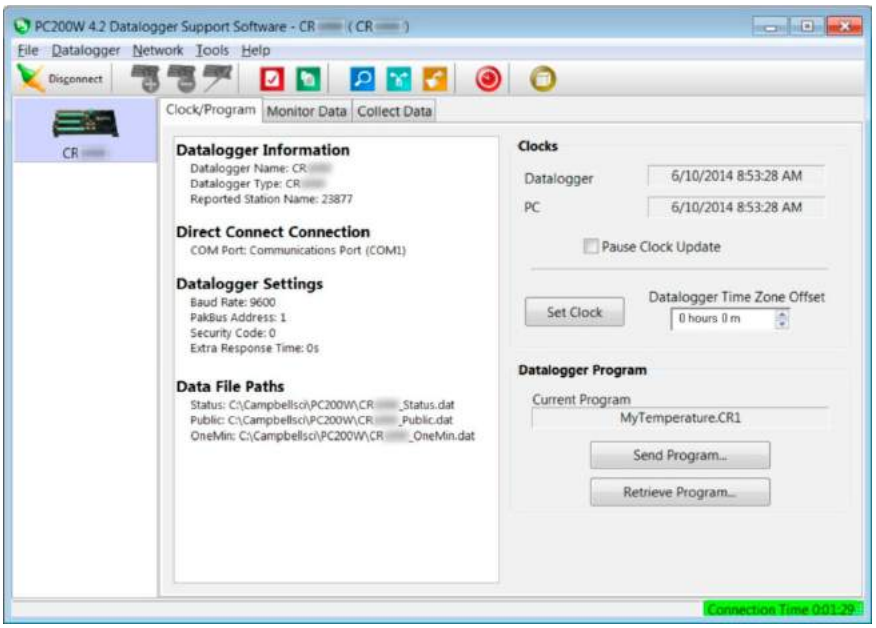
Назначение ПО поддержки регистратора данных PC200W:

- Отправка программы CRBasic, созданной в *Short Cut* в ходе выполнения предыдущей процедуры, в регистратор CR1000.
- Сбор данных из CR1000.
- Сохранение данных на ПК.

4.7.5.1. Процедура: (шаг 1)

1. На вкладке *PC200W* **Clock/Program** (Часы/Программа) нажмите кнопку **Connect** (Подсоединить), чтобы установить связь с CR1000. После установления соединения надпись на кнопке изменится на **Disconnect** (Отсоединить).

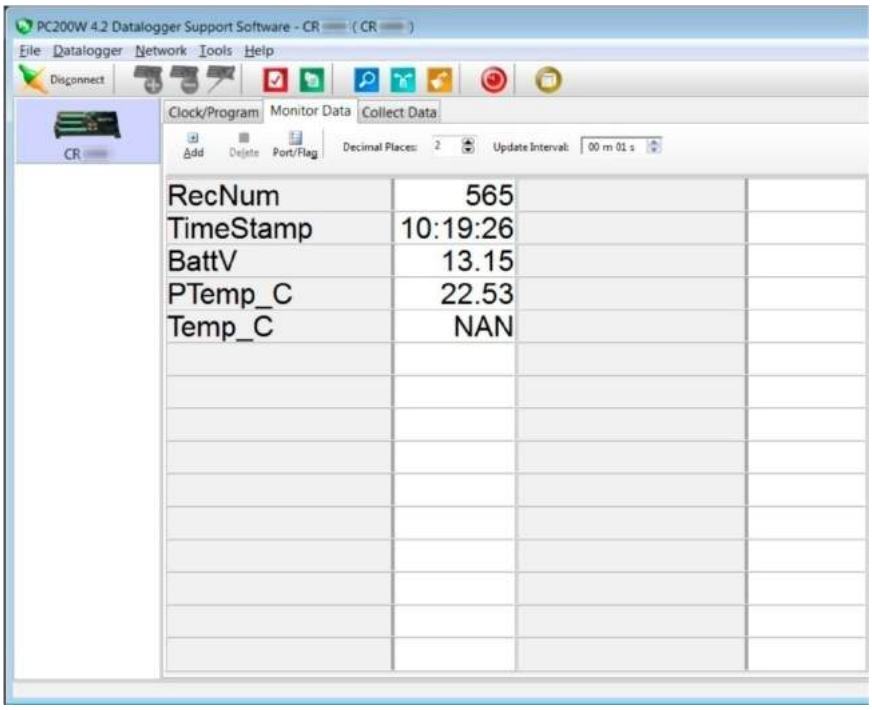
Рисунок 10. Главное окно PC200W



4.7.5.2. Процедура: (шаги 2–4)

2. Нажмите **Set Clock** (Установить часы) для синхронизации часов CR1000 с часами компьютера.
3. Нажмите **Send Program.... (Отправить программу).** Появится предупреждение о том, что данные на регистраторе данных будут удалены. Нажмите **Yes** (Да). Откроется диалоговое окно. Укажите путь к каталогу *C:\CampbellSci\SCWin*. Выберите файл **MyTemperature.cr1.** Нажмите **Open** (Открыть). При отправке программы в CR1000 появится строка состояния с последующим выводом предупреждения о том, что передача завершена успешно. Нажмите **OK**, чтобы закрыть подтверждение.
4. После отправки программы в CR1000 полезно проверить измерения, чтобы убедиться в их правильности. Выберите вкладку **Monitor Data** (Мониторинг данных). Теперь в окне отображаются данные из таблицы **Public** (Общие) регистратора CR1000.

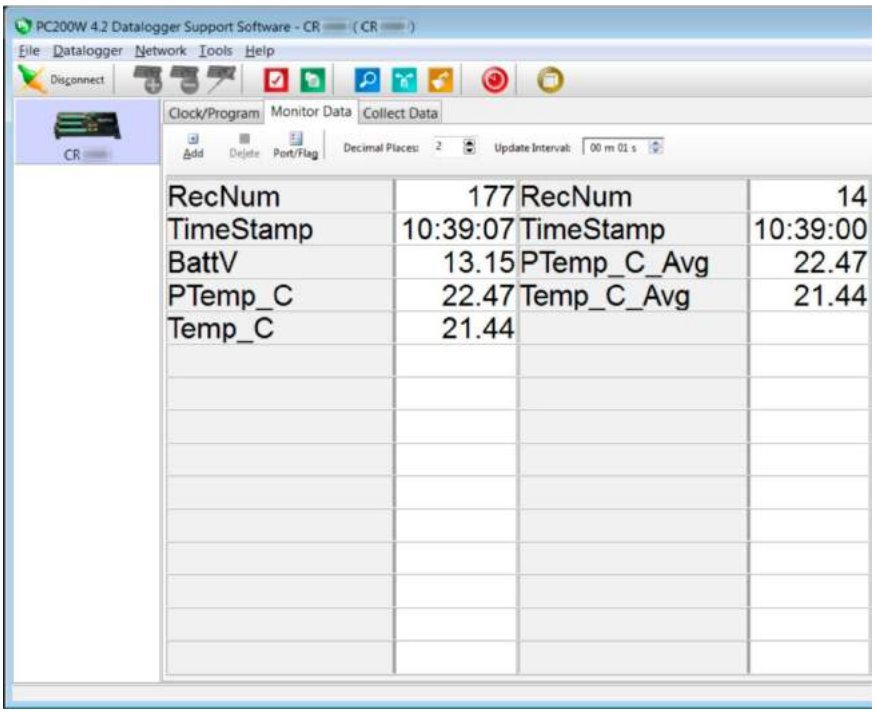
*Рисунок 11. Вкладка **Monitor Data** (Мониторинг данных) PC200W — таблица **Public***



4.7.5.3. Процедура: (шаг 5)

5. Чтобы просмотреть таблицу **OneMin**, выберите пустую в области отображения. Нажмите **Add** (Добавить). В окне **Add Selection** (Добавить выделение), поле **Tables** (Таблицы) нажмите **OneMin**, а затем нажмите **Paste** (Вставить). Отобразится таблица **OneMin**.

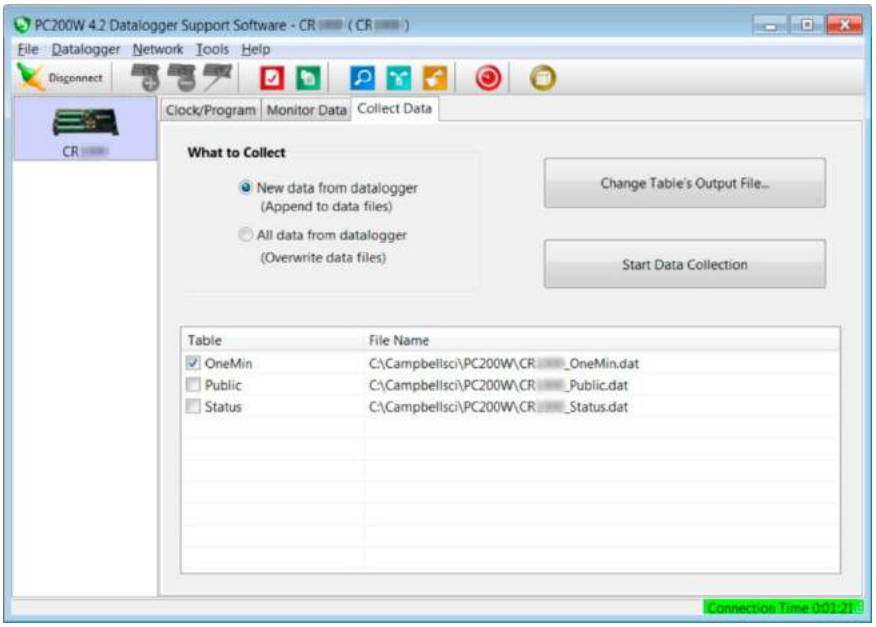
*Рисунок 12. Вкладка **Monitor Data** (Мониторинг данных) PC200W — таблицы Public и OneMin*



4.7.5.4. Процедура: (шаг 6)

6. Нажмите на вкладку **Collect Data** (Собрать данные) и выберите данные для сбора и укажите путь для хранения на ПК.

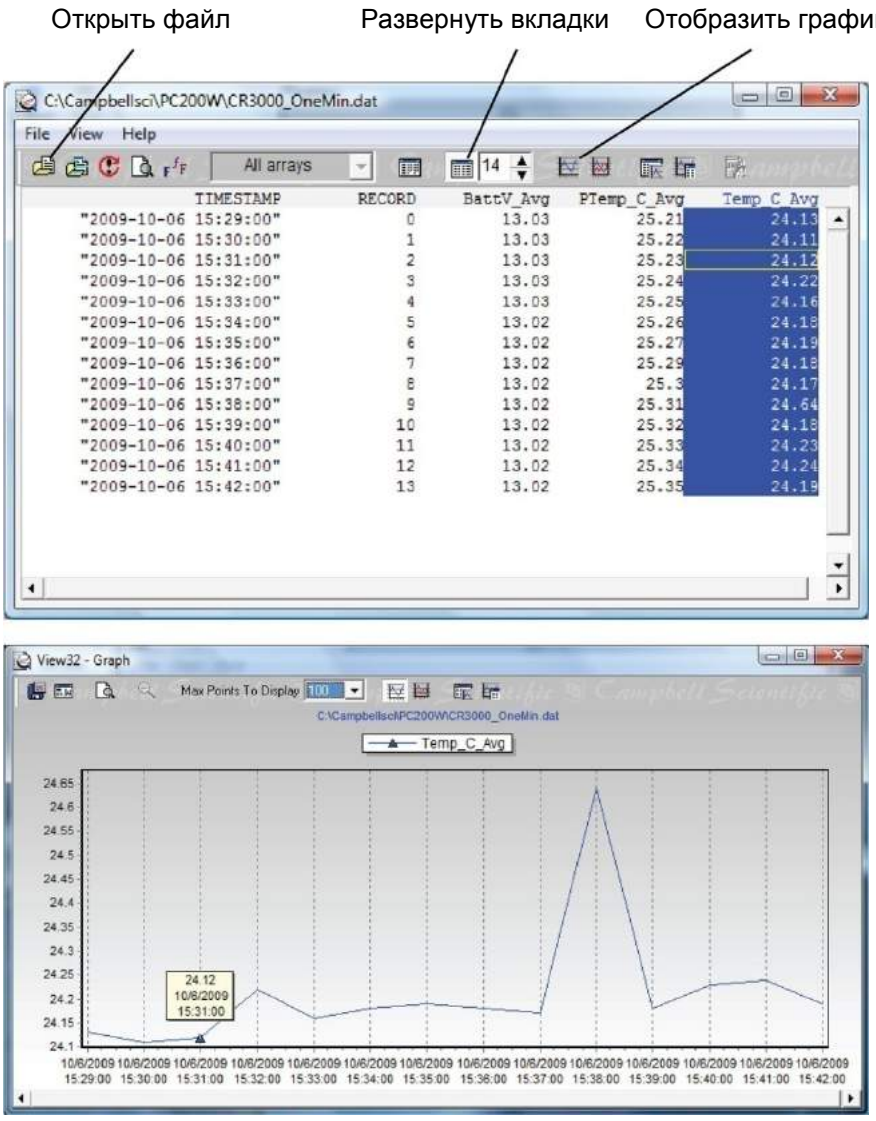
*Рисунок 13. Вкладка **Collect Data** (Собрать данные) PC200W*



4.7.5.5. Процедура: (шаги 7–10)

7. Установите флажок в клетке **OneMin**. В области **What to Collect** (Какие данные выбирать) выберите **New data from datalogger** (Новые данные из регистратора данных). Так выбираются данные для сбора.
8. Нажмите на таблицу в списке, чтобы выделить ее, а затем нажмите **Change Table's Output File...** (Изменить выходной файл таблицы...), чтобы изменить имя целевого файла.
9. Нажмите **Collect (Собрать).** Во время сбора данных отображается индикатор выполнения, после чего отображается сообщение **Collection Complete** (Сбор завершен). Нажмите **OK** для продолжения.
10. Чтобы просмотреть данные, нажмите пиктограмму в верхней части окна *PC200W*, чтобы открыть сервисную программу ***View*** (Просмотр).

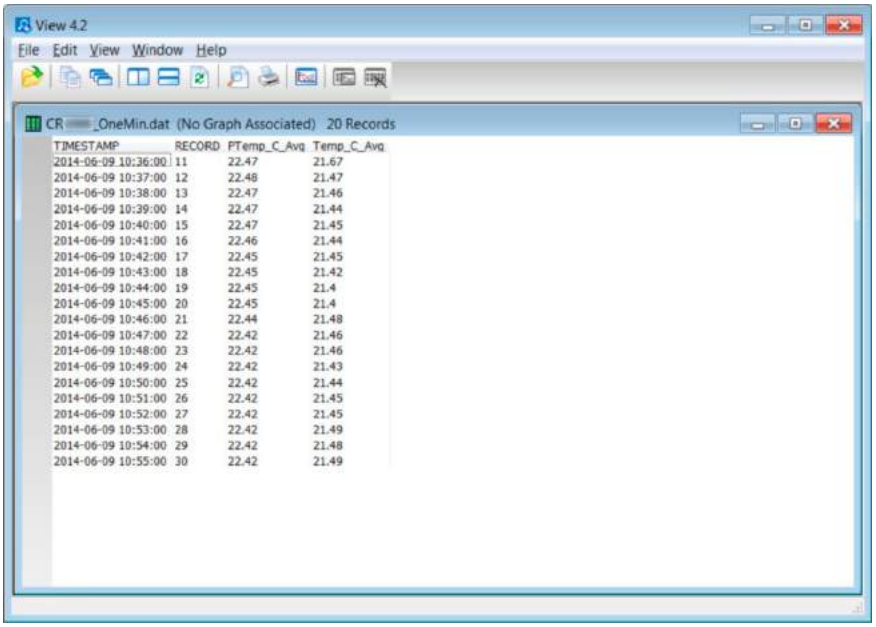
*Рисунок 14. Сервисная программа данных **View** (Просмотр) в PC200W*



4.7.5.6. Процедура: (шаги 11–12)

11. Нажмите на , чтобы открыть файл для просмотра. В диалоговом окне выберите файл **CR1OOO_OneMin.dat** и нажмите **Open (Открыть).**
12. Теперь отобразятся собранные данные.

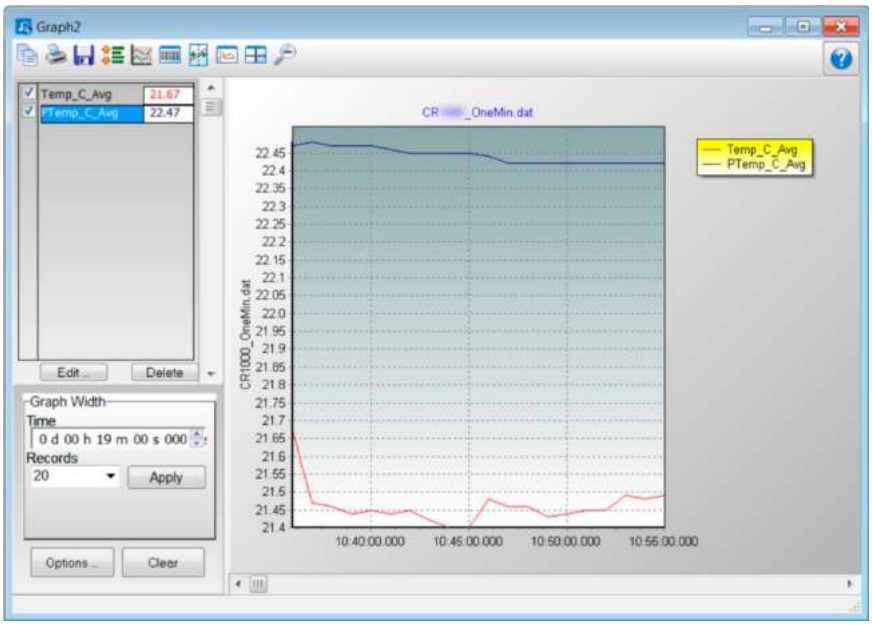
*Рисунок 15. Таблица данных **View** (Просмотр) в PC200W*



4.7.5.7. Процедура: (шаги 13–14)

13. Нажмите на заголовок любого столбца данных. Чтобы отобразить данные из этого столбца в виде линейного графика, нажмите на пиктограмму .
14. Закройте окна **Graph** (График) и **View** (Просмотр), а затем закройте программу *PC200W*.

*Рисунок 16. Линейный график **View** (Просмотр) в PC200W*



5. Общие сведения по системе

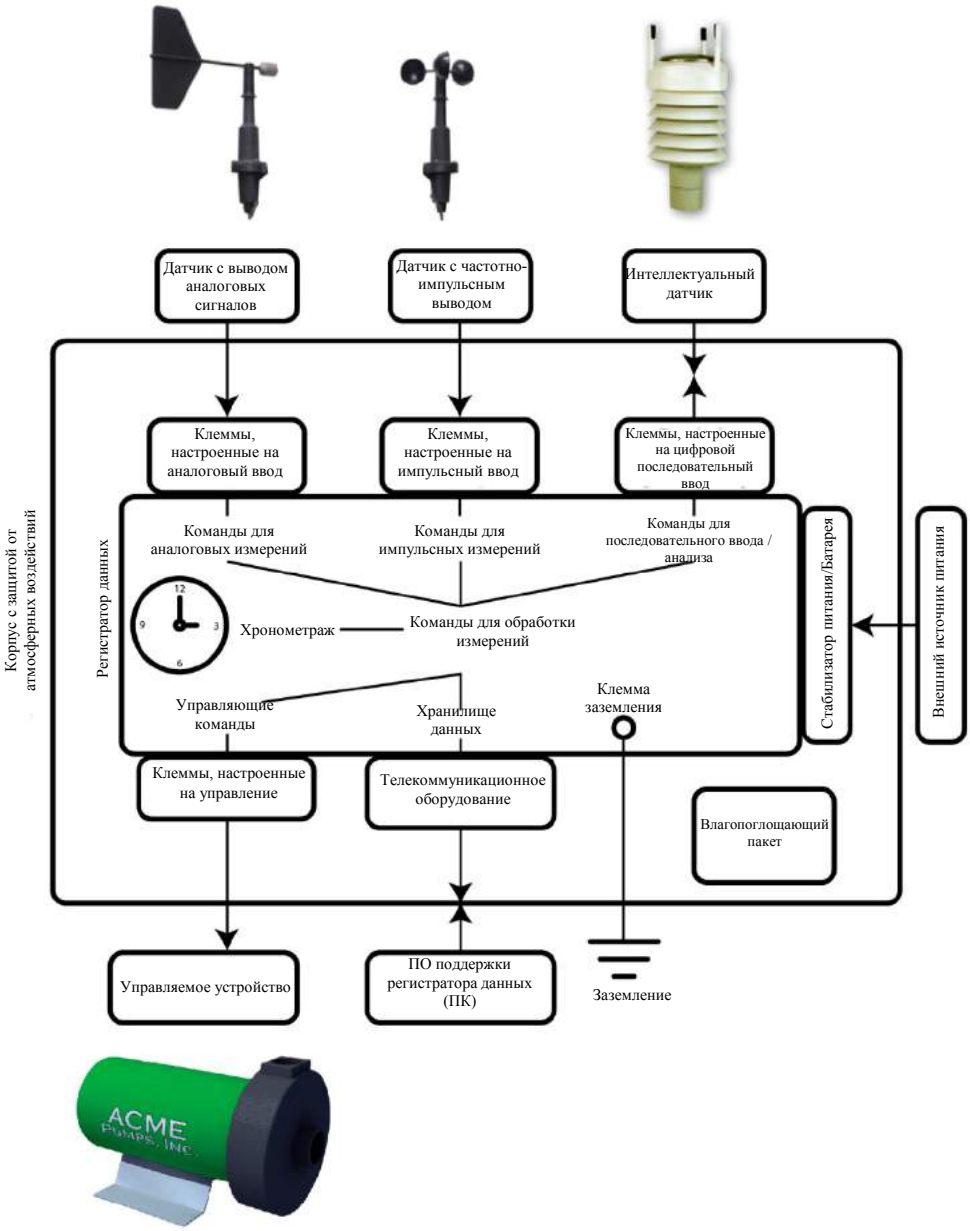
Подробнее:
- *Краткое руководство* *(стр. 41)*
- *Технические характеристики* *(стр. 97)*
- *Установка* *(стр. 99)*
- *Эксплуатация* *(стр. 303)*

Система сбора данных Campbell Scientific состоит из следующих основных компонентов:

- датчики;
- регистратор данных, который включает в себя следующие компоненты:
  - часы;
  - контрольно-измерительные схемы;
  - аппаратное обеспечение и встроенное ПО для обмена данными с телекоммуникационными устройствами;
  - введенная пользователем программа CRBasic;
- канал связи или внешнее запоминающее устройство;
- *ПО поддержки регистратора данных* *(стр. 512)*

На рисунке *Система сбора данных — общие сведения* *(стр. 62)* представлен общий случай системы сбора данных на базе CR1000.

Рисунок 17. Система сбора данных — общие сведения



5.1. Измерения — общие сведения

Связанные темы:

- *Датчики — краткое руководство* *(стр. 42)*
- *Измерения — общие сведения* *(стр. 62)*
- *Измерения — подробная информация* *(стр. 303)*
- *Датчики — списки* *(стр. 649)*

Большинство электронных датчиков, независимо от того, поставлены они компанией Campbell Scientific или нет, можно подключить непосредственно к CR1000.

Руководства, в которых рассматриваются альтернативные входные маршруты, такие как внешние мультиплексоры, периферийные измерительные устройства или беспроводные сенсорные сети, можно найти на сайте *www.campbellsci.com/manuals (http://www.campbellsci.com/manuals).* Также можно проконсультироваться с техническим специалистом Campbell Scientific.

В данном разделе рассматриваются прямые соединения между датчиком и регистратором данных и соответствующее программирование регистратора CR1000 по выполнению, обработке и хранению измерений. На коммутационной панели регистратора CR1000 предусматриваются клеммы для следующих измерительных вводов:

5.1.1. Хронометраж — общие сведения

Связанные темы:
- *Хронометраж — общие сведения* *(стр. 75)*
- *Хронометраж — подробная информация* *(стр. 303)*

Измерение времени является ключевой функцией регистратора CR1000. Измерение времени с помощью встроенных часов позволяет регистратору CR1000 присваивать временные метки данным, измерять интервалы между событиями и назначать время запуска функций управления.

5.1.2. Аналоговые измерения — общие сведения

Связанные темы:
- *Аналоговые измерения — общие сведения* *(стр. 63)*
- *Аналоговые измерения — подробная информация* *(стр. 305)*

Аналоговые датчики непрерывно выдают сигналы по напряжению или току, которые изменяются в зависимости от колебаний измеряемого явления. Датчики, совместимые с регистратором CR1000, выдают сигналы по напряжению. Совместимость сигналов по току обеспечивается благодаря шунтирующему сопротивлению.

Датчик подключается к клеммам **H/L]**, настраиваемым под дифференциальные (**DIFF**) или несимметричные (**SE**) входы. Например, дифференциальный канал 1 включает клеммы **1H** и **1L**, причем клемма **1H** предназначена для высоких значений, а **1L** — для низких.

5.1.2.1. Измерение напряжения — общие сведения

Связанные темы:
- Измерения по напряжению — технические характеристики
- *Измерения по напряжению — общие сведения* *(стр. 63)*
- *Измерения по напряжению — подробная информация* *(стр. 305)*

- Максимальный диапазон входного напряжения: ±5000 мВ
- Диапазон разрешающей способности измерительного прибора: 0,67–1333 мкВ

Несимметричные и дифференциальные соединения представлены на рисунках *Подключение аналогового датчика к несимметричному каналу 1* *(стр. 64)* и *Подключение аналогового датчика к дифференциальному каналу 1* *(стр. 64)***.** В таблице *Дифференциальные и несимметричные входные клеммы* *(стр. 65)* представлено назначение клемм для аналоговых входных каналов регистратора CR1000.

Концептуально аналоговые датчики с сигналом по напряжению генерируют два сигнала: высокого и низкого уровня. В редких случаях сигналом низкого уровня служит земля датчика. При несимметричных измерениях измеряются сигналы высокого уровня по отношению к заземлению, когда сигнал низкого уровня соединен с землей. При дифференциальных измерениях измеряются сигналы высокого уровня по отношению к сигналу низкого уровня. Обе конфигурации выполняют свои задачи, но более распространенной схемой являются дифференциальные измерения.

Дифференциальная конфигурация может существенно улучшить измерения по напряжению. Ниже приводятся условия, характерные для случаев, когда рекомендуется применять дифференциальные измерения:

- Ток заземления приводит к падению напряжения между датчиком и клеммой заземления сигнала. Токи > 5 мА, как правило, нежелательны. Такие значения тока могут подаваться датчиками типа «мост сопротивления» с возбуждением по напряжению, но данные токи подаются только во время применения возбуждения по напряжению. Обратные токи, связанные с возбуждением по напряжению, не могут влиять на несимметричные измерения низкого напряжения, если такая же клемма с возбуждением по напряжению не будет активирована во время несвязанных измерений.
- Измеренное напряжение составляет менее 200 мВ.

Рисунок 18. Подключение аналогового датчика к несимметричному каналу 1

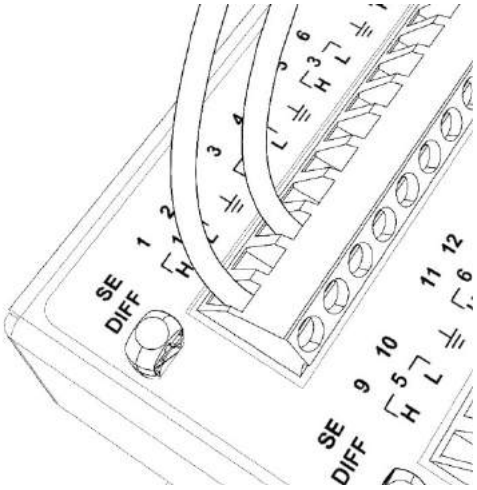


Рисунок 19. Подключение аналогового датчика к дифференциальному каналу 1

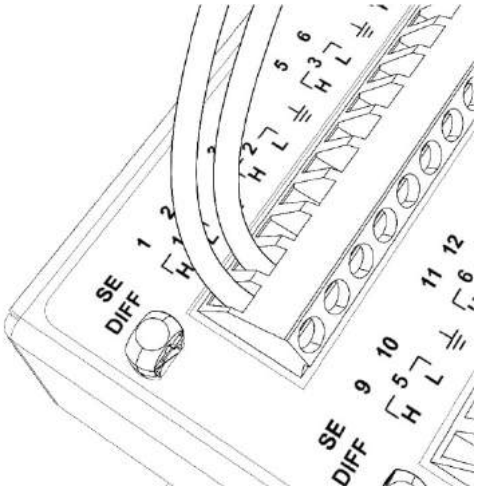


Таблица 2. Дифференциальные и несимметричные входные клеммы	
ДИФ. клеммы	***НЕСИМ. клеммы***
1H	**1**
1L	**2**
2H	**3**
2L	**4**
3H	**5**
3L	**6**
4H	**7**
4L	**8**
5H	**9**
5L	**10**
6H	**11**
6L	**12**
7H	**13**
7L	**14**
8H	**15**
8L	**16**

5.1.2.1.1. Несимметричные измерения — общие сведения

Связанные темы:

- *Несимметричные измерения — общие сведения* *(стр. 65)*
- *Несимметричные измерения — подробная информация* *(стр. 307)*

Несимметричные измерения оценивают разность потенциалов между клеммой, настроенной на несимметричный вход, и базовым заземлением. Последовательность измерения показана на рисунке *Упрощенная последовательность измерения напряжения* *(стр. 306)***.** Хотя предпочтительно использовать дифференциальные измерения, несимметричные измерения часто подходят для ситуаций, когда могут существовать несколько видов шумов и необходимо избежать проблем, вызванных блуждающими токами. Например, несимметричные измерения могут быть использованы в следующих случаях:

- Недостаточное количество дифференциальных клемм. Дифференциальные измерения используют в два раза больше клемм **H/L]**, чем несимметричные измерения.
- Требуется быстрая выборка. Несимметричные измерения выполняются примерно в два раза быстрее дифференциальных измерений.
- Датчик не предназначен для дифференциальных измерений. Многие датчики Campbell Scientific не предназначены для дифференциальных измерений, но погрешность несимметричного измерения обычно компенсируется большим запрограммированным возбуждением и/или выходным напряжением датчика.

Тем не менее следует помнить, что поскольку несимметричные измерения CR1000 используют заземление, то любое изменение в электрическом потенциале земли между датчиком и CR1000 приведет к ошибке. Это показано на следующих примерах:

- Если измерительный спай термопары, который используется для измерения температуры почвы, не изолирован, а потенциал заземления на датчике выше, чем в точке заземления CR1000, это приведет к погрешности измерения. Например, если разница заземления составляет 1 мВ относительно медно-константановой термопары, ошибка будет составлять приблизительно 25 °C.
- Если в газоанализаторе есть схема формирования сигнала, а CR1000 используют общий источник питания, различия в потреблении тока и сопротивлении выводов часто приводят к различному электрическому потенциалу земли на двух разных приборах, несмотря на использование общего заземления. Поэтому, чтобы избежать ошибок, дифференциальное измерение необходимо проводить на аналоговом выходе из внешнего блока формирования сигналов.

5.1.2.1.2. Дифференциальные измерения — общие сведения

Связанные темы:
- *Дифференциальные измерения — общие сведения* *(стр. 66)*
- *Дифференциальные измерения — подробная информация* *(стр. 308)*

Краткое описание. Если это возможно, используйте дифференциальное измерение напряжения.

Дифференциальное измерение позволяет осуществить оценку разности напряжений между двумя входными клеммами. Его последовательность показана на рисунке *Упрощенная последовательность дифференциального измерения напряжения* *(стр. 66)*. В нем используется множество автоматических измерений, результаты которых усредняются до конечного передаваемого значения. Например, последовательность дифференциального измерения с помощью команды **VoltDiff()** включает два измерения — сначала отношение высокого входа к низкому, а затем в обратном порядке. Изменение порядка входов перед вторым измерением позволяет отфильтровать общий для обоих выводов шум, а также небольшие ошибки, вызванные стыками различных металлов, которые присутствуют в любой измерительной электронике.

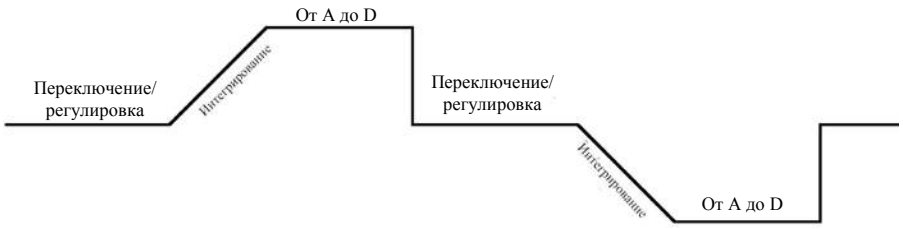


Рисунок 20. Упрощенная последовательность дифференциального измерения напряжения

5.1.2.2. Измерения тока — общие сведения

Связанные темы:
- *Измерения тока — общие сведения* *(стр. 66)*
- *Измерения тока — подробная информация* *(стр. 337)*

Измерение тока осуществляется за счет использования внешних резисторов для преобразования тока в напряжение и последующего измерения напряжения, как это описано в разделе *Дифференциальные измерения — общие сведения* *(стр. 66)*. Напряжение измеряется с помощью схемы измерения напряжения CR1000.

5.1.2.3. Измерения сопротивления — общие сведения

Связанные темы:

- Измерения сопротивления — технические характеристики
- *Измерение сопротивления — общие сведения* *(стр. 67)*
- *Измерение сопротивления — подробная информация* *(стр. 337)*
- *Измерение сопротивления — команды* *(стр. 551)*

Многие аналоговые датчики используют устройство с переменным сопротивлением в качестве основного чувствительного элемента. Эти элементы содержат в себе мост сопротивления или связанный контур. CR1000 может измерять большинство конфигураций мостовых схем. Измерение по мостовой схеме представляет собой отдельный вариант измерения напряжения. Например:

- Тензорезистор: сопротивление в преобразователе давления тензорезистора коррелирует с давлением воды.
- Положение потенциометра: изменение сопротивления во флюгере потенциометра коррелирует с изменением направления ветра.

5.1.2.3.1. Напряжение возбуждения

Сопротивление моста определяется путем измерения разности между известным напряжением, приложенным к возбуждаемому звену (входу) резистора моста, и напряжением, измеренным на выходном звене. CR1000 поставляет точное напряжение возбуждения через клеммы Vx. Возвратное напряжение измеряется на клеммах **H/L]**, которые могут быть настроены на несимметричный или дифференциальный вход. Примеры подключения мостового датчика с использованием возбуждения напряжения показаны на рисунках *Полумостовое подключение — потенциометр флюгера* *(стр. 67)* и *Полное мостовое подключение — измерительный преобразователь давления* *(стр. 68)*.

Рисунок 21. Пример полумостового подключения — потенциометр флюгера

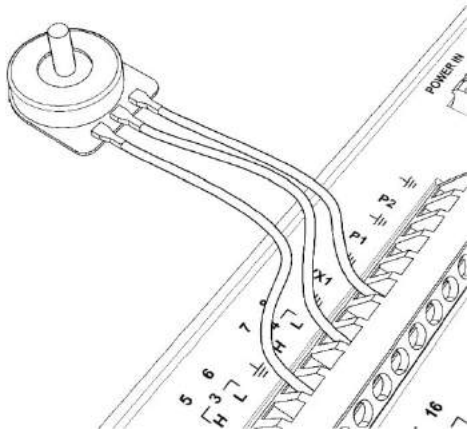
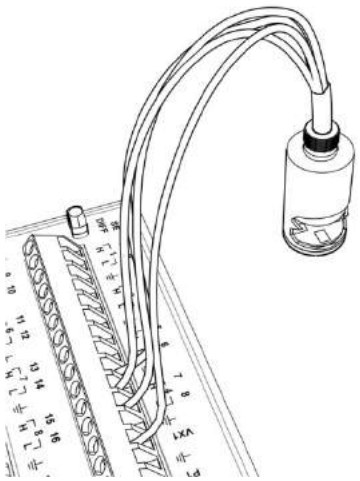


Рисунок 22. Пример полного мостового подключения — измерительный преобразователь давления



5.1.2.4. Тензометрические измерения — общие сведения

Связанные темы:
- *Тензометрические измерения — общие сведения* (стр. 68)
- *Тензометрические измерения — подробная информация* (стр. 342)
- *Примеры FieldCalStrain()* (стр. 223)

Тензометрические измерения, как правило, связаны с анализом напряжения конструкции. При проведении тензометрических измерений желательно предварительно проконсультироваться со специалистами Campbell Scientific.

5.1.3. Импульсные измерения — общие сведения

Связанные темы
- Импульсные измерения — технические характеристики
- *Импульсные измерения — общие сведения* (стр. 68)
- *Импульсные измерения — подробная информация* (стр. 349)
- *Импульсные измерения — команды* (стр. 553)

Выходной сигнал, формируемый датчиком импульсов, представляет собой серию волн напряжения. Датчик соединяет свой выходной сигнал с измеряемым явлением с помощью модуляции частоты волны. CR1000 определяет переход между состояниями, поскольку каждая волна колеблется между крайними напряжения (от высокого до низкого или низкого до высокого). Измерения обрабатываются и представляются в виде количества, частоты или временных характеристик.

Клеммы **P** могут настраиваться на входные импульсы, чтобы измерять количество или частоту следующих типов сигналов:

- Высокочастотный прямоугольный сигнал 5 В пост. тока
- Замыкание переключателя
- Низкоуровневый переменный ток

Клеммы **C** настраиваются на вход для следующих показателей:

- Состояние
- Подсчет крайних значений

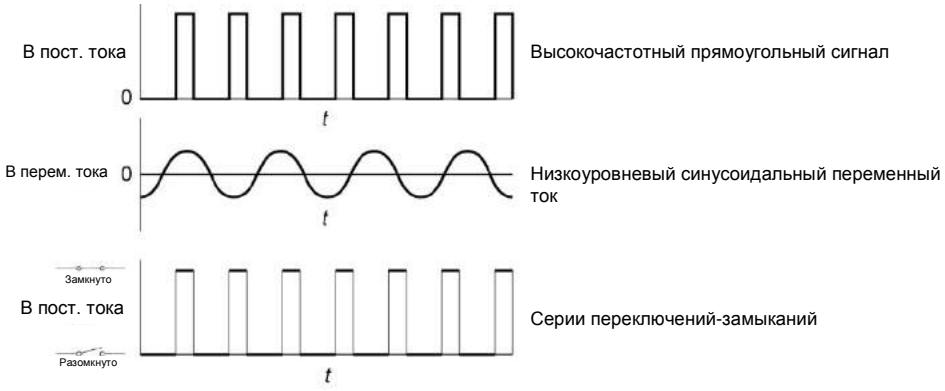
- Время крайних значений
  - Разрешение — 540 нс

Примечание. Датчик с усреднением данных за период имеет частотный выход, но он подключен к **НЕСИМ.** клемме, настроенной на ввод данных с усреднением за период, измеренных с помощью команды **PeriodAverage()** (см. раздел *Среднее значение за период — общие сведения* *(стр. 70)*).

5.1.3.1. Измерение импульсов

Импульсные выходы могут различаться. Эти различия показаны на рисунке *Типы выходных сигналов импульсного датчика* *(стр. 69)*.

Рисунок 23. Типы выходных сигналов импульсного датчика



5.1.3.2. Каналы импульсного входа

В таблице *Каналы импульсного входа и измерения* *(стр. 69)* указаны устройства, каналы и параметры измерения импульсных сигналов.

Таблица 3. Клеммы импульсного входа и измерения

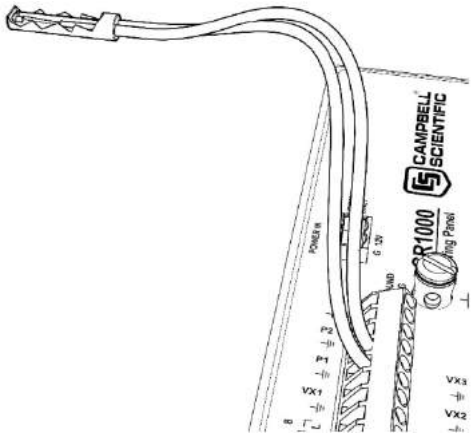
Клеммы импульсного входа	***Тип входа***	***Параметр данных***	***Команда CRBasic***
Клемма **P**	• Низкоуровневый переменный ток • Высокочастотный • Переключение-замыкание	• Значение • Частота • Скользящее среднее частоты	**PulseCount()**
Клемма **C**	• Низкоуровневый переменный ток с модулем *LLAC4* *(стр. 646)* • Высокочастотный • Переключение-замыкание	• Значение • Частота • Скользящее среднее частоты • Интервал • Период • Состояние	**PulseCount()** **TimerIO()**

5.1.3.3. Подключение импульсного датчика

Читать далее. См. раздел *рекомендации для импульсных измерений* *(стр. 356)*

Пример подключения импульсного датчика показан на рисунке *Пример импульсного входа — переключатель анемометра* *(стр. 70)***.** Импульсные датчики имеют два активных провода, один из которых — заземление. Подключите провод заземления к клемме (заземление сигнала). Подключите второй провод к клемме **P**. Иногда датчику требуется питание от CR1000, в этом случае могут быть использованы два провода питания, один из которых будет заземлением. Подключите заземление к клемме **G**. Не следует путать импульсный провод с положительным проводом питания, это может привести к повреждению датчика или CR1000. Для некоторых датчиков переключения-замыкания может потребоваться нагрузочный резистор.

Рисунок 24. Пример импульсного входа — анемометр



5.1.4. Среднее значение за период — общие сведения

Связанные темы:
- Среднее значение за период — технические характеристики
- *Среднее значение за период — общие сведения* *(стр. 70)*
- *Среднее значение за период — подробная информация* *(стр. 360)*

CR1000 может измерять период аналогового сигнала.

Номерные **НЕСИМ.** клеммы могут настаиваться на среднее значение за период:

- Коэффициент усиления напряжения: 1, 10, 33, 100
- Максимальная частота: 200 кГц
- Разрешение: 136 нс

Примечание. Для подсчета импульсов и среднего значения за период используются датчики с частотным выходом. Однако подсчеты импульсов и среднего значения за период имеют определенные отличия. Для подсчета импульсов используются специализированные устройства — накопительные счетчики импульсов, которые всегда контролируют сигнал, даже когда CR1000 находится в промежутке между этапами выполнения программы сканирования. Напротив, устройства для измерения среднего значения за период контролируют входной сигнал только во время программы сканирования. Следовательно, сканирование количества импульсов обычно выполняется значительно реже, чем сканирование среднего значения за период. Счетчики импульсов могут быть более восприимчивы к низкочастотным шумам, поскольку они всегда фиксируют шум, даже если период усреднения позволяет фильтровать его, поэтому большую часть времени они находятся в «спящем» режиме. Измерение количества импульсов

не подходит для датчиков, отключаемых между сеансами сканирования, при этом измерение среднего значения за период работает хорошо, так как измерения можно синхронизировать с процессом сканирования, чтобы они выполнялись только тогда, когда датчик включен и передает сигнал.

Для измерения среднего значения за период используются высокочастотные цифровые часы для измерения интервалов между сигнальными переходами, а при измерении количества импульсов просто измеряется число подсчетов. В результате измерение среднего значения за период дает гораздо более высокое разрешение по частоте за интервал по сравнению с измерением количества импульсов. Разрешающая способность измерения количества импульсов может быть улучшена за счет расширения интервала измерений, увеличения интервала сканирования, а также путем усреднения. Для получения информации о разрешающей способности по частоте см. *Разрешающая способность по частоте (стр. 353)*.

5.1.5. Вибрационные струнные измерения — общие сведения

Связанные темы:

- Вибрационные струнные измерения — технические характеристики
- *Вибрационные струнные измерения — общие сведения (стр. 71)*
- *Вибрационные струнные измерения — подробная информация (стр. 361)*

Вибрационные струнные датчики обеспечивают долгосрочную стабильность измерений в различных экологических и промышленных областях применения. CR1000 может фиксировать данные этих датчиков как непосредственно, так и через интерфейсные модули.

Терморезистор, который входит в большинство датчиков, позволяет проводить измерения для компенсации температурных погрешностей.

Измерение резонансной частоты с помощью среднего значения за период является классическим методом, однако Campbell Scientific разработала статические и динамические методы спектрального анализа (*VSPECT™ (стр. 532)*), которые дают превосходное подавление шумов, более высокое разрешение, возможность получать данные диагностики и проводить динамические измерения VSPECT с частотой до 333,3 Гц. Динамические измерения требуют добавления интерфейсного модуля.

НЕСИМ. клеммы можно настроить на пространственно-временные струнные измерения, этот метод в настоящее время заменен в большинстве случаев вибрационные струнным анализом *VSPECT (стр. 532)*. Для получения дополнительной информации см. приложение *Перечень модулей вибрационных струнных входов (стр. 647)*

5.1.6. Снятие показаний интеллектуальных датчиков — общие сведения

Связанные темы:

- *Снятие показаний интеллектуальных датчиков — общие сведения (стр. 71)*
- *Снятие показаний интеллектуальных датчиков — подробная информация (стр. 362)*

Интеллектуальный датчик оснащен независимой схемой измерения, которая проводит основные измерения и передает связанные данные в CR1000. Интеллектуальные датчики отличаются друг от друга режимами выводов. Многие из них имеют несколько вариантов выводов. Параметры вывода, которые поддерживаются CR1000: *SDI-12 (стр. 267)*, *RS-232 (стр. 245)*, *Modbus (стр. 411)* и *DNP3 (стр. 408)*.

На указанных клеммах поддерживаются измерения следующих типов интеллектуальных датчиков:

- Устройства SDI-12: **C**
- Синхронные устройства для измерения (SDM): **C**
- Интеллектуальные датчики: клеммы **C**, порт **RS-232** и порт **CS B/B** с соответствующим интерфейсом
- Сети Modbus или DNP3: порт **RS-232** и порт **CS B/B** с соответствующим интерфейсом
- Другие устройства последовательного ввода/вывода: клеммы **C**, порт **RS-232** и порт **CS B/B** с соответствующим интерфейсом

5.1.6.1. Поддержка датчика SDI-12 — общие сведения

Связанные темы:
- *Поддержка датчика SDI-12 — общие сведения* *(стр. 72)*
- *Поддержка датчика SDI-12 — подробная информация* *(стр. 363)*
- *Последовательный интерфейс ввода/вывода: поддержка датчика SDI-12 — программные ресурсы* *(стр. 267)*
- *Поддержка датчика SDI-12 — команды* *(стр. 555)*

SDI-12 — протокол интеллектуальных датчиков, который использует один порт SDI-12 и питается от постоянного тока 12 В. Он полностью поддерживается регистратором данных CR1000. См. таблицу *Определения клемм CR1000* *(стр. 76)*, где указаны клеммы C, которые могут быть настроены на вход SDI-12. Для получения дополнительной информации о поддержке SDI-12, см. раздел *Последовательный интерфейс ввода/вывода: поддержка датчика SDI-12 — подробная информация* *(стр. 267)*.

5.1.6.2. RS-232 — общие сведения

CR1000 имеет 6 портов RS-232, как это показано на рисунке *Настраиваемые клеммы для входа RS-232* *(стр. 73)*.

Примечание. При наличии совместимого адаптера порт **CS B/B** часто используется в качестве порта ввода/вывода RS-232.

Как показано на рисунке *Использование RS-232 и цифрового ввода/вывода для считывания устройств RS-232* *(стр. 73)*, датчики RS-232 часто подключаются к парам клемм **C**, настроенным последовательного ввода/вывода, к порту **RS-232** или последовательному вводу/выводу **CS B/B** с соответствующим адаптером. Порты могут быть настроены на скорость передачи данных, четность, стоп-бит и так далее, как описано в *Справка к редактору CRBasic*.

Рисунок 25. Настраиваемые клеммы для ввода RS-232

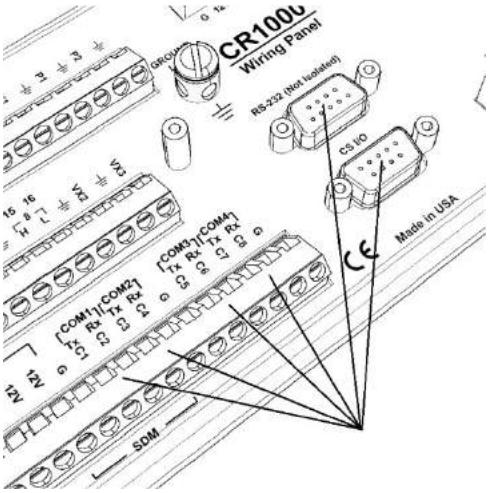
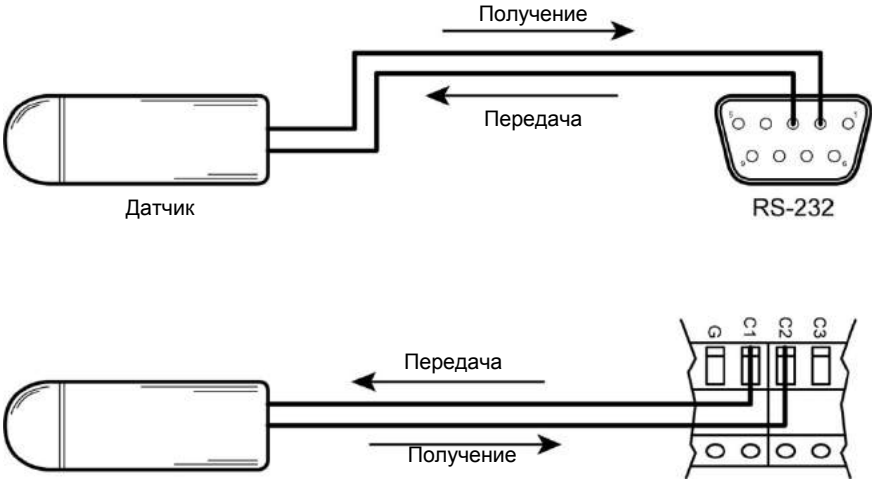


Рисунок 26. Использование RS-232 и цифрового ввода/вывода при чтении устройств RS-232



5.1.7. Калибровка на месте — общие сведения

Связанные темы:

- *Калибровка на месте — общие сведения* *(стр. 73)*
- *Калибровка на месте — подробная информация* *(стр. 210)*

Калибровка увеличивает точность измерительного устройства, регулируя его выходные параметры или измерение выходных параметров таким образом, чтобы они соответствовали верифицированным эталонным величинам. Непосредственная регулировка выходного сигнала датчика является предпочтительным вариантом, но он не всегда доступен. Добавляя команды **FieldCal()** или **FieldCalStrain()** к программе CR1000 CRBasic, можно регулировать линейный датчик, меняя применяемые к измерению запрограммированный множитель и сдвиг.

5.1.8. Эффекты подключения — общие сведения

Связанные темы:
- *Эффекты подключения — общие сведения* (стр. 74)
- *Эффекты подключения — подробная информация* (стр. 364)

Проводка кабелей может оказывать значительное влияние на реакцию и точность датчика. Это, как правило, не касается датчиков Campbell Scientific. Датчики Campbell Scientific разработаны для оптимальной работы с установленными на заводе кабелями.

5.1.9. Синхронизация измерений — общие сведения

Связанные темы:
- *Синхронизация измерений — общие сведения* (стр. 74)
- *Синхронизация измерений — подробная информация* (стр. 365)

Выбор времени измерения обычно контролируется на основании часов CR1000. Когда датчики в сети эксплуатируются с применением одного CR1000, время измерения синхронизируется, часто в течение нескольких миллисекунд, в зависимости от количества датчиков и типа измерений. Из-за большого количества датчиков, ограничений длины кабеля или больших расстояний между участками измерений может потребоваться установить несколько CR1000.

5.2. Управление ПЛК — общие сведения

Связанные темы:
- *Управление ПЛК — общие сведения* (стр. 74)
- *Управление ПЛК — подробная информация* (стр. 244)
- *Модули управления ПЛК — общие сведения* (стр. 368)
- *Модули управления ПЛК — списки* (стр. 648)
- *Контроль ПЛК — команды* (стр. 562)
- Переключаемое выходное напряжение — технические характеристики
- *Переключаемое выходное напряжение — общие сведения* (стр. 78)
- *Переключаемое выходное напряжение — подробная информация* (стр. 103)

Этот раздел предполагается расширить. Далее приведено несколько советов.

- Мастер программирования Short Cut имеет возможность быстрого включения и выключения.
- С помощью CR1000 можно проводить ПИД-регулирование. Для получения дополнительной информации обратитесь к специалистам Campbell Scientific.
- При управлении ПИД-алгоритмом задержка между процессором (вход алгоритма) и управлением (выход алгоритма) нежелательна. Задержка не будет возникать в любом *последовательном режиме* (стр. 527) или *режиме поточной обработки* (стр. 523), предполагая, что запрограммирован соответствующий быстрый интервал сканирования, и программа не пропускает сканирование. В последовательном режиме при возникновении какой-либо задачи, из-за которой время обработки выходит за интервал сканирования, будут происходить пропуски сканирования, что может привести к сбою ПИД-регулятора. В режиме поточной обработки при наличии буфера сканирования соответствующего размера пропуски сканирования не возникают. Однако в ПИД-регуляторе может произойти сбой в команде обработки из-за буфера сканирования.
- Чтобы избежать этих потенциальных проблем, в группе команды по обработке в программе CRBasic применяются **ProcHiPri** и **EndProcHiPri.** Команды по обработке этими командами получают одинаково высокий приоритет с командами измерения и не попадают в буфер сканирования при увеличении времени обработки. ProcHiPri и

EndProcHiPri могут быть недоступны в *Редакторе CRBasic*. Но вы можете ввести их в любом случае, и компилятор их распознает.

5.3. Регистратор данных — общие сведения

Связанные темы:
- *Регистратор данных — быстрый запуск* (стр. 43)
- *Регистратор данных — общие сведения* (стр. 75)
- *Регистратор данных — список* (стр. 645)

Регистратор данных CR1000 является основным компонентом системы сбора данных.

Это точный прибор, предназначенный для тяжелых условий эксплуатации с низким энергопотреблением. ЦП, аналоговые и цифровые измерения, аналоговые и цифровые выходы, а также использование памяти регулируются операционной системой, встроенными часами и прикладной программой, написанной на CRBasic.

Прикладная программа пишется на CRBasic — языке программирования, который включает процедуры анализа и стандартный набор основных команд для измерения и обработки данных. *Short Cut* (стр. 528) — очень удобное приложение, которое можно использовать для написания программ, используемых в основных сценариях измерений и управления. *Редактор CRBasic* — программное приложение, доступное в некоторых пакетах *программной поддержки регистратора данных* (стр. 512), которое используется для написания более сложных программ.

Данные измерений сохраняются в энергонезависимой памяти. В большинстве случаев сохранение каждого отдельного измерения не требуется. Чаще всего измерения суммируются в статистические или расчетные сводки. CR1000 может оценить запрограммированные команды последовательно (последовательный режим) или в более эффективном режиме поточной обработки. В режиме поточной обработки CR1000 определяет порядок выполнения команд.

5.3.1. Хронометраж — общие сведения

Связанные темы:
- *Хронометраж — общие сведения* (стр. 75)
- *Хронометраж — команды* (стр. 578)

Почти все функции CR1000 зависят от внутренних часов. Операционная система и пользовательская программа CRBasic используют внутренние часы для распределения операций во времени. Программа CRBasic использует данные о времени для выполнения различных команд, но время здесь почти всегда представлено в форме «время в интервале времени». Например, 6:00 утра в CRBasic представлено в следующем виде: «360-я минута в 1440-минутном интервале», 1440 — это общая продолжительность дня, а 360-я минута соответствует 6:00 утра.

Ноль означает начало интервала, то есть начало секунд, минут, часов или дней. Например, 0 минут в 1440-минутном интервале соответствует полуночи. При программировании недельного интервала он начинается в полночь понедельника.

5.3.2. Коммутационная панель — общие сведения

Связанные темы
- *Коммутационная панель — быстрый запуск* *(стр. 43)*
- *Коммутационная панель — общие сведения* *(стр. 76)*
- *Измерение и контроль внешнего оборудования* *(стр. 366)*

Коммутационная панель CR1000 является интерфейсом для большинства функций. Эти функции описаны в следующих разделах, где рассмотрены функции подключения панели, показанные на рисунке *Коммутационная панель* *(стр. 44)***.** В таблице *Описание клемм CR1000* *(стр. 76)* приведено описание функции различных клемм коммутационной панели. Измерение и контроль периферийных устройств расширяют возможности ввода и вывода на коммутационной панели.

Рисунок 27. Коммутационная панель

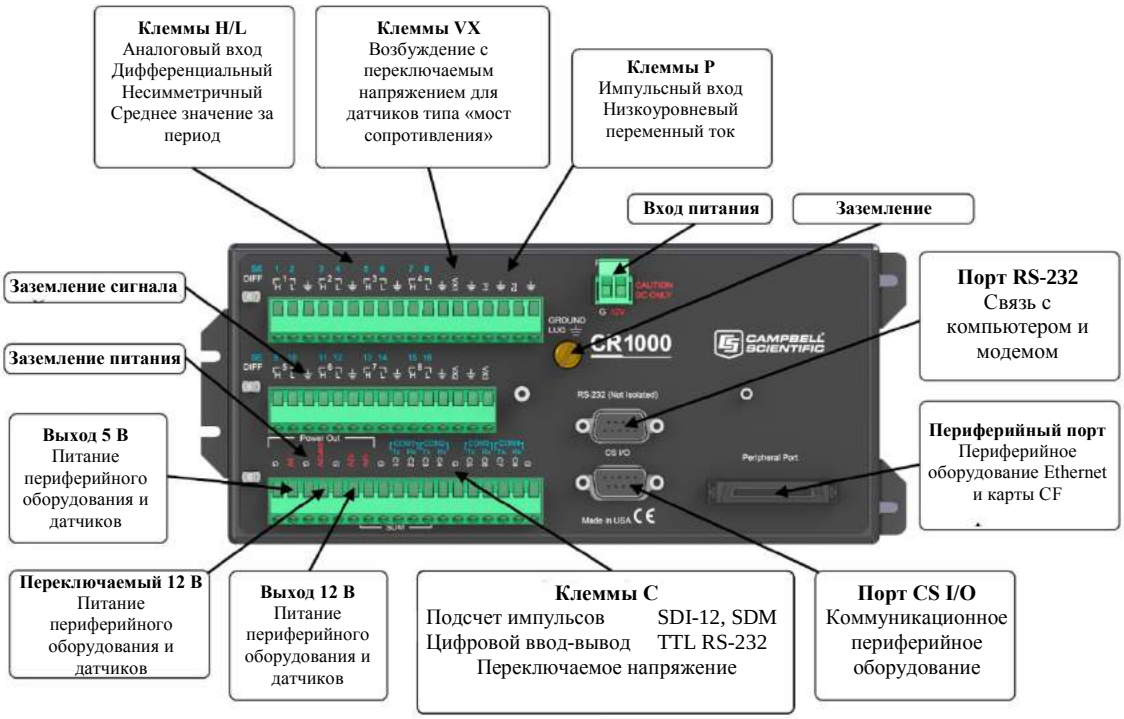


Таблица 4. Описание клемм коммутационной панели CR1000

Таблички		SE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16							COM1		COM2		COM3		COM4							
		DIFF	┌1┐		┌2┐		┌3┐		┌4┐		┌5┐		┌6┐		┌7┐		┌8┐								Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx						
			H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	VX1	VX2	VX3	P1	P2	C1	C2	C3	C4	C5	C6	C7	C8	5 В	12 В	12 В	SW-12	RS-	CS I/O	Макс.
Функция	Аналоговый вход																																					
	Несимметричный		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓																			16
	Дифференциальный (выс./низк.)		✓		✓		✓		✓		✓		✓		✓		✓																				8
	Среднее аналоговое значение за период		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓																			16
	Вибрационный струнный датчик [2]		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓																			16
	Аналоговый вывод																																					
	Переключаемое точное напряжение																		✓	✓	✓																	3
	Подсчет количества																																					
	Замыкание переключателя																					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓							10
	Высокая частота																					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓							10
	Низкоуровневый переменный ток																					✓	✓															2
	Цифровой ввод/вывод																																					
	Контроль																							✓	✓	✓	✓	✓	✓	✓	✓							8
	Состояние																							✓	✓	✓	✓	✓	✓	✓	✓							8
	Общий ввод/вывод (TX, RX)																							✓		✓		✓		✓								4
	Широтно-импульсная модуляция																							✓	✓	✓	✓	✓	✓	✓	✓							8
	Таймер ввода/вывода																							✓	✓	✓	✓	✓	✓	✓	✓							8
	Прерывание																							✓	✓	✓	✓	✓	✓	✓	✓							8
	Непрерывный, регулируемый [3]																																					
	5 В пост. тока																															✓						1
	Непрерывный, нерегулируемый [3]																																					
	12 В пост. тока																																✓	✓				2
	Переключаемый, регулируемый [3]																																					
	5 В пост. тока																							✓	✓	✓	✓	✓	✓	✓	✓							8
	Переключаемый, нерегулируемый [3]																																					
	12 В пост. тока																																		✓			1
	UART																																					
	Полный RS-232 (TX/RX)																																			✓	✓ 4	2
	TTL RS-232 (TX/RX)																							✓		✓		✓		✓								4
	SDI-12																							✓		✓		✓		✓								4
	SDM (дата/часы/включение)																							✓														1

[1] Можно использовать модули расширения клемм.

[2] Статическое измерение в частотной области.

[3] См. таблицу *Источник тока и ограничения при снижении* *(стр. 103)*.

[4] Требуется интерфейсное устройство для входа датчика. См. таблицу *Интерфейсы последовательного ввода/вывода в RS-232* *(стр. 651)*.

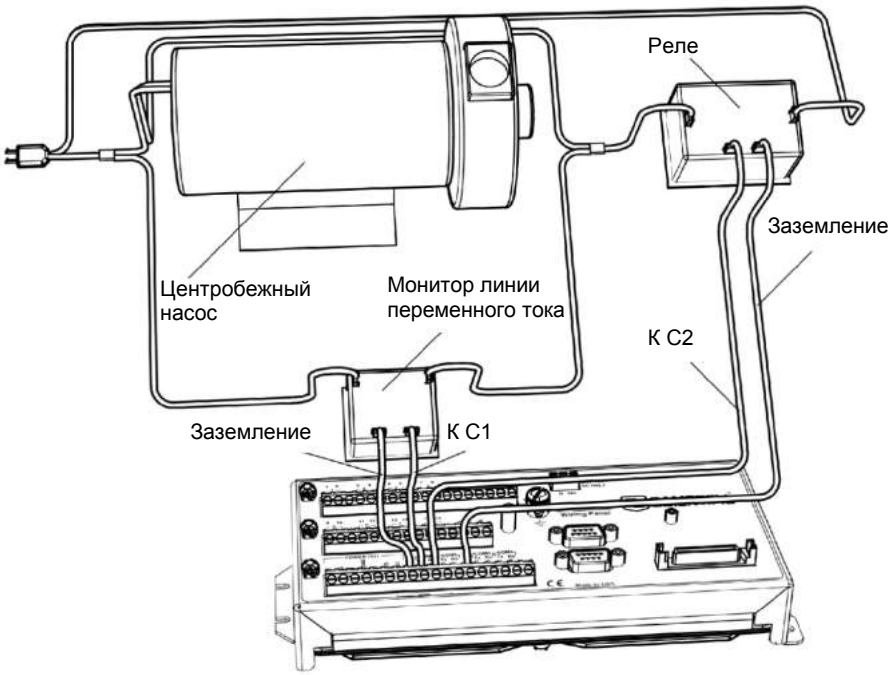
5.3.2.1. Переключаемое выходное напряжение — общие сведения

Связанные темы:
- Переключаемое выходное напряжение — технические характеристики
- *Переключаемое выходное напряжение — общие сведения* *(стр. 78)*
- *Переключаемое выходное напряжение — подробная информация* *(стр. 103)*
- *Управление ПЛК — общие сведения* *(стр. 74)*
- *Управление ПЛК — подробная информация* *(стр. 244)*
- *Модули управления ПЛК — общие сведения* *(стр. 368)*
- *Модули управления ПЛК — списки* *(стр. 648)*
- *Контроль ПЛК — команды* *(стр. 562)*

Клеммы **C** могут выбираться в виде двоичных входов, выходов управления или коммуникационных портов. Краткое описание измерительных функций см. в разделе *Измерения — общие сведения* *(стр. 62)*. Другие функции включают обработку прерываний, управляемых устройством, управляемых асинхронных коммуникаций и связи SDI-12. В таблице *Описание клемм CR1000* *(стр. 76)* приведены доступные варианты.

Рисунок *Управление и контроль с клеммами C* *(стр. 79)* иллюстрирует простую сферу применения, где клемма **C**, настроенная на цифровой вход, и вторая клемма для управления вывода используются для управления устройством (включения и выключения), а также для контроля состояния устройства (независимо от того, включено устройство или выключено).

*Рисунок 28. Контроль и управление клеммами **C***



5.3.2.2. Напряжение возбуждения — общие сведения

Связанные темы:
- Напряжение и ток возбуждения — технические характеристики
- *Напряжение возбуждения — общие сведения* *(стр. 79)*

CR1000 имеет несколько клемм, предназначенных для подачи напряжения на периферийные устройства, датчики или управляющие устройства:

- Напряжение возбуждения (переключаемый аналоговый выход) — клеммы **Vx** подают точное напряжение в диапазоне ±2500 мВ. Эти клеммы обычно используются для измерений резистивного моста. Каждая клемма пропускает до ±25 мА.
- Цифровой ввод/вывод — клеммы **C**, настроенные на включение/выключение, ШИМ (широтно-импульсную модуляцию) или МДИ (модуляцию по длительности импульса) на **C4, C5** и **C7**.
- Переключаемый 12 В постоянного тока — клеммы **SW12**. Первичное напряжение аккумулятора под управлением программы для переключения внешних устройств (например, датчиков влажности) требует номинального напряжения 12 В постоянного тока. Клеммы **SW12** могут пропускать до 900 мА. См. таблицу *Источник тока и ограничения при снижении* *(стр. 103)***.**
- Непрерывный аналоговый выход — доступен путем добавления периферийного аналогового выводящего устройства Campbell Scientific. Для получения информации о доступных модулях расширения см. раздел *Модули аналогового вывода* *(стр. 367)***.**

5.3.2.3. Заземляющие клеммы

Читать далее. См. в разделе *Заземление (стр. 105)*.

Правильное заземление обеспечивает стабильность и защиту системы сбора данных. Это самый простой и экономичный способ защиты от потери данных — и часто самый непопулярный. Следующие клеммы предназначены для подключения датчиков и заземления регистратора CR1000:

- Заземление сигнала (⏚) — применяется для несимметричных аналоговых входов, импульсных входов, возвратов возбуждения, а также в качестве заземления для экранированных проводов датчиков. Сигнал, который возвращается на импульсные входы ⏚, должен использовать клеммы, расположенные рядом с импульсным входом клеммы. При этом датчики токовой петли должны быть заземлены к силовому заземлению.
- Силовое заземление (**G**) — обратная линия к **5 В, SW12, 12 В**, датчикам токовой петли и **C**, настроенная на управление. Использование заземления **G** для этих выходов при необходимости минимизирует большой ток через секцию коммутационной панели для аналогового измерения напряжения, приводящий к погрешности измерения несимметричного напряжения.
- Заземляющая клемма (⏚) — точка подключения заземляющего провода большого сечения. Для обеспечения нулевого потенциала CR1000 и снижения импульсных помех необходимо надлежащее заземление. Рекомендуется использовать провода сечением 14 AWG и выше.

5.3.2.4. Силовые клеммы

Связанные темы:
- Источники питания — технические характеристики
- *Источники питания — краткое руководство (стр. 44)*
- *Источники питания — общие сведения (стр. 85)*
- *Источники питания — общие сведения (стр. 100)*
- *Источники питания — продукты (стр. 657)*
- *Источники питания (стр. 101)*
- *Диагностика — источники питания (стр. 494)*

5.3.2.4.1. Подача питания

Разъем **POWER IN** (ВХОД ПИТАНИЯ) является точкой подключения внешних источников питания.

5.3.2.4.2. Выходные силовые клеммы

Примечание. Для получения дополнительной информации об использовании CR1000 в качестве источника питания для датчиков и периферийных устройств см. раздел *Переключаемое выходное напряжение — подробная информация (стр. 103)*.

CR1000 может использоваться в качестве источника питания для датчиков и периферийных устройств. Можно использовать следующие напряжения:

- Клеммы **12 В**: нерегулир. ном. 12 В пост. тока. Этот блок питания отслеживает первичное напряжение питания CR1000, поскольку оно может подняться выше или упасть ниже уровня, необходимого для датчика или периферийного устройства. Следует принять меры предосторожности, чтобы предотвратить повреждение датчиков или периферийных устройств из-за избыточного или недостаточного

напряжения, а также свести к минимуму ошибки, связанные с измерением датчиками с пониженной мощностью. См. раздел *Источники питания — общие сведения (стр. 85)*.

- Клеммы **5 В**: регулир. 5 В пост. тока при 300 мА. Питание 5 В постоянного тока регулируется в пределах нескольких милливольт, при условии что напряжение основного источника питания CR1000 не опускается ниже <MinPwrSupplyVolts> (минимальное напряжение источника питания).

5.3.2.5. Коммуникационные порты

Читать далее. См. в разделе *RS-232 и TTL (стр. 362)*, *Поиск данных и связь — подробная информация (стр. 391)* и *PakBus — общие сведения (стр. 88)*.

CR1000 оснащен аппаратными портами, которые позволяют поддерживать связь с другими устройствами и сетями, например:

- ПК
- Интеллектуальные датчики
- Протокол Modbus и сети DNP3
- Ethernet
- Модемы
- Сети Campbell Scientific PakBus
- Другие регистраторы Campbell Scientific
- Внешнее оборудование регистратора Campbell Scientific

Коммуникационные порты:

- **CS B/B**
- **RS-232**
- SDI-12
- SDM
- CPI (требуется периферийное устройство)
- Ethernet (требуется периферийное устройство)
- **Периферийный порт** — поддерживает Ethernet и карты памяти CompactFlash

5.3.2.5.1. Порт CS B/B

Читать далее. См. в разделе *Распиновка последовательного порта (стр. 633)*.

- Один девятиконтактный порт, обозначенный **CS B/B**, для связи с ПК или модемом через интерфейсы связи, модемы или периферийные устройства Campbell Scientific. Интерфейсы связи CS B/B указаны в приложении *Список модулей последовательного ввода/вывода (стр. 646)*.

Примечание. Связь CS B/B обычно функционирует удовлетворительно только при длине кабеля, не превышающей приблизительно 1 м.

5.3.2.5.2. Порты RS-232

Примечание. Порты RS-232 обычно функционируют удовлетворительно при емкостном сопротивлении кабеля 2500 пФ или при длине стандартного кабеля для последовательной передачи данных, не превышающей приблизительно 15 м.

- Один из девятиконтактных портов DCE **RS-232** обычно используются для связи ПК с *ПО поддержки регистратора данных* *(стр. 654)* или с подключенным сторонним модемом. Вместе с прилагаемым адаптером нуль-модема он служит в качестве устройства DTE.

Читать далее. См. в разделе *Распиновка последовательного порта* *(стр. 633)*.

- Двухклеммный порт RS-232 (TX и RX) может быть настроен следующим образом:
  - до четырех TTL портов, настроенных на клеммы **C**.

Примечание. Порты RS-232 не *изолированы* *(стр. 518)*.

5.3.2.5.3. Периферийный порт

При условии подключения некоторых модулей для карт памяти CF Campbell Scientific и аппаратуры IP-сети. См. приложения *Список сетевых ссылок* *(стр. 652)* и *Устройства хранения данных — список* *(стр. 653)*. Меры предосторожности при использовании карт памяти см. в разделе *Карта памяти (CRD: диск) — общие сведения* *(стр. 89)*.

Читать далее. См. в разделе *TCP/IP* *(стр. 289)*.

- Один многоконтактный порт, обозначенный как **периферийный порт**.

5.3.2.5.4. Порты SDI-12

Читать далее. См. в разделе *Последовательный интерфейс ввода/вывода: поддержка датчика SDI-12 — подробная информация* *(стр. 267)*.

SDI-12 представляет собой протокол 1200 бод, который поддерживает множество интеллектуальных датчиков. Каждому порту требуется одна клемма и он поддерживает до 16 индивидуально адресуемых датчиков.

- До четырех TTL портов, настроенных на клеммы **C**.

5.3.2.5.5. Порт SDM

SDM — это фирменный протокол Campbell Scientific, который поддерживает различные цифровые датчики, коммуникационные входы, расширения выходов периферийных устройств и некоторые интеллектуальные датчики Campbell Scientific.

- Один порт SDM настроен на клеммы **C1, C2** и **C3**.

5.3.2.5.6. Порт CPI

CPI представляет собой новый фирменный протокол, который поддерживается постоянно расширяющейся линейкой модулей CDM Campbell Scientific. Модули CDM — высокоскоростные периферийные устройства для расширения ввода и вывода. Сеть между совместимыми регистраторами данных Campbell Scientific также можно создавать с помощью портов CPI.

- Для подключения к устройствам CDM требуется периферийный интерфейс CPI, указанный в приложении *Интерфейсы CDM/CP* *(стр. 647)*.

5.3.2.5.7. Порт Ethernet

Читать далее. См. в разделе *TCP/IP* *(стр. 289)*.

- Для Ethernet требуются периферийные интерфейсные устройства Ethernet, указанные в приложении *Список сетевых ссылок* *(стр. 652)*.

5.3.3. Экранная клавиатура — общие сведения

Связанные темы:
- *Экранная клавиатура — общие сведения* *(стр. 83)*
- *Экранная клавиатура — подробная информация* *(стр. 451)*
- *Экранная клавиатура — список* *(стр. 651)*
- *Пользовательские меню — общие сведения* *(стр. 84, стр. 581)*

Экранная клавиатура CR1000KD является удобным инструментом для работы в полевых условиях. Клавиатура CR1000KD, отображенная на рисунке *Экранная клавиатура CR1000KD* *(стр. 83)*, является необязательным периферийным устройством CR1000.

Экранная клавиатура играет важную роль при установке, обслуживании и поиске неисправностей для многих областей применения. Она позволяет проводить опрос и программирование регистратора данных CR1000 независимо от других коммуникационных элементов. Для получения дополнительной информации об использовании экранной клавиатуры см. раздел *Пользовательские меню — общие сведения* *(стр. 84, стр. 581)*. Для получения дополнительной информации о доступных изделиях см. приложение *Список экранных клавиатур* *(стр. 651)*.

Рисунок 29. Экранная клавиатура CR1000KD

5.3.3.1. Набор символов

Доступ к набору символов экранной клавиатуры можно получить одним из трех следующих способов:

- Большинству клавиш соответствуют символы, нанесенные выше клавиш синим цветом. Для ввода символа нажмите клавишу **Shift** от одного до трех раз, чтобы выбрать позицию размещенного выше символа, а затем нажмите клавишу. Например, чтобы ввести **Y**, нажмите **Shift** три раза, а затем нажмите **PgDn** (Страница вниз).

- Для вставки пробела (**Spc**) или изменения регистра букв (**Cap**) нажмите **Shift** один или два раза, чтобы выбрать нужный регистр, а затем нажмите клавишу **BkSpc** (Стирание назад).
- Чтобы вставить символ, которого нет на клавиатуре, нажмите **Ins** (Вставка), прокрутите вниз до **Character** (Символ), затем нажмите **Enter** (Ввести) и выполните прокрутку вверх, вниз, влево или вправо до нужного символа в списке, а затем нажмите **Enter** (Ввести).

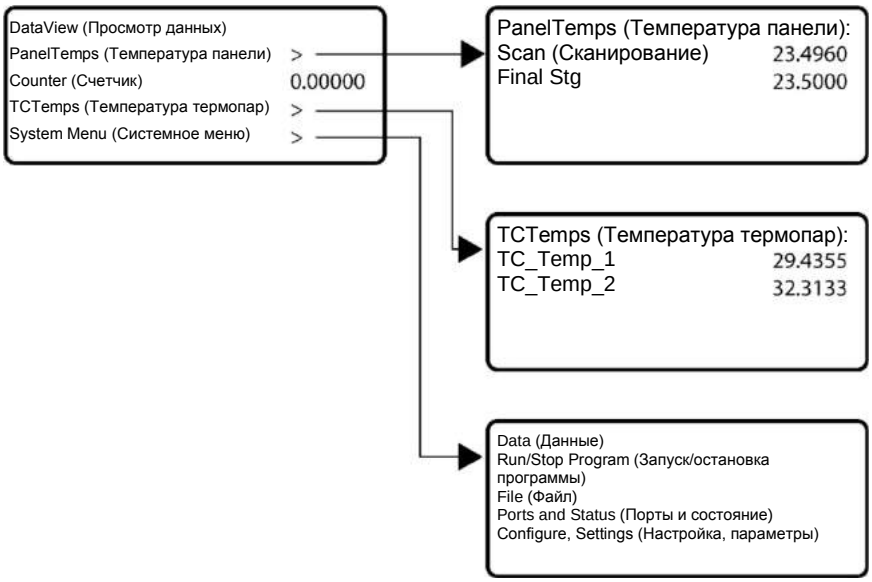
5.3.3.2. Пользовательские меню — общие сведения

Связанные темы:
- *Пользовательские меню — общие сведения (стр. 84, стр. 581)*
- *Экран данных: пользовательские меню — подробная информация (стр. 182)*
- *Пользовательские меню — набор команд (стр. 581)*
- *Экранная клавиатура — общие сведения (стр. 83)*
- *Справка к редактору CRBasic* для **DisplayMenu().**

Программирование CRBasic в CR1000 облегчает создание пользовательских меню для отображения клавиатуры CR1000KD.

На рисунке *Пример пользовательского меню (стр. 84)* показаны окна простого пользовательского меню **DataView (Просмотр данных).** Меню **DataView** (Просмотр данных) отображается в главном меню экранной клавиатуры. В **DataView** (Просмотр данных) есть пункт меню **Counter** (Счетчик), подменю **PanelTemps (Температура панели), TCTemps (Температура термопар)** и **System Menu (Системное меню). Counter** (Счетчик) позволяет выбрать одно из четырех значений. В каждом подменю отображается два значения из памяти CR1000. **PanelTemps** (Температура панели) отображает температуру коммутационной панели CR1000 при каждом сканировании, а также информацию о температуре панели за последнюю минуту. **TCTemps** (Температура термопар) отображает два значения температуры термопар. Для получения дополнительной информации о создании пользовательских меню см. раздел *Отображение данных: пользовательские меню — подробная информация (стр. 182)*.

Рисунок 30. Пример пользовательского меню



5.3.4. Внешнее оборудование для измерения и контроля — общие сведения

Связанные темы:
- *Внешнее оборудование для измерения и контроля — общие сведения (стр. 85)*
- *Внешнее оборудование для измерения и контроля — подробная информация (стр. 366)*
- *Внешнее оборудование для измерения и контроля — списки (стр. 645)*

Campbell Scientific предлагает модули для расширения количества клемм для CR1000. Сюда входит следующее:

Мультиплексоры

Мультиплексоры увеличивают входную мощность клемм, настроенных для аналогового ввода, а также выходную мощность клемм возбуждения Vx.

Устройства SDM

Устройство с последовательным интерфейсом для расширения мощности ввода и вывода CR1000. Эти устройства подключаются к CR1000 через клеммы **C1, C2** и **C3**.

Устройства CDM

Модули Campbell входят в постоянно расширяемую линейку измерительных и контрольных модулей, которые используют высокоскоростную технологию шины CAN интерфейса периферийных устройств (CPI). Они соединяются с помощью интерфейса SC-CPI.

5.3.5. Источники питания — общие сведения

Связанные темы:
- Источники питания — технические характеристики
- *Источники питания — краткое руководство (стр. 44)*
- *Источники питания — общие сведения (стр. 85)*
- *Источники питания — общие сведения (стр. 100)*
- *Источники питания — продукты (стр. 657)*
- *Источники питания (стр. 101)*
- *Диагностика — источники питания (стр. 494)*

CR1000 запитан от источника с номинальным напряжением 12 В пост. тока. Допустимый диапазон составляет от 9,6 до 16 В пост. тока.

Внешнее питание подключается через зеленый соединитель **POWER IN** на лицевой панели CR1000. Положительный вывод питания подключается к клемме **12V**. Отрицательный вывод — к клемме **G**. Соединение имеет внутреннюю защиту от обратной полярности.

CR1000 внутренне защищен от случайного изменения полярности питания.

У CR1000 низкие требования к входной мощности. Например, в составе систем малой мощности он может в течение нескольких месяцев работать от батареи без подзарядки. Источники питания для долгосрочного размещения на удаленных участках, как правило, состоят из источника зарядки, контроллера зарядки и аккумулятора. При наличии линии переменного тока можно использовать адаптер для преобразования переменного тока в постоянный или переменный ток, регулятор заряда периферийного устройства и перезарядный аккумулятор, чтобы создать ИБП (источник бесперебойного питания).

5.3.6. Конфигурация CR1000 — общие сведения

Связанные темы:
- *Конфигурация CR1000 — общие сведения* (стр. 86)
- *Конфигурация CR1000 — подробная информация* (стр. 111)
- *Состояние, настройки и информация о таблицах данных (Status/Settings/DTI)* (стр. 603)

CR1000 поставляется заводом-изготовителем с установленной операционной системой (ОС). Настройки по умолчанию выбраны для тех областей применения, где необходима связь с ПК по **RS-232** и получение и исполнение пользовательских прикладных программ. При более сложных условиях может потребоваться корректировка некоторых настроек. Настройки можно изменить следующим образом:

- *DevConfig (служебная программа конфигурации устройства).* См. раздел *Служебная программа конфигурации устройства* (стр. 111).
- Экранная клавиатура CR1000KD. См. раздел *Экранная клавиатура — подробная информация* (стр. 451) и приложение *Экранная клавиатура — список* (стр. 651).
- ПО поддержки регистратора данных. См. раздел *ПО поддержки регистратора данных — общие сведения* (стр. 95).

Файлы ОС передаются в CR1000 через DevConfig или с помощью кнопки **Отправить** в ПО поддержки регистратора данных. Когда операционная система отправляется через *DevConfig*, большинство настроек аннулируется, а при отправке с помощью ПО поддержки регистратора данных большинство настроек сохраняется. Операционные системы также можно передавать в CR1000 с устройства памяти большой емкости или карты памяти Campbell Scientific.

Обновления ОС периодически выкладываются на сайте *www.campbellsci.com*. При отключении и повторном включении питания ОС и настройки не меняются.

5.3.7. Программирование CRBasic — общие сведения

Связанные темы:
- *Программирование CRBasic — общие сведения* (стр. 86)
- *Программирование CRBasic — подробная информация* (стр. 122)
- *Программирование CRBasic — команды* (стр. 537)
- *Библиотека программных ресурсов* (стр. 169)
- *Справка для редактора CRBasic*

Программа CRBasic определяет, как и когда датчики CR1000 должны проводить измерения, расчеты и сохранять данные. Программа создается на компьютере и передается в CR1000. В памяти CR1000 может храниться множество программ, но при этом активной является только одна программа. Для создания программ для CR1000 используются два программных приложения Campbell Scientific: *Short Cut* и *CRBasic Editor*.

- *Short Cut* позволяет за четыре простых этапа создать программу регистратора и коммутационною схему. Она поддерживает большинство датчиков Campbell Scientific, и ее рекомендуется использовать для создания простых программ для измерительных датчиков и хранения данных.
- Программы, созданные с помощью *Short Cut*, легко импортируются в *CRBasic Editor* для дополнительного редактирования. Для сложных приложений опытные программисты часто готовят основную часть программы, описывающую проведение измерений и хранение данных с помощью *Short Cut*, а затем добавляют более сложный код с помощью *CRBasic Editor*.

Примечание. Когда программа, созданная в *Short Cut*, была отредактирована в *CRBasic Editor* *(стр. 125)*, ее больше нельзя редактировать в программе *Short Cut*.

5.3.8. Память — общие сведения

Связанные темы:

- *Память — общие сведения* *(стр. 87)*
- *Память — подробная информация* *(стр. 370)*
- *Устройства хранения данных — список* *(стр. 653)*

Данные о памяти R1000 размещаются в таблице **Состояние** *(стр. 603)*. Память организована следующим образом:

- Флеш-память ОС:
  - 2 Мб;
  - операционная система (ОС);
  - серийный номер и код платы;
  - код загрузки;
  - стирается при загрузке новой операционной системы (код загрузки стирается только в случае его изменения).
- Последовательная флеш-память:
  - 512 КБ;
  - настройки устройства;
  - защита от записи;
  - энергонезависимая память;
  - ЦП: размещение диска:
    - выделяется автоматически;
    - файловая система FAT;
    - ограниченные циклов записи (100 000);
    - невысокая скорость (последовательный доступ).
- Основная память:
  - 4 Мб статической памяти RAM (SRAM);
  - резервная батарея;
  - переменные ОС;
  - программа, скомпилированная в CRBasic, с двоичной структурой (490 KB максимум);
  - переменные CRBasic;
  - память данных;
  - коммуникационная память;
  - USR: диск:
    - назначается пользователем;
    - диск RAM FAT32;
    - графические изображения (см. приложение «Камеры»);
    - команды файлов данных **TableFile()** (TOA5, TOB1, CSIXML и CSIJSON);
  - *хранение* *(стр. 519)* памяти (переменные ОС не инициализированы);
  - динамическое распределение памяти при выполнении процедур.

Примечание. CR1000 с серийными номерами меньше 11832 обычно поставляются только с 2 Мб статической памяти RAM (SRAM).

Объем памяти для хранения данных может быть увеличен добавлением карты *CF (стр. 510)* и модуля хранения CF (подключается к **периферийному** порту), а также запоминающего устройства (флеш-накопитель), которое подключается к **CS B/B** или обоим. Для получения информации о возможностях для расширения памяти см. приложение *Устройства хранения данных — список (стр. 653)*.

По умолчанию хранение данных (пространство для сохранения данных) организовано в циклическом режиме. После завершения цикла, когда память заполнена, старые данные перезаписываются новыми. Однако команда **DataTable()** позволяет установить таблицу данных ***заполнения и остановки записи.***

5.3.9. Передача и сбор данных — общие сведения

Связанные темы:
- *Сбор и обмен данными — краткое руководство (стр. 45)*
- *Сбор и обмен данными — общие сведения (стр. 88)*
- *Сбор и обмен данными — подробная информация (стр. 391)*
- *Периферийное оборудование для сбора и обмена данными — списки (стр. 651)*

Окончательные данные записываются в таблицы памяти для окончательных данных. Данные можно скопировать на компьютер по коммуникационной линии (*Передача и сбор данных — подробная информация (стр. 391)*), или записав их на карту CompactFlash® (CF) (CRD: диск) или запоминающее устройство большой емкости Campbell Scientific (USB: диск), а потом перенеся на компьютер.

5.3.9.1. Связь PakBus® — общие сведения

Связанные темы:
- *Связь PakBus® — общие сведения (стр. 88)*
- *Связь PakBus® — подробная информация (стр. 393)*
- *Связь PakBus® — команды (стр. 584)*
- *Руководство по работе в сети PakBus* (доступно на сайте *www.campbellsci.com/manuals* (*http://www.campbellsci.com/manuals*)).

CR1000 поддерживает связь с *ПО поддержки регистратора данных (стр. 654)*, *коммуникационными периферийными устройствами (стр. 651)* и другими *регистраторами (стр. 645)*, оснащенными PakBus, узкоспециализированным протоколом сетевой связи. Протокол PakBus по своей структуре сходен с межсетевым протоколом Internet. При использовании подписанных пакетов данных PakBus увеличивает число коммуникационных и сетевых опций, доступных CR1000. Связь может осуществляться по протоколу TCP/IP через порт **RS-232**, **CS B/B** и клеммы **C**.

Преимущества PakBus:

- Одновременная связь между CR1000 и другими устройствами.
- Одноранговая связь — ПК не требуется. Специальные команды CRBasic упрощают передачу данных между регистраторами для распределенного принятия решений и контроля.
- Консолидации данных — другие регистраторы PakBus могут использоваться в качестве «датчиков» для объединения всех данных в один CR1000.
- Маршрутизация — CR1000 может выступать в качестве маршрутизатора, передавая сообщения, предназначенные для другого регистратора Campbell Scientific. PakBus поддерживает автоматическое обнаружение и выбор маршрута.

- Сети на небольшое расстояние — без дополнительных аппаратных средств CR1000 может поддерживать связь с другим CR1000 на расстояние до 30 футов (около 9 метров) путем по проводам для передачи и приема, а также проводам заземления между регистраторами данных.

В сети PakBus каждому регистратору присваивается уникальный адрес. В большинстве устройств PakBus установлен адрес по умолчанию 1. Для связи с CR1000 в ПО поддержки регистратора необходимо указать PakBus адрес CR1000. Адрес PakBus изменяется с помощью *Экранной клавиатуры CR1000KD* *(стр. 451)*, *программы DevConfig* *(стр. 111)*, Таблицы **состояния** CR1000 *(стр. 603)* или программы *PakBus Graph* *(стр. 522)*.

5.3.9.2. Связь

Данные обычно передаются по коммуникационной линии в файл на ПК с помощью *ПО поддержки регистратора данных* *(стр. 654)* *Campbell Scientific*. Описание используемого программного обеспечения см. в руководстве и *Справке*.

5.3.9.3. Устройство памяти большой емкости

Внимание. Отключение устройства хранения данных Campbell Scientific (флеш-накопителя) от CR1000 запрещается выполнять в тот момент, когда индикатор горит или мигает. Извлечение активного устройства может привести к повреждению данных.

Данные, хранящиеся на запоминающем устройстве Campbell Scientific, передаются в CR1000 по коммутационной линии, если устройство подключено к порту **CS B/B**. Кроме того, можно отключить запоминающее устройство, затем подключить его к персональному компьютеру и скопировать файлы с помощью *Проводника Windows*.

5.3.9.4. Карта памяти (CRD: диск) — общие сведения

Связанные темы:
- *Карта памяти (CRD: диск) — общие сведения* *(стр. 89)*
- *Карта памяти (CRD: диск) — подробная информация* *(стр. 376)*
- *Карты памяти и номера записей* *(стр. 466)*
- *Выходные данные: запись высокочастотных данных на карты памяти* *(стр. 205)*
- *Ошибки файловой системы* *(стр. 389)*
- *Устройства хранения данных — список* *(стр. 653)*
- *Примеры форматов файлов данных* *(стр. 379)*
- *Таблица устройств для хранения данных* *(стр. 373)*

Внимание! При использовании карт памяти соблюдайте следующие меры предосторожности:

- Перед установкой карты памяти выключите питание CR1000.
- Перед извлечением карты из слота отключите его, нажав кнопку **Eject**, подождите, пока загорится зеленый свет, а затем выключите питание CR1000.
- Не извлекайте карту памяти, когда привод активен, это может привести к повреждению данных или карты памяти.
- Чтобы избежать потери данных, сохраните их перед отправкой программы с карты памяти на CR1000. При отправке программы с карты памяти на CR1000 часто стираются все данные.

Данные, хранящиеся на карте памяти, передаются на ПК по коммуникационной линии CR1000, также это можно сделать отключением карты и непосредственной передачей данных на компьютера через адаптер стороннего производителя.

Коммуникации

CR1000 получает доступ к данным на карте памяти, когда необходимо заполнить запросы по сбору данных, инициированных командой **Collect** (Сбор) *(стр. 509)* ПО поддержки регистратора данных. Либо, при принятии необходимых мер, сбор данных ведется в двоичной форме. Двоичные данные передаются с помощью команды *Управление файлами | Получение (стр. 515)* в ПО поддержки регистратора данных. Перед началом такого сбора данных остановите выполнение программы CR1000, это гарантирует, что данные не будут повреждены или записаны на карту памяти.

Прямая передача данных с помощью адаптера для ПК

Передача данных с помощью адаптера выполняется быстрее, чем по коммуникационному каналу. Разница в скорости особенно заметна при передаче больших файлов.

Формат файлов данных, полученных на ПК через адаптер отличается от стандартных форматов файлов данных Campbell Scientific. Для получения дополнительной информации см. раздел *Примеры форматов файлов данных (стр. 379)*. Файлы данных можно преобразовать в формат Scientific Campbell с помощью программы *CardConvert (стр. 509)*.

5.3.9.5. Форматы файлов данных в памяти CR1000

При выполнении обычных операций CR1000 хранит данные в двоичных таблицах данных. Однако при использовании команды **TableFile()** данные сохраняются в одном из нескольких форматов в дискретных текстовых файлах, размещенных во внутренней или внешней памяти. Для получения дополнительной информации об использовании команды **TableFile()** см. раздел *Хранение данных — на устройстве (стр. 374)*.

5.3.9.6. Формат данных на компьютере

Данные CR1000, сохраненные на ПК с помощью *ПО поддержки регистратора данных (стр. 654)*, находятся в формате ASCII или в виде двоичного файла в зависимости от типа файла, выбранного в программе. Подробную информацию о доступных форматах файлов данных см. в инструкции к программному обеспечению.

5.3.10. Альтернативная связь — общие сведения

Связанные темы:

- *Альтернативная связь — общие сведения (стр. 90)*
- *Альтернативная связь — подробная информация (стр. 407)*

CR1000 поддерживает связь с внешними устройствами для приема программ, передачи данных или согласования действий в сети. Первичный протокол коммуникации *PakBus (стр. 522)*.Устройство поддерживает и другие коммуникационные протоколы, включая *Web API (стр. 423)*, *Modbus (стр. 411)*, и *DNP3 (стр. 408)*. Для получения полного списка поддерживаемых протоколов см. раздел *Технические характеристики (стр. 97)*. В приложении *Передача и сбор данных — списки периферийных устройств (стр. 651)* перечислены доступные коммуникационные периферийные устройства Campbell Scientific.

Экранная клавиатура также обменивается данными с CR1000. Дополнительный сведения см. в разделе *Экранная клавиатура — общие сведения (стр. 83)*.

5.3.10.1. Modbus

Связанные темы:
- *Modbus — общие сведения* *(стр. 91)*
- *Modbus — подробная информация* *(стр. 411)*

CR1000 поддерживает протокол Modbus, в качества ведущего и ведомого устройства в сетях Modbus SCADA. Modbus — это широко распространенный протокол передачи данных SCADA, который облегчает обмен информацией и данными между компьютером/программным обеспечением HMI, устройствами (RTU) и датчиками, совместимыми с протоколом Modbus. CR1000 обменивается данными с Modbus через порты RS-232, RS-485 (с адаптером RS-232 в RS-485) и TCP.

Системы, основанные на протоколе Modbus, состоят из ведущего устройства (компьютера) и ведомого устройства (RTU/PLC), портативных устройств (датчиков) и оборудования сети передачи данных. Коммуникационный порт, скорость передачи данных, биты данных, стоп-биты и четность устанавливаются драйвером Modbus ведущего и/или ведомых устройства. Стандарт Modbus использует два режима связи: RTU и ASCII. Но CR1000 используется только в режиме RTU.

CR1000 позволяет опрашивать портативные устройства. Поскольку Modbus имеет структуру набора команд, запрограммировать CR1000 на получение данных от датчиков для получения данных от портативных устройств гораздо проще, чем для получения данных от серийных датчиков. Поскольку Modbus использует общую шину и обращается отдельно к каждому узлу, портативные устройства эффективно передают данные в мультиплексном режиме в CR1000 без дополнительного оборудования.

5.3.10.2. DNP3 — общие сведения

Связанные темы:
- *DNP3 — общие сведения* *(стр. 91)*
- *DNP3 — подробная информация* *(стр. 408)*

CR1000 поддерживает протокол DNP3 в качестве ведомого устройства в сетях DNP3 SCADA.

5.3.10.3. TCP/IP — общие сведения

Связанные темы:
- *TCP/IP — общие сведения* *(стр. 91)*
- *TCP/IP — подробная информация* *(стр. 423)*
- *протокол (TCP/IP — команды* *(стр. 593)*
- *ссылки TCP/IP — список* *(стр. 652)*

CR1000 поддерживает следующие протоколы TCP/IP:

- DHCP
- DNS
- FTP
- HTML
- HTTP
- микро-последовательный сервер;
- NTCIP;
- NTP;

- PakBus через TCP/IP;
- Ping;
- POP3;
- SMTP
- SNMP;
- Telnet;
- Web API;
- XML

5.3.11. Безопасность — общие сведения

Связанные темы:
- *Безопасность — общие сведения* (стр. 92)
- *Безопасность — подробная информация* (стр. 467)

CR1000 поставляется без активных средств обеспечения безопасности. По умолчанию протоколы RS-232, Telnet, FTP и HTTP, которые обеспечивают высокий уровень доступа к данным CR1000, и программа CRBasic не защищены паролем.

Вы можете защитить CR1000 от ошибок или подделки данных. Защита может требоваться в следующих случаях:

- сбор конфиденциальных данных;
- использование критически важных систем;
- использование сетей, доступ к которым имеют многие пользователи.

При наличии угроз безопасности, в частности проникновения по протоколу TCP/IP, необходимо загрузить в CR1000 последнюю версию *операционной системы* (стр. 86), отключить неиспользуемые протоколы и защитить те, которые используются. Для обеспечения безопасности можно предпринять следующие действия:

- установка блокировки кода доступа;
- установка пароля PakBus/TCP;
- задание имени пользователя и пароля для FTP;
- установка ключа шифрования PakBus AES-128;
- установка файла формата .csipasswd для защиты HTTP и веб-API;
- подписи записей;
- шифрование файлов программы, если они содержат конфиденциальную информацию;
- для усиления защиты файлы программы можно скрыть;
- установка CR1000 и источник питания в корпус, закрываемый на ключ.

Примечание. Все функции безопасности можно изменить, имея физический доступ к CR1000. Если требуется абсолютная безопасность, CR1000 должен храниться в безопасном месте.

Связанные темы
- *Автоматическая калибровка — общие сведения* (стр. 92)
- *Автоматическая калибровка — подробная информация* (стр. 344)
- *Автоматическая калибровка — ошибки* (стр. 490)
- *Компенсация напряжения смещения* (стр. 323)
- *Заводская калибровка* (стр. 94)
- *Заводская калибровка или процедуры ремонта* (стр. 476)

В CR1000 используется функция автоматической калибровки для компенсации изменений, вызванных приработкой оборудования и изменением рабочей температуры. При отключении автоматической калибровки точность измерения вне рабочего диапазона ухудшается в 10 раз. То есть, в расширенном температурном диапазоне от –40 °C до 85 °C погрешность составляет ±0,12 % от показаний прибора, а в случае отключения автоматической калибровки может снизиться до ±1 %. Но даже если температура CR1000 остается без изменений, при отключении автоматической калибровки присутствует дрейф относительно первоначально снятой градуировочной кривой. Автоматическую калибровку можно отключить, если интенсивность сканирования слишком мала. Ее можно отключить с помощью программы CRBasic командой **Calibrate()**.

Примечание. CR1000 оснащен внутренним эталоном напряжения для калибровки. Там, где требуется высокая точность измерения аналогового напряжения, эталон напряжения должен периодически проверяться и повторно калиброваться Campbell Scientific. Повторную калибровку рекомендуется проводить не реже чем каждые два года.

Если команда **Calibrate()** не задана, CR1000 проводит автоматическую калибровку в свободное время в фоновом режиме, используя *медленную последовательность* *(стр. 157)*, когда калибровка последующего сегмента выполняется через четыре секунды. Если не хватает времени на фоновую калибровку из-за сканирования пользовательской программы, при компиляции CR1000 отобразится следующее предупреждение: **Предупреждение! Фоновая калибровка отключена.**

5.3.12. Обслуживание — общие сведения

Связанные темы:
- *Обслуживание — общие сведения* *(стр. 93)*
- *Обслуживание — подробная информация* *(стр. 473)*

При надлежащем уходе CR1000 будет надежно работать много лет.

5.3.12.1. Защита от влаги — общие сведения

- *Защита от влаги — общие сведения* *(стр. 93)*
- *Защита от влаги — подробная информация* *(стр. 99)*
- *Защита от влаги — изделия* *(стр. 660)*

CR1000 и большинство его периферийных устройств должны быть защищены от влаги. Влага может повредить электронные компоненты и привести к неремонтопригодности CR1000. Вода может попадать внутрь прибора при затоплении или из-за сильных дождей, однако чаще всего источником ее является конденсация. В большинстве случаев, чтобы защитить CR1000 от влаги, достаточно поместить его в защищенный от атмосферных воздействий корпус с абсорбентом, подняв его на достаточное расстояние от земли. CR1000 поставляется с пакетами осушителя, которые снижают влажность внутри корпуса. Пакеты с осушителем, находящиеся внутри корпуса, необходимо периодически менять.

Примечание. При наличии свинцово-кислотных аккумуляторов запрещается полностью герметизировать корпус, поскольку рассеиваемый ими газообразный водород будет скапливаться внутри корпуса и может дойди до взрывоопасной концентрации.

Для получения дополнительной информации о непроницаемых для влаги корпусах см. раздел *Список корпусов* *(стр. 659)*.

5.3.12.2. Защита от перепадов напряжения

Читать далее. См. в разделе *Заземление (стр. 105)*.

CR1000 необходимо заземлить, чтобы свести к минимуму риск поломки из-за перепадов напряжения в сети и напряжения, созданного молнией. Для формирования полной схемы внутренних устройств для фиксации напряжения CR1000 требуется заземление. Для получения информации о доступных устройствах для защиты от перенапряжения см. раздел *Список ограничителей переходного напряжения (стр. 648)*.

5.3.12.3. Заводская калибровка

Связанные темы
- *Автоматическая калибровка — общие сведения (стр. 92)*
- *Автоматическая калибровка — подробная информация (стр. 344)*
- *Автоматическая калибровка — ошибки (стр. 490)*
- *Компенсация напряжения смещения (стр. 323)*
- *Заводская калибровка (стр. 94)*
- *Заводская калибровка или процедуры ремонта (стр. 476)*

CR1000 использует внутренний эталон напряжения для регулярной калибровки. Campbell Scientific рекомендует обновлять заводскую калибровку каждые два года. При необходимости в калибровке см. раздел *Поддержка (стр. 5)* в начальной части данного руководства.

5.3.12.4. Внутренняя батарея — подробная информация

Связанные темы:
- *Внутренняя батарея — краткое руководство (стр. 45)*
- *Внутренняя батарея — общие сведения (стр. 94)*

Предупреждение! Неправильное использование или установка внутренней литиевой батареи может привести к серьезным травмам, пожару, взрыву и тяжелым ожогам. Запрещается перезаряжать, разбирать, нагревать до температуры свыше 100 °C, припаивать к ячейке, сжигать или погружать содержимое в воду. Утилизация отработанных литиевых батарей должна выполняться в соответствии со всеми местными требованиями.

В CR1000 встроена литиевая аккумуляторная батарея, которая питает часы и SRAM, когда CR1000 не подключен к внешнему источнику питания. Если CR1000 хранится при комнатной температуре, срок эксплуатации литиевой батареи составляет около трех лет (меньше при экстремальных температурах). При непрерывной работе CR1000 ячейки литиевой батареи служат гораздо дольше. Внутреннее напряжение литиевой батареи можно контролировать с помощью таблицы **Status** (Состояние) CR1000. Рабочий диапазон батареи составляет приблизительно от 2,7 до 3,6 В постоянного тока. Если напряжение падает ниже 2,7 В пост. тока, замените батарею, как указано в разделе *Замена внутренней батареи (стр. 473)*.

Литиевые батареи не подлежат повторной зарядке. В таких батареях используется хлористый тионил, и эта конструкция является одной из самых безопасных. Максимальный ток разряда ограничен до нескольких мА. Он защищен от чрезмерного тока разрядки от внутренних цепей (прямой канал передачи энергии отсутствует) благодаря резистору 100 Ом. Конструкция имеет ультратонкий дизайн. См.

http://www.tadiran-batterie.de/download/eng/LBR06Eng.pdf.

Батарея рассчитана на работу при температуре от –55 до 85 °C.

5.4. ПО поддержки регистратора данных — общие сведения

Подробнее:
- *ПО поддержки регистратора данных — краткое руководство* *(стр. 46)*
- *ПО поддержки регистратора данных — общие сведения* *(стр. 95)*
- *ПО поддержки регистратора данных — подробная информация* *(стр. 450)*
- *ПО поддержки регистратора данных — списки* *(стр. 654)*

ПО поддержки регистратора данных Campbell Scientific для компьютера с Windows или Linux ускоряет связь между компьютером и CR1000. Существует широкий спектр программного обеспечения, однако большую часть задач можно выполнить с помощью следующих инструментов:

- Мастер программирования *Short Cut* для ОС Windows (SCWin) — *Short Cut* используется для написания простых программ CRBasic, и не требует изучения языка программирования CRBasic. *Short Cut* представляет собой простой в использовании мастер, который поможет вам подготовить программу.
- ПО начального уровня *PC200W* для Windows — поддерживает только прямое последовательное подключение к CR1000 по проводной связи или радиосвязи с передачей сигналов в широком спектре. Он поддерживает передачу программы CRBasic, сбор данных и настройку часов CR1000. *PC200W* можно бесплатно скачать на сайте *www.campbellsci.com/downloads (http://www.campbellsci.com/downloads).*
- *LoggerLink Mobile Apps* — простой инструмент, который позволяет устройству с операционной системой iOS или Android поддерживать связь с CR1000 по протоколу IP. Эта программа поддерживает большую часть функций *PC200W*.
- ПО поддержки регистратора данных *PC400* — включает функции *PC200W*, *редактор CRBasic* и в ручном режиме поддерживает все виды связи (за исключением спутника).
- ПО поддержки регистратора данных *LoggerNet* использует все функции *PC400*, а также в автоматическом режиме поддерживает все виды коммуникаций (за исключением спутника). Оно также включает множество дополнений, например экраны графических данных.

Примечание. Подробную информацию о ПО Campbell Scientific можно найти на сайте *www.campbellsci.com* *http://www.campbellsci.com*. Для получения рекомендации по выбору ПО, подходящего для вашей сферы деятельности, обратитесь к техническому специалисту Campbell Scientific.

6. Технические характеристики

Технические характеристики CR1000, если не указано иное, действительны для температуры от –25 до 50 °C без конденсации. Рекомендуется каждые два года проводить повторную калибровку оборудования. Перед покупкой необходимо получить подтверждение специалиста по применению Campbell Scientific для критически важных характеристик и параметров системы.

ДИАПАЗОН ИСПОЛНЕНИЯ ПРОГРАММЫ

От 10 мс до одного дня с шагом 10 мс

Аналоговые входы (НЕСИМ. 1–16, ДИФ. 1–8)

Восемь дифференциальных (ДИФ.) или 16 несимметричных (НЕСИМ.), индивидуально настраиваемых входных каналов. Расширение канала обеспечивается с помощью дополнительных аналоговых мультиплексоров.

ДИАПАЗОНЫ И РАЗРЕШЕНИЕ. Базовое разрешение, указанное в следующей таблице, — разрешение для единичного преобразования A/D *(стр. 507)*. Для дифференциального измерения с изменением полярности входа используется более высокое разрешение, которое два раза превышает базовое.

Диапазон (мВ)[1]	DIFF Разреш. (мкВ)[2]	Погрешность разреш. (мкВ)
±5000	667	1333
±2500	333	667
±250	33,3	66,7
±25	3,33	6,7
±7.5	1,0	2,0
±2.5	0,33	0,67

[1] Граница в ≈ 9 % на всех диапазонах гарантирует, что максимальное напряжение не приведет к выходу за пределы диапазона.
[2] Разрешение для дифференциального измерения с изменением полярности входа.

ПОГРЕШНОСТЬ АНАЛОГОВОГО ВВОДА [3]:
±(0,06 % от показаний + смещение [3]), от 0 до 40 °C
±(0,12 % от показаний + смещение [3]), от –25 до 50 °C
±(0,18 % от показаний + смещение [3]), от –55 до 85 °C (только -XT)

[3] Погрешность не учитывает помехи датчика и помехи при измерении.

Определения смещения:
Смещение = 1,5 × основ. разреш. + 1,0 мкВ (для диф. измерения с изменением полярности входа)
Смещение = 3 × основ. разреш. + 2,0 мкВ (для диф. измерения без изменения полярности входа)
Смещение = 3 × основ. разреш. + 3,0 мкВ (для несинхронного измерения)

СКОРОСТЬ АНАЛОГОВОГО ИЗМЕРЕНИЯ.

Интеграция типовых кодов	Время интеграции	Время регулирования	---Общее время[4]--- Несим. без изм. поляр.	Диф. с изм. поляр.
250	250 мкс	450 мкс	≈1 мс	≈12 мс
_60 Гц[5]	16,67 мс	3 мс	≈20 мс	≈40 мс
_50 Гц[5]	20,00 мс	3 мс	≈25 мс	≈50 мс

[4] Включает 250 мкс для преобразования в единицы измерения.
[5] Фильтр помех на линии переменного тока.

ВХОДНОЙ ШУМ НАПРЯЖЕНИЯ. Для дифференциальных измерений с изменением полярности входа — ±2,5 мВ от входного диапазона (цифровое разрешение доминирует для более высоких диапазонов):
Интеграция 250 мкс: 0,34 мкВ
Интеграция 50/60 Гц: 0,19 мкВ

ВХОДНЫЕ ПРЕДЕЛЫ: ±5 В пост. тока

ПОДАВЛЕНИЕ СИНФАЗНОГО СИГНАЛА ПОСТ. ТОКА: >100 дБ

ОСЛАБЛЕНИЕ СИНФАЗНОГО СИГНАЛА: 70 дБ при 60 Гц при использовании ослабления 60 Гц

ДИАПАЗОН ВХОДНОГО НАПРЯЖЕНИЯ БЕЗ НАРУШЕНИЙ ИЗМЕРЕНИЯ: макс. ±8,6 мА

НАПРЯЖЕНИЕ ДЛИТЕЛЬНОГО ВХОДНОГО СИГНАЛА БЕЗ ПОВРЕЖДЕНИЯ: макс. ±16 В пост. тока

ВХОДНОЙ ТОК: обычно ±1 нА, макс. ±6 нА при 50 °C; ±90 нА при 85 °C

ВХОДНОЕ СОПРОТИВЛЕНИЕ: обычно 20 ГОм

ПОГРЕШНОСТЬ ВСТРОЕННОГО ЭТАЛОНА СОЕДИНЯЮЩЕГО ТЕРМИСТОРА (для измерений с помощью термопары):
±0,3 °C, от –25 до 50 °C
±0,8 °C, от –55 до 85 °C (только —XT)

АНАЛОГОВЫЕ ВЫВОДЫ (VX 1–3)

Три выхода с переключаемым напряжением последовательно включаются только во время измерения.

ДИАПАЗОНЫ И РАЗРЕШЕНИЕ

Канал	Диапазон	Разрешение	Источник тока / потребление энергии
(VX 1–3)	±2,5 В пост. тока	0,67 мВ	±25 мА

ПОГРЕШНОСТЬ АНАЛОГОВОГО ВЫХОДА (VX):
±(0,06 % от показаний + 0,8 мВ, от 0 до 40 °C
±(0,12 % от показаний + 0,8 мВ, от –25 до 50 °C
±(0,18 % от показаний + 0,8 мВ, от –55 до 85 °C (только -XT)

ФУНКЦИЯ РАЗВЕРТКИ ПО ЧАСТОТЕ VX: импульсные выходы обеспечивают программируемое смещенной частоты прямоугольного импульса от 0 до 2500 мВ для возбуждения струнных датчиков.

СРЕДНЕЕ ЗНАЧЕНИЕ ЗА ПЕРИОД

Любой из 16 несим. аналоговых входов можно использовать для получения среднего значения за период. Точность ± (0,01 % от показания + разрешение), где разрешение составляет 136 нс, разделенных на заданное число циклов измерения.

ВХОДНАЯ АМПЛИТУДА И ЧАСТОТА

Коэффициент усиления напряжения	Диапазон кода	Наибольший из пиковых входных сигналов Мин. мВ[6]	Макс. В[7]	Мин. длительность импульса мкс	Макс. частота кГц[8]
1	mV250	500	10	2,5	200
10	mV25	10	2	10	50
33	mV7_5	5	2	62	8
100	mV2_5	2	2	100	5

[6] Сигнал центрируется около *порогового значения* (см. описание команды **PeriodAvg()**).
[7] Сигнал центрируется относительно заземления.
[8] Максимальная частота = 1 / (двойная минимальная длительность импульса) для 50 % сигналов рабочего цикла.

ЛОГОМЕТРИЧЕСКИЕ ИЗМЕРЕНИЯ

ВИДЫ ИЗМЕРЕНИЙ: CR1000 поддерживает логометрические измерения сопротивления с помощью напряжения возбуждения. При выхода с переключаемым напряжением возбуждения позволяют проводить измерения четырех- и шестиконтактных полных мостов, а также двух-, трех- и четырехконтактных полумостов. Дополнительное изменение полярности возбуждения минимизирует ошибки при измерении постоянного тока.

ПОГРЕШНОСТЬ ЛОГОМЕТРИЧЕСКИХ ИЗМЕРЕНИЙ [9,11]
Примечание. Сноска 9 содержит два важных допущения:
± (0,04 % от измеряемого напряжения + смещение [12])

[9] Характеристика погрешности предполагает изменение полярности возбуждение для напряжения возбуждения < 10001000 мВ. Допущение не учитывает ошибки резистора моста, а также помехи датчика и помехи при измерении.

[11] Расчетная погрешность ΔX (где X — значение, получаемое при измерении, где **мультипликатор** = 1, **смещение** = 0):
Команда BRHalf(): $\Delta X = \Delta V_1/V_X$.
Команда BRFull(): $\Delta X = 1000 \times \Delta V_1/V_X$, выражать в виде мВ•В$^{-1}$.
Примечание. ΔV_1 рассчитывается с учетом логометрической погрешности измерения. Для получения дополнительной информации см. раздел *Измерение сопротивления* *(стр. 337)*.

[12] Определение смещения:
Смещение = 1,5 × основ. разреш. + 1,0 мкВ (для диф. измерения с изменением полярности входа)
Смещение = 3 × основ. разреш. + 2,0 мкВ (для диф. измерения без изменения полярности входа)
Смещение = 3 × основ. разреш. + 3,0 мкВ (для несинхронного измерения)
Примечание. Изменение полярности возбуждения в два раза сокращает смещение.

СЧЕТЧИКИ ИМПУЛЬСОВ (P 1–2)

Два входа специально выбираются для замыкания переключателя, высокочастотного импульса или переменного тока низкого уровня. Независимые 24-разрядные счетчики для каждого входа.

МАКСИМАЛЬНЫЙ ИМПУЛЬС НА СКАНИРОВАНИЕ: 16,7 × 10^6

РЕЖИМ ПЕРЕКЛЮЧЕНИЙ-ЗАМЫКАНИЙ:
Минимальное время замыкания переключателя: 5 мс
Минимальное время размыкания переключателя: 6 мс
Макс. время размыкания: 1 мс открытия без подсчета

ВЫСОКОЧАСТОТНЫЙ ИМПУЛЬСНЫЙ РЕЖИМ:
Максимальная входная частота: 250 кГц
Максимальное входное напряжение: ±20 В
Пороговое напряжение: Подсчет ведется при переходе от ниже 0,9 В до выше 2,2 В после входного фильтра с постоянной времени 1,2 мкс.

РЕЖИМ НИЗКОУРОВНЕВОГО ПЕРЕМЕННОГО ТОКА.
Внутренняя связь по переменному току меняет смещение постоянного тока до ±0,5 В.
Гистерезис на входе: 12 мВ среднекв. при 1 Гц
Максимальное входное переменное напряжение: ±20 В
Минимальное входное переменное напряжение:

Синусоидальные колебания (мВ среднекв.)	Диапазон (Гц)
20	от 1,0 до 20
200	от 0,5 до 200
2000	от 0,3 до 10 000
5000	от 0,3 до 20 000

ЦИФРОВЫЕ ПОРТЫ ВВОДА/ВЫВОДА (C 1–8)

Восемь портов задаются программно, в качества двоичных входов или управляющих выходов. Они обеспечивают включение/выключение, широтно-импульсную модуляцию, время крайних значений, прерывание/запуск подпрограмм, подсчет импульсов переключений-замыканий, подсчет высокочастотных импульсов, асинхронную связь (UART) и связь по SDI-12.

Также поддерживается связь по SDM.

ЦИФРОВЫЕ ПОРТЫ ВВОДА/ВЫВОДА (C 1–8)

Восемь портов задаются программно, в качества двоичных входов или управляющих выходов. Они обеспечивают включение/выключение, широтно-импульсную модуляцию, время крайних значений, прерывание/запуск подпрограмм, подсчет импульсов переключений-замыканий, подсчет высокочастотных импульсов, асинхронную связь (UART) и связь по SDI-12. Также поддерживается связь по SDM.

МАКС. НИЗКОЧАСТОТНЫЙ РЕЖИМ: < 1 кГц
МАКС. ВЫСОКОЧАСТОТНЫЙ РЕЖИМ: 400 кГц
МАКС. ЧАСТОТА ЗАМЫКАНИЯ КЛЮЧА: 150 Гц
РАЗРЕШАЮЩАЯ СПОСОБНОСТЬ ПО ВРЕМЕНИ: 540 нс
Выходные напряжения (без нагрузки): высокое 5,0 В ± 0,1 В; низкое < 0,1 В
ВЫХОДНОЕ СОПРОТИВЛЕНИЕ: 330 Ом
СОСТОЯНИЕ ВХОДА: высокое от 3,8 до 16 В; низкое от -8,0 до 1,2 В
ВХОДНОЙ ГИСТЕРЕЗИС: 1,4 В
ВХОДНОЕ СОПРОТИВЛЕНИЕ:
100 кОм с входами < 6,2 В пост. тока
220 Ом с входами < 6,2 В пост. тока
СЕРИЙНОЕ УСТРОЙСТВО / ПОДДЕРЖКА RS-232: от 0 до 5 В пост. тока УАПП

ПЕРЕКЛЮЧАЕМЫЕ 12 В ПОСТ. ТОКА (SW-12)

Одна независимая нерегулируемая клемма 12 В пост. тока включается и выключается под управлением программы. Плавкий предохранитель держит ток = 900 мА при 20 °C, 650 мА при 50 °C и 360 мА при 85 °C.

СООТВЕТСТВИЕ ЕВРОПЕЙСКИМ СТАНДАРТАМ CE

СТАНДАРТЫ, КОТОРЫМ СООТВЕТСТВУЕТ ИЗДЕЛИЕ: IEC61326:2002

КОММУНИКАЦИОННЫЙ ИНТЕРФЕЙС

ПОРТЫ RS-232:
Девятиконтактный DCE: (электрически не изолирован) для подключения компьютеров или модемов, производства сторонних компаний.
От COM1 до COM4: четыре независимых пары Tx/Rx на управляющих портах (неизолированных); от 0 до 5 В пост. тока UART
Скорость передачи данных: может быть выбрана от 300 бит/с до 115,2 кбит/с.
Формат по умолчанию: восемь бит данных; один стоп-бит; без контроля четности.
Дополнительные форматы: семь бит данных; два стоп-бита; нечетный, контроль четности.
ПОРТ CS В/В: интерфейс с коммуникационными периферийными устройствами Campbell Scientific.
SDI-12: цифровые порты управления C1, C3, C5, C7 настраиваются отдельно и соответствуют стандарту SDI-12 вер. 1.3 для режима регистратора. Каждый порт поддерживает до десяти датчиков SDI-12.
ПЕРИФЕРИЙНЫЙ ПОРТ: 40-контактный интерфейс для подключения периферийных устройств CompactFlash или Ethernet.
ПОДДЕРЖИВАЕМЫЕ ПРОТОКОЛЫ: PakBus, шифрованный AES-128 PakBus, Modbus, DNP3, FTP, HTTP, XML, HTML, POP3, SMTP, Telnet, NTCIP, NTP, веб-API, SDI-12, SDM.

СИСТЕМА

ПРОЦЕССОР: Renesas H8S 2322 (16-разрядный процессор с 32-битовым внутренним ядром 7,3 МГц)
ПАМЯТЬ: 2 Мб флеш-памяти для операционной системы; 4 Мб с статической памяти ОЗУ (SRAM) с аварийным питанием от батареи для ЦП, программы CRBasic и данных.
ПОГРЕШНОСТЬ ЧАСОВ РЕАЛЬНОГО ВРЕМЕНИ: ±3 мин. в год. Коррекция с помощью GPS предоставляется по запросу.
РАЗРЕШЕНИЕ ЧАСОВ РЕАЛЬНОГО ВРЕМЕНИ ЧАСЫ: 10 мс

ТРЕБОВАНИЯ К СИСТЕМЕ ЭЛЕКТРОПИТАНИЯ

НАПРЯЖЕНИЕ: от 9,6 до 16 В пост. тока
ВНУТРЕННЯЯ БАТАРЕЯ: 1200 мА·ч литиевая батарея для часов и резервная копия SRAM. Как правило, она обеспечивает резервное питание в течение трех лет.

ВНЕШНЯЯ БАТАРЕЯ: Дополнительно доступны щелочные и перезаряжаемые батареи 12 В пост. тока. Подключение к источнику тока защищается от изменения полярности тока.
ОБЫЧНЫЙ РАСХОД ТОКА ПРИ 12 В пост. тока:
Режим сна: обычно 0,7 мА; макс. 0,9 мА
Частота дискретизации 1 Гц (одно быстрое несим. измерение) мА
Частота дискретизации 100 Гц (одно быстрое несим. измерение): 16 мА
Частота дискретизации 100 Гц (одно быстрое несим. измерение с RS 232): 28 мА
Активная внешняя экранная клавиатура добавляет 7 мА (100 мА с подсветкой).

ФИЗИЧЕСКИЕ

РАЗМЕРЫ: 239 × 102 × 61 мм (9,4 × 4,0 × 2,4 дюйма); дополнительное свободное пространство для кабелей и проводов.
ВЕС: 1,0 кг / 2,1 фунта

ГАРАНТИЯ

Предоставляемая гарантия указана в опубликованном прайс-листе, а также на первых страницах данного и других руководств пользователя.

7. Установка

Подробнее:
- *Краткое руководство* (стр. 41)
- *Технические характеристики* (стр. 97)
- *Установка* (стр. 99)
- *Эксплуатация* (стр. 303)

7.1. Защита от влаги — подробная информация

Защита от влаги — общие сведения (стр. 93)
Защита от влаги — подробная информация (стр. 99)
Защита от влаги — изделия (стр. 660)

Когда уровень влажности достигает точки росы, появляется конденсат, который может повредить CR1000. Поэтому пользователь должен обеспечить эффективный контроль влажности прибора.

Модуль CR1000 защищен от влаги пакетом силикагеля, который вкладывается на заводе-изготовителе. При каждом ремонте CR1000 в Campbell этот пакет заменяется. В обычной ситуации модуль открыть не следует, кроме замены внутренней литиевой батареи.

Для обеспечения дополнительной защиты внутрь корпуса необходимо поместить специальный осушитель.

7.2. Диапазон рабочих температур

CR1000 может надежно работать при температуре от –40 до 75 °C (дополнительно от –55 до 85 °C) без образования конденсата.

7.3. Защитные корпуса

Защитные корпуса — подробная информация (стр. 99)
Защитные корпуса — продукты (стр. 659)

На рисунке *Защитный корпус* (стр. 100) показан обычный защитный корпус Campbell Scientific для размещения CR1000. Данный корпус относится к типу NEMA 4X (водостойкий, пыленепроницаемый, устойчивый к коррозии, для установки внутри и вне помещений). В корпусе есть крепежная пластина, на которую устанавливается CR1000 и подключенные к нему периферийные устройства. В крепежной пластине сделаны отверстия диаметром один дюйм с анкерными нейлоновыми вставками. В основании CR1000 есть крепежные отверстия (в некоторых моделях в эти отверстия уже вставлены резиновые вставки), через которые небольшие винты вкручиваются в специальные нейлоновые фиксаторы. Чтобы получить доступ к крепежным отверстиям, **удалите резиновые вставки**, если они установлены. Винты и нейлоновые фиксаторы входят в комплект поставки защитного корпуса.

Рисунок 31. Защитный корпус

7.4. Источники питания — подробная информация

Связанные темы:

- Источники питания — технические характеристики
- *Источники питания — краткое руководство* *(стр. 44)*
- *Источники питания — общие сведения* *(стр. 85)*
- *Источники питания — общие сведения* *(стр. 100)*
- *Источники питания — продукты* *(стр. 657)*
- *Источники питания* *(стр. 101)*
- *Диагностика — источники питания* *(стр. 494)*

Надежная подача электроэнергии является основой надежной работы системы сбора данных. При проектировании электроснабжения необходимо учитывать максимальную потребность в энергии и экстремальные условия окружающей среды. Например, потребляемая мощность метеорологической станции может быть существенно выше при низкой температуре, одновременно очень низкая температура ограничивает мощность источника питания.

CR1000 внутренне защищен от случайного изменения полярности питания.

У CR1000 низкие требования к входной мощности. Например, в составе систем малой мощности он может в течение нескольких месяцев работать от батареи без подзарядки. Источники питания для долгосрочного размещения на удаленных участках, как правило, состоят из источника зарядки, контроллера зарядки и аккумулятора. При наличии линии переменного тока адаптер для преобразования переменного тока в постоянный или переменный ток, можно использовать регулятор заряда периферийного устройства и перезарядный аккумулятор, чтобы создать ИБП (источник бесперебойного питания).

Если вам требуется помощь в выборе источника питания, например для экстремальных условий эксплуатации, обратитесь к специалистам Scientific Campbell.

7.4.1. Потребность CR1000 в электроэнергии

Для работы CR1000 требуется питание напряжением от 9,6 до 16 В пост. тока, подаваемое на клеммы **POWER IN** зеленого соединителя на лицевой части коммутационной панели.

CR1000 внутренне защищен от случайного изменения полярности питания. Диод ограничителя бросков напряжения на клемме **ВХОД ПИТАНИЯ 12 В** обеспечивает защиту от периодических всплесков напряжения, ограничивая его в пределах от 19 до 21 В. Длительное входное напряжение более 19 В может повредить диод ограничителя бросков напряжения.

Внимание! Уровни напряжения клемм **12 В**, переключаемых клемм **SW12**, а также 8-контактного порта **CS B/B**, тесно связаны с уровнями напряжения основного источника питания. Например, если ток подается по клеммам **ВХОД ПИТАНИЯ 12 В** и **G** 16 В пост. тока, то клеммы **12 В** и **SW12**, а также 8-контактный порт **CS B/B** будут подавать 16 В пост. тока на подключенное периферийное устройство. Если подключенное периферийное устройство или датчик не рассчитаны на такое напряжение, это может привести к поломке.

7.4.2. Расчетная потребляемая мощность

Читать далее. См. в разделе *Источники питания — общие сведения (стр. 85)*.

Время работы системы от батарей можно определить, разделив емкость батарей (в ампер-часах) на среднее потребление тока (в амперах). CR1000 обычно имеет утечку тока покоя 0,5 мА (при выключенном дисплее), 0,6 мА при частоте выборке 1 Гц и > 10 мА при частоте сканирования 100 Гц. При включенной экранной клавиатуре потребление тока возрастает на дополнительные 7 мА, а включение подсветки добавляет еще 100 мА.

7.4.3. Источники питания

Связанные темы:

- Источники питания — технические характеристики
- *Источники питания — краткое руководство (стр. 44)*
- *Источники питания — общие сведения (стр. 85)*
- *Источники питания — общие сведения (стр. 100)*
- *Источники питания — продукты (стр. 657)*
- *Источники питания (стр. 101)*
- *Диагностика — источники питания (стр. 494)*

Обратите внимание, что некоторые преобразователи переменного тока в постоянный создают коммутационные помехи или пульсацию *переменного тока (стр. 507)*. Чрезмерные коммутационные помехи на выходе питания могут привести к увеличению помех при измерении, ухудшая его точность. Помехи электросети также могут передаваться через трансформатор или возникать из-за влияния расположенных рядом электродвигателей, обогревателей и линий электропередач.

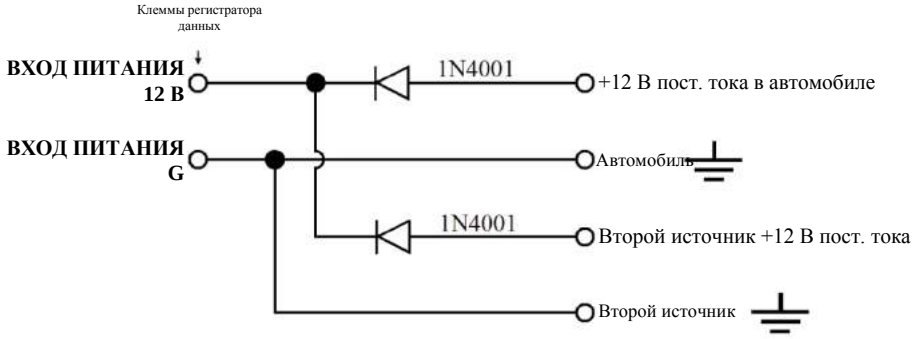
Высококачественные регуляторы мощности обычно снижают помехи благодаря регулированию мощности. Использование дополнительных аргументов подавления для 50 или 60 Гц при измерении аналогового входа для команды CRBasic (см. раздел *Поддержка датчика* *(стр. 303)*) часто улучшает подавление шумов из сети электропитания. Для оценки помех при измерении можно использовать команду стандартного отклонения CRBasic **SDEV().**

Электропитание CR1000 обеспечивается внешним источником питания.

7.4.3.1. Подключение питания от автомобиля

Если CR1000 подключен к источнику питания легкового автомобиля, может потребоваться второй источник питания. При запуске двигателя автомобиля напряжение батареи часто падает ниже уровня, необходимого для работы регистратора данных. Это может привести к остановке измерений CR1000, пока напряжение не превысит нижний допустимый уровень. Чтобы предотвратить возникновение ошибок во время запуска транспортного средства, необходимо предусмотреть дополнительный источник питания. На схеме *Подключение CR1000 к источнику питания автомобиля* *(стр. 102)* показано подключение второго источника питания для CR1000. Диод ИЛИ соединение вызывают подачу энергии к CR1000 с наибольшим напряжением и предотвращают использование резервного источника питания при попытках завести автомобиль.

Рисунок 32. Подключение к источнику питания автомобиля



7.4.4. Источник бесперебойного питания (ИБП)

Если используется внешняя щелочная батарея, она подключается непосредственно к **ВХОДУ ПИТАНИЯ 12 В** и клеммам **G** (от 9,6 до 16 В пост. тока).

Чаще всего лучшим вариантом при установке оборудования на длительный срок является использование ИБП (источник бесперебойного питания). Внешний ИБП состоит из основного источника, внешнего регулятор зарядки CR1000 и внешней батареи. Первичный источник питания, в качестве которого часто выступает трансформатор, силовой преобразователь или солнечные панели, подключается к регулятору зарядки, который подает номинальное напряжение 12 В пост. тока на аккумуляторную батарею. Третий контакт соединяет регулятор зарядки и клеммы **12 В** и **G** разъема **ВХОД ПИТАНИЯ**.

7.4.5. Установка внешнего источника питания

При подключении внешнего источника питания к CR1000 отключите зеленый разъем **ВХОД ПИТАНИЯ** на лицевой панели CR1000. Подключите положительный провод в зеленый разъем, затем подключите отрицательный провод. Переустановите разъем зеленого цвета в CR1000. CR1000 имеет внутреннюю защищу от изменения полярности внешнего питания. Если это произойдет измените подключение проводов.

7.5. Переключаемое выходное напряжение — подробная информация

Связанные темы:

- Переключаемое выходное напряжение — технические характеристики
- *Переключаемое выходное напряжение — общие сведения* *(стр. 78)*
- *Переключаемое выходное напряжение — подробная информация* *(стр. 103)*
- *Управление ПЛК — общие сведения* *(стр. 74)*
- *Управление ПЛК — подробная информация* *(стр. 244)*
- *Модули управления ПЛК — общие сведения* *(стр. 368)*
- *Модули управления ПЛК — списки* *(стр. 648)*
- *Контроль ПЛК — команды* *(стр. 562)*

Коммутационная панель CR1000 представляет собой удобное устройство для энергоснабжения датчиков и периферийных устройств, которым требуется источник питания 5 или 12 В пост. тока. Она имеет две непрерывные соединительные клеммы 12 В пост. тока (**12 В**), одну переключаемую клемму с программным управлением 12 В пост. тока (**SW12**) и одну непрерывную клемму 5 В пост. тока (**5 В**). Клеммы внутреннего ограничения тока **SW12, 12 В** и **5 В** используются для защиты от случайных коротких замыканий. Напряжение на клеммах **12 В** и **SW12** может меняться в широких пределах и колеблется в зависимости от источника постоянного тока CR1000, поэтому будьте внимательны, источник питания регистратора данных должен соответствовать характеристикам датчиков. Клемма **5 В** внутренне регулируется в пределах ± 4 %, это хороший показатель для источника энергии, однако обычно не подходит для мостового датчика возбуждения. В таблице *Лимиты подачи тока* *(стр. 103)* указаны текущие ограничения для клемм **12 В** и **5 В**. При превышении ограничений по току на этих клеммах может возникнуть пониженное выходное напряжение. Для получения дополнительной информации см. раздел *Клеммы, настроенные на управление* *(стр. 368)***.**

Таблица 5. Источник тока и ограничения при снижении

Клемма	***Предел*** [1]
VX или **EX** (напряжение возбуждения) [2]	макс. ±25 мА
SW-12[3]	< 900 мА при 20 °C < 630 мА при 50 °C < 450 мА при 70 °C
12 В + SW-12 (комбинированный)[4]	< 3,00 А при 20 °C < 2,34 А при 50 °C < 1,80 А при 70 °C < 1,50 А при 85 °C
5 В + CS Б/В (комбинированный)[5]	< 200 мА

Таблица 5. Источник тока и ограничения при снижении

Клемма	*Предел* [1]

[1] «Источник» — положительный ток; «минус» — отрицательный ток (–).

[2] Превышение ограничения тока приводит к неустойчивости выходного напряжения. Напряжение стабилизируется после возврата установленного уровня тока.

[3] Для ограничения мощности используется многоконтактный предохранитель. Результат перегрузки от перепада напряжения. Для сброса отключите цепь и дайте остыть. Работа с пороговым током допустима, пока колебания не выходят за допустимый диапазон.

[4] Многоконтактный предохранитель защищен. См. сноску 3.

[5] Ток ограничен цепью ограничения тока, которая поддерживает максимальный ток, сбрасывая напряжение при слишком большой нагрузке.

7.5.1. Переключаемое напряжение возбуждения

Три переключаемые клеммы с аналоговым выходом (возбуждение) (от **VX1** до **VX3**) под управлением программы обеспечивают ±2500 мВ постоянного тока возбуждения. Данные о погрешности клемм возбуждения можно проверить в разделе *Спецификации CR1000* *(стр. 97)* и узнать об их ограничениях. Технические характеристики указаны для нагрузок, не превышающих ±25 мА.

Читать далее. Таблица *Источник тока и ограничения при снижении* *(стр. 103)* содержит дополнительную информацию об уровне перегрузки возбуждения.

Команды CRBasic, которые контролируют напряжение возбуждения, включают:

- **BrFull**()
- **BrFull6W**()
- **BrHalf**()
- **BrHalf3W**()
- **BrHalf4W()**
- **ExciteV**()

Примечание. Возбуждение прямоугольной волны переменного тока для использования с поляризованными мостовыми датчиками настраивается с помощью параметра ***RevEx*** команд моста.

7.5.2. Непрерывно регулируемый (клемма 5 В)

Клемма **5 В** регулируется и остается на уровне 5 В пост. тока (± 4 %), пока напряжение питающей сети CR1000 остается выше 9,6 В пост. тока. Он предназначен для питания датчиков или устройств, требующих питания 5 В постоянного тока. Он не предназначен для использования в качестве источника возбуждения для мостовых измерений. Однако измерение вывода клеммы **5 В** благодаря перемычке на аналоговый вход на том же CR1000 способствует точному измерению по мостовой схеме, если необходимо использовать клемму **5 В**.

Примечание. Таблица *Источник тока и ограничения при снижении* *(стр. 103)* содержит дополнительную информацию об уровне перегрузки возбуждения.

7.5.3. Непрерывное нерегулируемое напряжение (клемма 12 В)

Используйте клеммы **12 В** для постоянного питания устройств, требующих 12 В пост. тока. Напряжение на клеммах **12 В** меняется питающим напряжением CR1000.

Внимание! Уровни напряжения клемм **12 В**, переключаемых клемм **SW12 ,** а также 8-контактного порта **CS B/B**, тесно связаны с уровнями напряжения основного источника питания. Например, если ток подается по клеммам **ВХОД ПИТАНИЯ 12 В** и **G** 16 В пост. тока, то клеммы **12 В** и **SW12**, а также 8-контактный порт **CS B/B** будут подавать 16 В пост. тока на подключенное периферийное устройство. Если подключенное периферийное устройство или датчик не рассчитаны на такое напряжение, это может привести к поломке.

7.5.4. Переключаемое нерегулируемое напряжение (клемма SW12)

Клемма **SW12** часто используется для питания устройств, например датчиков, которым во время измерения требуется 12 В пост. тока. При температуре 20 °C ток источника питания должен составлять 900 мА или менее. См. раздел *Источник тока и ограничения при снижении (стр. 103)*. Напряжение на клемме **SW12** будет меняться при изменении напряжения питания источника CR1000. Две команды CRBasic **SW12()** и **PortSet()** контролируют клемму **SW12**. В CR1000 каждая команда обрабатывается по-разному. **SW12 ()** — задача обработки. Используйте ее для управления мощностью SDI-12 и серийных датчиков, которые используют команды **SDI12Recorder()** или **SerialIn()** соответственно. Программирование CRBasic с помощью модели **IF THEN** (ЕСЛИ ТО) используется для управления **SW12**, например, для управления мобильным телефоном следует также использовать команду **SW12().**

PortSet() — команда измерения. Применяется для подключения питания входных аналоговых датчиков, которым требуется электричество непосредственно перед измерением.

Схема коммутации Campbell Scientific 12 В пост. тока предназначен для подключения к клемме **C**. Он включен в перечень в приложении *Драйверы реле — продукты (стр. 649)*.

Примечание. Силовая клемма SW12 нерегулируемая и может подавать 900 мА при температуре 20 °C. См. таблицу *Источник тока и ограничения при снижении (стр. 103)*. Самовосстанавливающийся полимерный предохранитель используется для защиты от перегрузки по току. Сброс осуществляется снятием нагрузки или отключением клеммы **SW12** на несколько секунд.

Клемма **SW12** может по-разному вести себя в *конвейерном (стр. 152)* и *последовательном (стр. 153)* режимах. Для получения дополнительной информации см. разделе *Справка по редактору CRBasic*.

7.6. Заземление

Во всех случаях заземление CR1000 с его периферийными устройствами и датчиками очень важно. Правильное заземление обеспечивает максимальную защиту от ЭСР (электростатического разряда) и точность измерения.

7.6.1. Защита ЭСР

Подробнее:
- *Защита ЭСР (стр. 105)*
- *Грозозащита (стр. 107)*

Существует несколько источников ЭСР (электростатического разряда), наиболее распространенным и разрушительным из которых являются удары молнии. Первичные удары молнии непосредственно поражают CR1000 или датчики. Вторичные удары создают высокое напряжение в электрической линии или проводах датчиков.

Чаще всего для защиты от ЭСР используются газоразрядные трубки. Все критически важные входы и выходы CR1000 защищены с помощью газоразрядных трубок или диодов подавления выбросов напряжения. Газоразрядные трубки срабатывают при напряжении 150 В, позволяя перенаправить ток на клеммы заземления. Для достижения желаемого результата заземляющий наконечник необходимо правильно подключить к заземлению (заземлению на шасси). Как показано на рисунке *Схема заземления* *(стр. 107)*, заземление цепей передачи полезных сигналов и цепей питания использует независимое подключение заземляющей клемме.

Коммуникационные порты представляют собой еще один путь, который может использоваться в переходных режимах. Поэтому каналы связи, например телефонная линия или линия модем ближней связи, должны быть защищены с помощью искрового разрядника. Такая защита является дополнительной опцией, поэтому ее необходимо специально указать при заказе. Искровые разрядники подключаются к клемме заземления, заземлению корпуса или шине заземления (заземлению на массу).

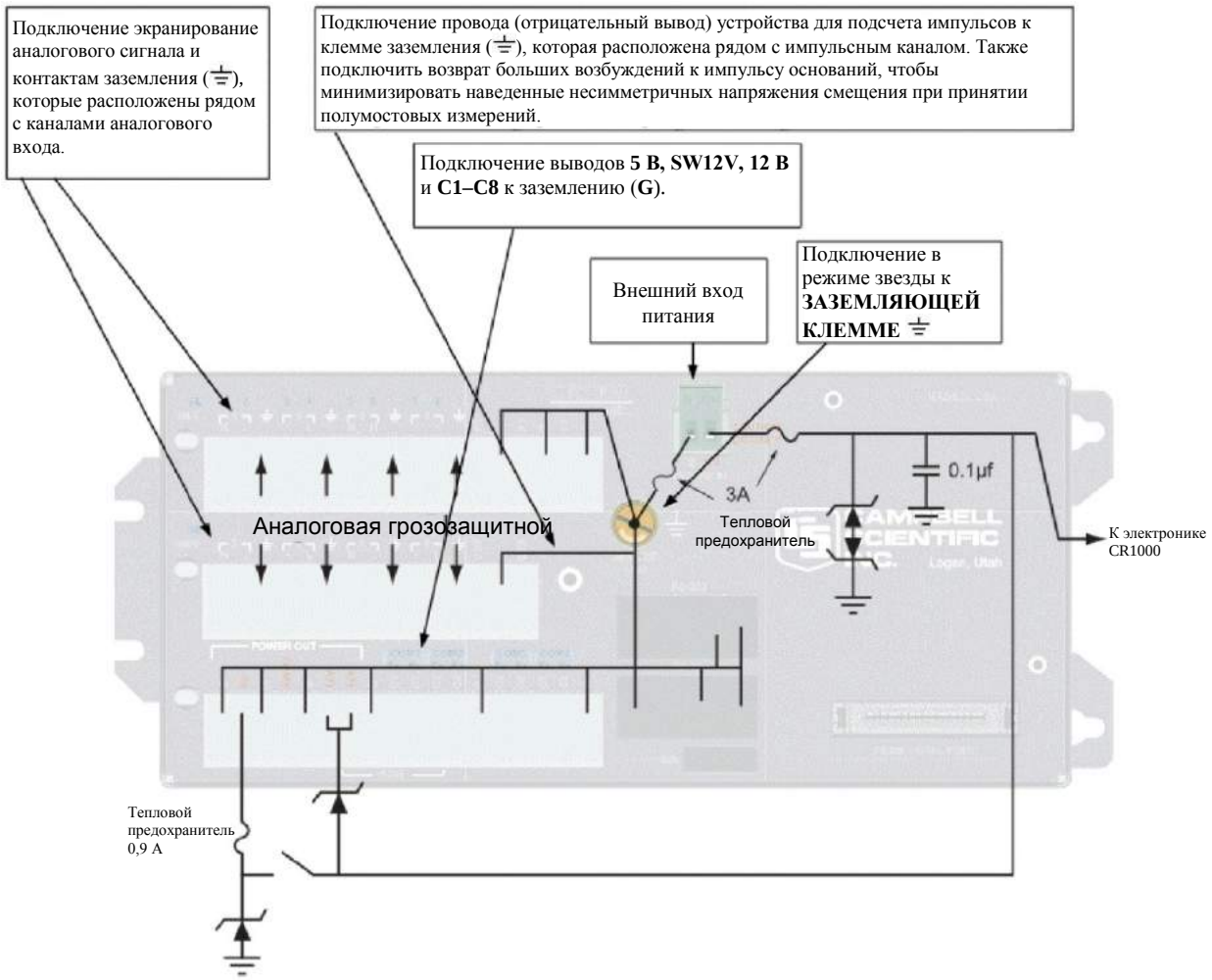
Правильное заземление (заземлению на массу) позволяет свести к минимуму повреждения регистратора и датчиков, создавая канал с низким сопротивлением от системы до точки с низким потенциалом. Campbell Scientific рекомендует заземлять (заземлять на шасси) все регистраторы. Все компоненты системы (регистраторы, датчики, внешние источники питания, опоры, корпуса и т. д.) должны быть подключены к одному заземляющему контакту (шасси).

В полевых условиях правильно заземление должно состоять из вбитого в землю заземляющего стержня длиной 6–8 футов (1,8–4,8 м) с медной оплеткой, подключенного к большой медной клемме заземления на монтажной панели с помощью провода 12 AWG. В условиях почв с низкой проводимостью, например песка, очень сухой почвы, льда или скал, один заземляющий стержень, скорее всего, не сможет обеспечить достаточное заземление. В такой ситуации необходимо ознакомиться с рекомендациями по грозозащите, опубликованными в открытой литературе, или обратиться к квалифицированному консультанту по этому вопросу.

Для автомобиля клемму заземления необходимо надежно закрепить на шасси транспортного средства проводом 12 AWG или больше.

В лабораторных условиях обеспечение стабильного заземления может быть сложной задачей, которую однако все равно необходимо выполнить. В старых зданиях, где новая электрическая проводка подключается к старой, может возникнуть ситуация, когда при наличии заземляющего контакта, розетка все равно не заземлена. Поэтому при наличии заземляющих контактов стоит убедиться в их работоспособности. Если целостность заземления источника питания переменного тока находится под вопросом, дополнительно заземлите систему через водопровод здания или используйте другой проверенный способ заземления.

Рисунок 33. Схема заземления



7.6.1.1. Грозозащита

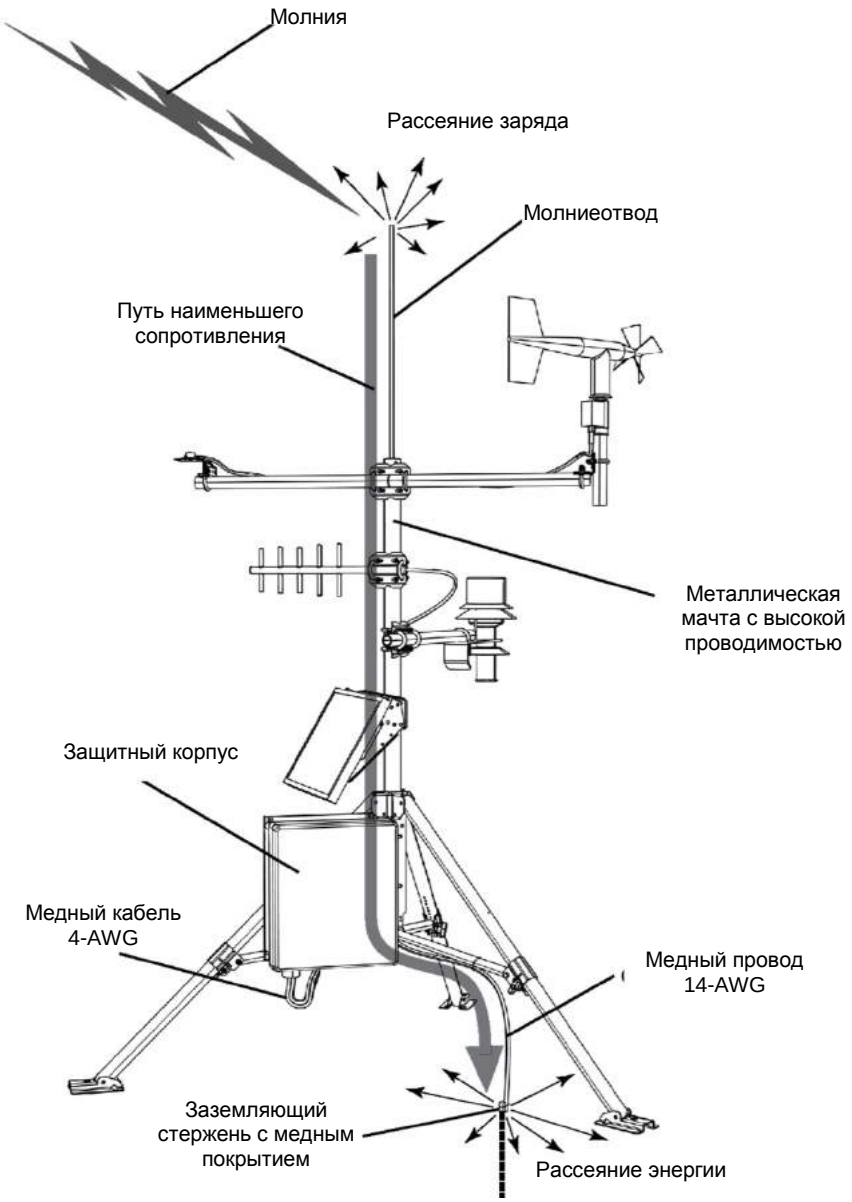
Подробнее:
- *Защита ЭСР (стр. 105)*
- *Грозозащита (стр. 107)*

Наиболее распространенными и разрушительными электростатическими разрядами являются первичные и вторичные удары молний. Прямые удары молнии повреждают приборы непосредственно. Вторичные удары вызывают напряжение в электрической линии или проводов, подключенных к измерительному оборудованию. Хотя на рынке предлагаются сложные, дорогие и почти безошибочные системы грозозащиты, Campbell Scientific, в течение многих лет, использует простую и недорогую конструкцию, которая в большинстве случаев может защитить большинство систем. Система включает громоотвод, металлическую мачту, провод заземления большого сечения и заземляющий стержень, которые отводят от CR1000 ток, который может вызвать повреждения. Однако и эта система не является безошибочной. На рисунке *Схема грозозащиты (стр. 108)* показан пример типичного использования данной системы.

Примечание. Удары молнии могут повредить или разрушить CR1000 и связанные с ними датчики и источники питания.

Кроме описанных средств для адекватной защиты различных установок необходимо использовать простой громоотвод и создать канал низкого сопротивления к клемме заземления.

Рисунок 34. Схема грозозащиты



7.6.2. Несимметричное эталонное измерение

Несимметричные измерения низкого напряжения (< 200 мВ) чувствительны к колебанию потенциала земли из-за изменения обратных токов от клемм **12V, SW12, 5V** и **C1–C8**. Схема заземления CR1000 обеспечивает минимизацию этих колебаний, поскольку в ней отделено заземление сигнальных устройств (⏚) от заземления цепи питания (**G**). Чтобы воспользоваться преимуществами данной конструкции, необходимо соблюдать следующие правила:

- Подключите заземление клемм **12 В, SW12, 5 В** и **C1–C8** к клеммам **G**.
- Подключите заземление ⏚ возбуждения к ближайшей клемме в том же клеммном блоке.
- Подключите нижнюю сторону несимметричных ⏚ датчиков к ближайшему клемме в том же клеммном блоке.
- Подключите экранированные провода ⏚ к клемме, которая находится ближе всего к клеммам, к которым подключены сигнальные провода датчика.

Примечание. Несколько проводов могут быть подключены к одной клемме заземления.

Если возникают проблемы со смещением из-за экранирования или заземления провода с высоким протеканием тока, может помочь подключение проблемных проводов к клеммам ⏚, расположенным рядом с клеммами, настроенными на возбуждение и подсчет импульсов. Проблемные провода также можно подключить непосредственно к заземляющей клемме, чтобы минимизировать наведенное несимметричное смещение напряжения.

7.6.3. Разница потенциала заземления

Следует помнить, что поскольку при несимметричных измерениях CR1000 используется заземление, то любое изменение в электрическом потенциале земли между датчиком и CR1000 приведет к погрешности измерений. Дифференциальные измерения ДОЛЖНЫ использоваться в тех случаях, когда известно, что заземление входа имеет отличие потенциала от заземления CR1000. Для получения дополнительной информации см. раздел *Несимметричные измерения — подробная информация* *(стр. 307)*.

Разность потенциала заземления является общей проблемой при измерении с помощью датчиков полного моста (тензорезисторов, датчиков давления и т. д.), а также при измерении параметров почву с помощью термопар.

7.6.3.1. Термопара для измерения температуры почвы

Если измерительный спай термопары не изолирован во время ее нахождения в почве или в воде, а потенциал заземления, например, на 1 мВ больше на датчике, чем в точке заземления CR1000, измеренное напряжение будет на 1 мВ больше, чем выходное значение термопары. Для медь-константановой термопары 1 мВ погрешность измерения в этом случае составляет приблизительно 25 °C.

7.6.3.2. Внешний преобразователь сигнала

Для проведения измерений и передачи аналоговой информации в CR1000 часто используются внешние инструменты со встроенными преобразователями сигнала, например инфракрасные газоанализаторы. Эти инструменты часто подключены к тому же источнику, что и линейный источник переменного тока CR1000. Несмотря на подключение к одной и той же земле, разница между отдачей тока и сопротивлением проводов приводит к различному потенциалу земли двух измерительных приборов. Поэтому дифференциальное измерение необходимо проводить на аналоговом выходе от внешнего блока формирования сигналов.

7.6.4. Контур заземления при ионных измерениях

При измерении влажности почвы с помощью гипсового блока или коэффициента фильтрации с ячейкой сопротивления существует вероятность ошибки из-за контура заземления. При использовании датчика ионного матричного потенциала (влажность почвы) возникает петля заземления, поскольку почва и вода обеспечивают альтернативный путь возбуждения от заземления CR1000. Этот пример рассмотрен на схеме *Модель контура*

заземления с резистивным датчиком (стр. 110). Благодаря R_g в цепи резистора сигнал, измеренный датчиком, описывается следующим уравнением:

$$V_1 = V_x \frac{R_s}{(R_s + R_f) + R_x R_f / R_g}$$

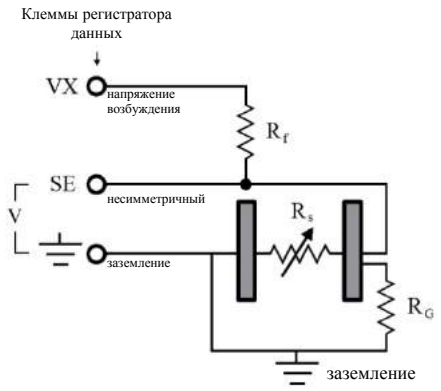
где

V_x – напряжение возбуждения
R_f – постоянный резистор
R_s – сопротивление датчика
R_g – сопротивление между возбужденным электродом и заземлением CR1000

R_xR_f/R_g — источник ошибок, связанных с контуром заземления. Когда R_g велико, ошибка незначительна. Обратите внимание, что геометрия электродов оказывает значительное влияние на величину этой ошибки. Гипсовые блоки Delmhorst, которые используются в зондах Campbell Scientific 227, имеют два соосных цилиндрических электрода. Центральный электрод используется для возбуждения; поскольку он окружен заземляющим электродом, длина контура с замыканием через почву значительно сокращается. Блоки для измерения влаги, которые состоят из двух электродов с параллельными пластинами, особенно восприимчивы к проблемам в цепи обратной связи через землю. Аналогичные соображения применимы к геометрии электродов в датчиках для измерения коэффициента фильтрации.

Заземляющий электрод от датчика фильтрации воды или влажности почвы и заземление CR1000 с гальваническим элементом, а также с раствором воды/почвы, который выступает в роли электролита. Если ток может протекать, то со временем окисление или восстановление приводит к повреждению электрода, если возбуждение постоянного тока используется для проведения измерений. В резистивных почвенных датчиках и датчиках проводимости Campbell Scientific используются конденсаторы, которые блокируют этот постоянный ток. Конденсаторы не только не позволяют ухудшаться чувствительности датчиков, но и блокируют любую составляющую постоянного тока, которая влияет данные измерения.

Рисунок 35. Модель контура заземления с резистивным датчиком



7.7. Конфигурация CR1000 — подробная информация

Связанные темы:
- *Конфигурация CR1000 — общие сведения* (стр. 86)
- *Конфигурация CR1000 — подробная информация* (стр. 111)
- *Состояние, настройки и информация о таблицах данных (Status/Settings/DTI)* (стр. 603)

Поставляемые CR1000 уже настроены на связь с *ПО поддержки регистратора данных* (стр. 95) Campbell Scientific через порт **RS-232**, а также по большинству коммуникационных каналов. Если более старый CR1000 не поддерживает связь по этим простым каналам, проведите полный сброс параметров устройства, как описано в разделе *Сброс CR1000* (стр. 381). Для некоторых приложений, особенно тех, которые реализуют функции TCP/IP, может потребоваться изменить заводские настройки по умолчанию.

Настройка (команда) предполагает действия, которые модифицируют прошивку или программное обеспечение CR1000. Большинство из этих действий связано с настройками регистров CR1000. В данном документе программа CRBasic, с помощью которой можно настраивать CR1000, описана в отдельном разделе (*Программирование CRBasic — подробная информация* (стр. 122)).

7.7.1. Средства конфигурирования

Средства конфигурирования включают:

- *Служебная программа конфигурации устройства* (стр. 111)
- *Планировщик сети* (стр. 112)
- *Параметры состояния/настроек/таблиц* (стр. 114)
- *Программа CRBasic* (стр. 115)
- *Исполняемые файлы CPU* (стр. 115)
- *Экранная клавиатура* (стр. 462)
- Эмулятор терминала

7.7.1.1. Конфигурация с DevConfig

Наиболее универсальным инструментом конфигурации является *Служебная программа конфигурации устройства* или *DevConfig*. Она поставляется с *LoggerNet, PC400, RTDAQ*, также ее можно загрузить с *www.campbellsci.com/downloads (http://www.campbellsci.com/downloads).* Основные характеристики:

- Обширная контекстно-зависимая справка
- Подключается непосредственно к CR1000 через последовательный порт или IP-соединение
- Облегчает доступ к большинству настроек, регистров состояния и регистров информационных таблиц данных
- Включает эмулятор терминала, который облегчает доступ к командной строке CR1000

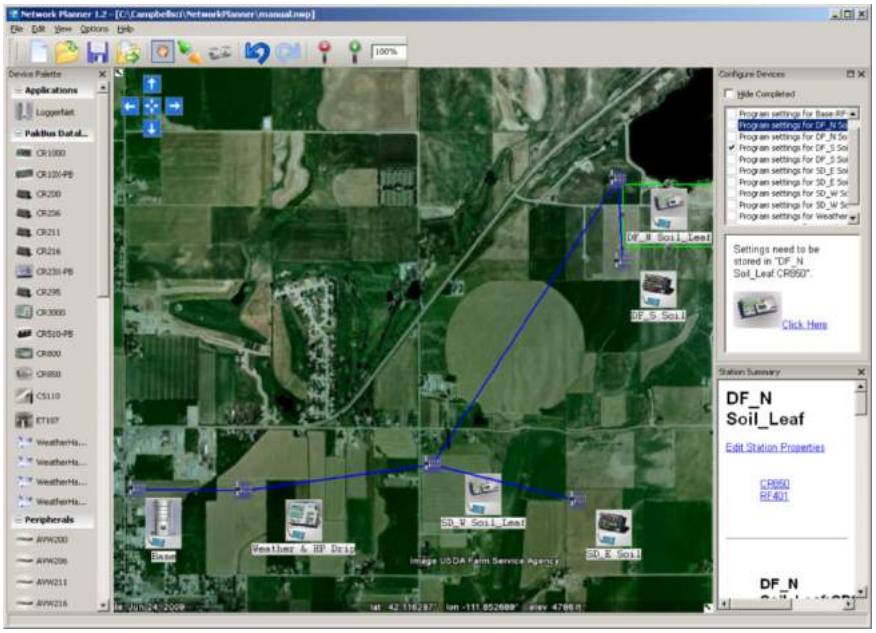
Справка DevConfig позволяет пользователям разобраться в особенностях подключения и использования прибора. Самый простой вариант представляет собой прокладку последовательного кабеля от COM или USB порта компьютера к порту **RS-232** на CR1000, как показано на рисунке *Силовые и последовательные коммуникационные соединения* (стр. 48). Обновления *DevConfig* доступны на сайте *www.campbellsci.com/downloads (http://www.campbellsci.com/downloads).*

Рисунок 36. Служебная программа конфигурации устройства (DevConfig)

7.7.1.2. Планировщик сети

Планировщик сети представляет собой приложение, работающее по принципу перетаскивания, которое используется при проектировании сетей PakBus регистратора данных. Взаимодействовать с *Планировщиком сети* можно через схему, на которой размещаются узлы регистратора данных и ПК. На схеме также указаны коммуникационные каналы между узлами и их взаимодействие. *Планировщик сети* автоматически определяет параметры отдельных устройств и создает настройки XML-файлов для загрузки каждого устройства через *DevConfig* *(стр. 111)*.

Рисунок 37. Установка планировщика сети



7.7.1.2.1. Общие сведения

Планировщик сети позволяет:

- Создавать графическое представление сети, как показано на рисунке *Установка планировщика сети* *(стр. 113)*.
- Задавать параметры для устройств и *LoggerNet*.
- Программно задавать новые параметры устройствам и *LoggerNet*.

Зачем необходим *Планировщик сети*?

- Протокол PakBus позволяет разрабатывать сложные сети.
- Индивидуальная настройка устройств представляет собой достаточно сложную задачу.
- Позволяет применять параметры сразу по всей сети.
- При наличии различных типов устройств требуется согласование параметров.

Пояснения

- *Планировщик сети* облегчает, но не заменяет процесс проектирования.
- Он применяется только для PakBus сетей.
- Не дает рекомендации о выборе оборудования.
- Не создает программы для регистратора данных.
- Не анализирует расстояния или топографию, то есть не предупреждает о превышении дальности передачи или наличии препятствий на пути передачи сигнала.

Для получения дополнительной информации о *Планировщике сети* см. руководство для *LoggerNet*, размещенное по адресу *www.campbellsci.com*.

7.7.1.2.2. Основы

Настройка PakBus

- Адреса устройств пристраиваются автоматически, но их можно изменить вручную.
- Подключения устройств используются для определения того, что следует указать в перечне соседних устройств.
- Поверочный интервал зависит от взаимодействия устройств.
- Сигнальные интервалы можно назначить, но им также можно присвоить значения по умолчанию.
- Сетевая функция (например, маршрутизатор или конечный узел) назначается на основании связей между устройствами.

Связь между устройствами и коммуникационные ресурсы

- Аннулирование нерабочих связей
- Сравнение приемлемости связей
- Предотвращение чрезмерного выделения ресурсов
- Оптимальная скорость передачи данных через RS-232 и CS I/O на основе связи между устройствами
- Определение оптимального размера пакета на основании предполагаемого маршрута

Принципы использования планировщика сети

- Добавление фона (дополнительно)
- Размещение станции, периферийных устройств и т. д.
- Установка связей
- Настройка действий (опрос по расписанию, обратный вызов)
- Настройка устройств
- Настройка *LoggerNet* (добавляет запланированную сеть в карту сети *LoggerNet* **Network Map**)

7.7.1.3. Конфигурация на основе параметров состояния/настроек/таблиц

Связанные темы:
- *Состояние, настройки и информация о таблицах данных (Status/Settings/DTI)* *(стр. 603)*
- *Общее назначение таблицы состояния* *(стр. 604)*
- *Таблица состояния, как средство отладки* *(стр. 485)*

Таблица **Status** (Состояние), настройки CR1000 и информационная таблица **DataTableInfo** (Информация о таблицах данных), вместе называемые **Status/Settings/DTI** (состояние/настройки/таблицы), содержат регистры, настройки и информацию, относящуюся к установке, программированию и отладке многих расширенных систем CR1000. Параметров состояния/настроек/таблиц достаточно много. Следует отметить:

- Все параметры состояния/настроек/таблиц, за исключением некоторых, доступны с помощью ключевых слов. Настоящее описание посвящено этим ключевым словам. Ключевые слова и описания приведены в алфавитном порядке в разделе *Описание параметров состояния/настроек/таблиц (в алфавитном порядке)* *(стр. 611)* *настоящего приложения.*
- Поля состояния только считываются (чаще всего). Некоторые из них могут сбрасываться.
- Параметры чтения/записи (чаще всего).
- Информация о таблицах данных только для чтения.
- Каталоги в приложении *Каталоги параметров состояния/настроек/таблиц* *(стр. 604)* содержат список нескольких групп ключевых слов. Каждому ключевому слову, указанному в этих

группах, соответствует описание.

- Некоторые параметры состояния/настроек/таблиц имеют несколько имен в зависимости от интерфейса для доступа к ним.
- Ни один интерфейс не имеет доступа ко всем параметрам состояния/настроек/таблиц. Интерфейсы доступа включают:

Таблица 6. Параметры состояния/настроек/таблиц: узлы доступа

Узел доступа	*Расположен в...*
Редактор настроек	*Служебная программа конфигурации устройства, экран Connect (соединение) программного обеспечения LoggerNet, программное обеспечение PakBus Graph. См. раздел ПО поддержки регистратора данных — подробная информация (стр. 450).*
Состояние	Просмотр информации в виде таблицы данных на *цифровом мониторе (стр. 521).*
DataTableInfo	Просмотр информации в виде таблицы данных на *цифровом мониторе (стр. 521).*
Состояние устройства	Пункт меню *ПО поддержки администратора регистратора данных (стр. 654).*
Изменение настроек	Пункт меню ПО *PakBusGraph.*
Настройки	Пункт меню экранной клавиатуры CR1000KD **Configure, Settings (Настройка, параметры)**
status.keyword/settings.keyword	Синтаксис программы CRBasic

[1] Информация, представленная в поле **Station Status** (Состояние устройства) автоматически не обновляется. Для того, чтобы обновить данные, нажмите кнопку **Refresh** (Обновить).

Примечание. При создании таблиц **Status** (Состояние) и **DataTableInfo** (Информация о таблицах данных) используются мощности и пропускная способность процессора. Если мощности CR1000 загружены обработкой, например, при выполнении длинных или сложных операций, постоянный вызов таблицы **Status** (Состояние) может привести к *пропускам сканирования (стр. 487).*

Status*603/Settings/DTI (Параметры состояния/настроек/таблиц) (стр. 603)* можно установить или получить с помощью команд CRBasic **SetStatus()** или **SetSetting().**

Например, чтобы присвоить параметру **StationName** (Имя устройства) значение **BlackIceCouloir**, используется следующий синтаксис:

```
SetSetting("StationName","BlackIceCouloir")
```

где ***StationName*** (Имя устройства) — ключевое слово, а ***BlackIceCouloir*** — установленное значение.

Настройки можно запросить через программу CRBasic, используя следующий синтаксис:

```
x = Status.[setting]
```

где ***Setting*** (Настройки) — ключевое слово для настройки.

Например, чтобы получить нужное значение параметра **StationName** (Имя устройства), используйте следующее выражение:

```
x = Status.StationName
```

7.7.1.4. Конфигурация с исполняемыми файлами ЦП

Многие параметры CR1000 можно изменить дистанционно по коммуникационному связи, как непосредственно, так и с помощью процедур программы CRBasic, описанных в разделе *Конфигурирование с помощью программы CRBasic (стр. 115)*. Однако использование удаленных инструментов может привести к изменению настроек и отключению связи. В этом случае, скорее всего, придется выезжать непосредственно на место, чтобы исправить проблему, если хотя бы одно из положений, обсуждаемых в данном разделе, не выполняется. Например, модемы для беспроводных

локальных (сотовых) сетей часто контролируются коммутируемой клеммой 12 В пост. тока (**SW12**). **SW12** обычно отключена, поэтому, если отключена программа управления **SW12**, например, с помощью замены программой, которая не реагирует на управление **SW12**, ячейка модема отключается, и удаленный CR1000 выпадает из системы коммуникаций.

Исполняемые файлы ЦП автоматически исполняются в соответствии с графиком, указанным в таблице. Каждый из них может содержать код для установки определенных параметров CR1000.

Исполняемые файлы ЦП включают:

- *«Включаемый» файл* *(стр. 147)*
- *Файл «Default.crl»* *(стр. 116)*
- *Файл «Powerup.ini»* *(стр. 386)*

Чтобы начать использование, необходимо создать каждый файл, а затем поместить на диск ЦП CR1000. Файлы «включаемый» и «default.cr1» содержат код CRBasic. «Powerup.ini» использует другой, ограниченный язык программирования.

7.7.1.4.1. Файл «Default.cr1»

Файл «default.cr1» можно сохранить на диск ЦП CR1000. При включении CR1000 активирует «default.cr1», если другой программе не присвоен более высокий приоритет (см. раздел *Приоритеты запуска исполняемого файла* *(стр. 116)*). Файл «default.cr1» можно отредактировать, чтобы сохранить важные параметры регистратора данных, например, параметры связи, но изменить при этом можно лишь несколько строк кода.

Для загрузки операционной системы по коммуникационному каналу требуется больше свободной памяти CR1000. Если предполагается загрузить операционную систему по коммуникационной линии и сохранить файл «default.cr1» в CR1000, программа «default.cr1» не должна занимать значительный объем памяти, что может произойти при выделения большого диска USR. Также не рекомендуется использовать набор команд **DataTable()** (Таблица данных) для автоматического выделения памяти. Для получения информации об передаче операционной системы см. раздел *Обновление операционной системы (ОС)* *(стр. 117)***.**

Исполнение файла «default.cr1» при включении питания можно прервать, удерживая клавишу **DEL** на экранной клавиатуре CR1000KD.

CRBasic: пример 1. Простой файл «Default.crl» для управляющей клеммы SW12

```
'В этом примере программа демонстрирует использование файла «Default.cr1». Он должен содержать
'только нескольких строк кода. Эта программа контролирует коммутируемую силовую клемму SW12,
'которая может быть полезной, чтобы по умолчанию поддерживать ВКЛЮЧЕННЫМ питание удаленного
'модема.

BeginProg
  Scan(1,Sec,0,0)
    If TimeIntoInterva1(15,60,Sec) Then SW12(1)
    If TimeIntoInterva1(45,60,Sec) Then SW12(0)
  NextScan
EndProg
```

7.7.1.4.2. Приоритеты запуска исполняемого файла

1. При включении CR1000 выполняет команды из файла «powerup.ini» (находящегося на запоминающем устройстве или карте памяти Campbell Scientific, включая команды установки атрибутов программного файла CRBasic **Run Now** (Запустить сейчас) или **Run on Power-up (Выполнить при включении электропитания).**
2. При включении CR1000 файл программы с маркировкой **Run on Power-up** (Выполнить при включении электропитания) будет использоваться в качестве текущей программы. Иначе будет использоваться любой файл с пометкой **Run Now** (Выполнить сейчас).

3. Если существует файл, указанный в параметре **Include File Name** (Имя включаемого файла), он компилируется в конце программы, выбранной на этом этапе.
4. Если нет файла, выбранного на этапе 1, или если выбранный файл не может быть скомпилирован, CR1000 попробует запустить программу, указанную в параметре **Include File Name** (Имя включаемого файла). CR1000 позволяет оператору **SlowSequence** (Показать последовательность) занять место оператора **BeginProg** (Начало программы). Это позволяет «включаемому файлу» выполнять роль программы по умолчанию.
5. Если программу, которая указана в параметре **Include File Name** (Имя включаемого файла), не удается запустить или если не указана ни одна программа, CR1000 попробует запустить программу «default.cr1» с диска ЦП.
6. При отсутствии файла «default.cr1» или если его не удается скомпилировать, CR1000 не будет автоматически запускать программу.

7.7.2. Конфигурация CR1000 — подробная информация

Далее приведено несколько действий общей конфигурации:

- *Обновление операционной системы* *(стр. 117)*
- Доступ к *регистру* CR1000 *(стр. 114)* для устранения неисправностей
- Установка часов CR1000
- Сохранение текущей конфигурации
- Восстановление конфигурации

Инструменты, доступные для выполнения этих действий, указаны в следующей таблице:

Таблица 7. Общие действия и инструменты для конфигурации

Действия	***Используемые инструменты***[1]
Обновление операционной системы	*Программа DevConfig* *(стр. 111)**, отправка программы* *(стр. 524)**, карта памяти* *(стр. 89)*, запоминающее устройство.
Доступ к регистру	DevConfig, PakBus Graph, программа CRBasic, *«включаемый» файл* *(стр. 147)**, файл «Default.crl»* *(стр. 116)*.
Установка часов CR1000	DevConfig, PC200W, PC400, LoggerNet
Сохранение/восстановление конфигурации	DevConfig
[1] Инструменты перечислены в порядке предпочтения.	

7.7.2.1. Обновление операционной системы (ОС)

CR1000 поставляется с предварительно загруженной операционной системой. Проверьте предварительно загруженную версию, подключив компьютер к CR1000 согласно процедуре, описанной в *Справке DevConfig*. Версия операционной системы указана:

Вкладка Deployment (Развертывание),

Вкладка Datalogger (Регистратор данных),

Текстовое поле OS Version (Версия ОС).

Обновите ОС CR1000, как указано в Справке DevConfig. Текущая версия ОС размещена на сайте www.campbellsci.com/downloads. Обновления ОС проводится бесплатно.

Примечание. Файл ОС имеет расширение .obj. Его можно сжать с помощью алгоритма сжатия gzip. Регистратор данных примет и распакует файл при получении. См. приложение *Программа и сжатие ОС* *(стр. 463)*.

Обратите внимание на следующие меры предосторожности:

- Поскольку отправка ОС сбрасывает память CR1000, это безусловно приведет к потере данных. Кроме того, в зависимости от некоторых факторов CR1000 может на определенный промежуток времени перестать работать.
  - Необходима ли отправка ОС для устранения критической проблемы? Если нет, то, возможно, следует подождать следующего планового посещения для технического обслуживания.
  - Насколько легко можно получить доступ к оборудованию для решения проблем или сброса параметров, требуются ли для этого дополнительные расходы?
  - Если необходимо отправить ОС, а непосредственный доступ к оборудованию затруднен или требует чрезмерных затрат, попробуйте загрузить ОС на CR1000 с аналогичной программной, который размещен более удобно.
- Campbell Scientific рекомендует для обновления операционной системы использовать только прямую проводную связь. Хотя кнопка **Send Program** (Отправка программы) *(стр. 524)* в *ПО поддержки регистратора данных* *(стр. 654)* позволяет отправить ОС по всем видам связи, которые поддерживаются программным обеспечением.
  - Файлы операционной системы имеют очень большой объем — **это может повлиять на плату за пользование линией.**
  - Обновление ОС может привести к сбросу настроек CR1000, включая настройки, необходимые для поддержки линий связи. В более новых операционных системах такой риск сведен к минимуму.

Примечание. Начиная с 25 версии файлы ОС имеют достаточно большой размер, поэтому при некоторых обстоятельствах CR1000 с серийным номером ≤ 11831, имеющим только 2 Мб SRAM, может не хватить объема памяти. В случае возникновения проблем с CR1000, имеющим 2 Мб памяти, отправка файла по прямому последовательному соединению, как правило, решает проблему.

Операционная система обновляется в с помощью одного из следующих инструментов:

7.7.2.1.1. Обновление ОС с помощью вкладки DevConfig для отправки ОС

Данный метод позволяет восстановить на CR1000 заводские значения по умолчанию. При этом текущая ОС, если она была получена, перезаписывается. Неспособность получить и установить совершенно новую ОС переводит CR1000 в нестабильное состояние. Используйте этот метод только при прямом последовательном кабельном подключении.

Процедура выполнения

Использование следующей процедуры с *DevConfig*: не **подключайте** ПО к CR1000.

1. Выберите CR1000 из списка устройств слева.
2. Выберите соответствующий порт связи и скорость передачи данных в левом нижнем углу.
3. Щелкните вкладку **Send OS** (Отправить ОС) в верхней части окна

DevConfig.

4. Следуйте экранным инструкциям **по загрузке ОС.**

Преимущества и недостатки

Хороший способ восстановления CR1000, который оказался в неуправляемом состоянии. Часто операционную систему можно загрузить, даже если CR1000 не поддерживает другие коммуникационные каналы.

Такая загрузка операционной системы позволит:
1. Вернуть CR1000 к заводским настройкам по умолчанию.
2. Удалить данные их итогового хранилища.
3. Удалить данные с диска USR, а также удалить сам диск.
4. Удалить программные файлы, которые хранятся на регистраторе данных.

7.7.2.1.2. Обновление ОС с помощью DevConfig

Этот метод очень похож на отправку ОС, как программы, за исключением того, что вам необходимо вручную подготовить регистратор данных к получению новой ОС.

Процедура выполнения

1. Подключение к CR1000 с помощью функции *Connect (Подключение) или DevConfig.*
2. Сбор данных.
3. Передача программного файла *default.CRl* *(стр. 116)* на диск ЦП CR1000.
4. Остановка текущей программы и выбор варианта удаления связанных с ней данных (это позволит освободить память SRAM, выделенную для хранения данных).
5. Сбор файлы с диска USR (если это применимо).
6. Удаление файлов с диска USR (если это применимо).
7. Отправка нового файла ОС «.obj» на CR1000.
8. Перезапуск предыдущей программы (файл «default.CR1» будет запущен после компилирования ОС).

Преимущества и недостатки

Этот метод является предпочтительным, поскольку здесь пользователь должен вручную настроить регистратор данных для получения ОС и, следовательно, он осведомлен о происходящем (потеря данных, остановка программы и т. д.).

Такая загрузка операционной системы позволит:

1. Сохранение всех настроек CR1000.
2. Удаление всех данных в итоговом хранилище.
3. Удаление диска USR.
4. Остановка текущей программы удаляет данные и очищает опции выполнения.
5. Удаление данных запускается с помощью команд **CardOut()** или **TableFile()**.

7.7.2.1.3. Обновление ОС с помощью DevConfig

Команда отправки программы является частью *DevConfig* и другого *ПО поддержки регистратора данных* *(стр. 654)***.** Расположение этой команды в разных вариантах программного обеспечения приведено в таблице Размещение команды Program Send (Отправка программы)

Размещение команды Program Send (Отправка программы)		
Программная поддержка регистратора данных	***Название кнопки***	***Размещение кнопки***
DevConfig	**Send Program (Отправка программы)**	**Вкладка Logger Control (Управление регистратором) в левом нижнем углу**
LoggerNet	**Send New... (Отправить новый...)**	**Окно Connect (подключение) в нижнем правом углу**
PC400	**Send Program (Отправка программы)**	**Главное окно в нижнем правом углу**
PC200W	**Send Program (Отправка программы)**	**Главное окно в нижнем правом углу**
RTDAQ	**Send Program (Отправка программы)**	**Главное окно в нижнем правом углу**

Этот метод позволяет CR1000 сохранить параметры (начиная с16-й версии ОС). Новый файл ОС временно сохраняется в памяти SRAM CR1000, для чего требуется:

- Наличие достаточного объема памяти. Перед отправкой ОС, возможно, потребуется удалить другие файлы на дисках ЦП и USR, а также весь диск USR. Поскольку на CR1000 с 2 MB памяти недостаточно места для ОС версии 25 и старше.
- Для освобождения места будет очищена SRAM, поэтому параметры запуска программы будут потеряны. Если CR1000 связь контролируется текущей программой, сначала загрузить программу «default.cr1» CRBasic на диск ЦП. «Default.cr1» будет запущен по умолчанию после того, как CR1000 компилирует новую ОС и удалит текущие параметры запуска.

Процедура выполнения

Из окна *LoggerNet* **Connect** (Подключение) выполните следующие действия:

1. Подключитесь к устройству.
2. Сбор данных.
3. Нажмите **Send New... (Отправить новый...).**
4. Выберите файл ОС для отправки.
5. Перезапустите текущую программу с помощью функции **File Control** (Управление файлами) или отправьте новую программу с помощью *Редактора CRBasic* и укажите новые параметры запуска.

Преимущества и недостатки

Это лучший способ загрузки новой операционной системы на CR1000, при котором сохраняются его настройки (в большинстве случаев). Это означает, что вы сможете и дальше поддерживать связь с устройством, поскольку сохраняется адрес PakBus и поддерживается соединение через TCP-клиент PakBus. Кроме того, при соединении по протоколу TCP/IP передача файлов выполняется гораздо быстрее, чем загрузка новой ОС напрямую через *DevConfig*.

Недостаток заключается в том, что при очистке параметров запуска текущей программы вы можете потерять связь с устройством, если питание переключается на периферийное устройство, контролируемое специальной программой, которая, например, включает/выключает мобильный модем для экономии энергии.

Кроме того, нехватка памяти может привести к нестабильности. Поэтому перед отправкой нового файла ОС желательно очистить память. Если был определен диск USR, его также, вероятно, придется удалить.

Такая загрузка операционной системы позволит:

1. Сохранение всех настроек CR1000.
2. Удаление всех данных в итоговом хранилище.
3. Остановите текущую программу (остановка и удаление данных) и очистите настройки выполнения программы.
4. Удаление данных запускается с помощью команды CardOut().

7.7.2.1.4. Обновление ОС с помощью DevConfig

Процедура выполнения

1. Сохраните текстовый файл *powerup.ini* *(стр. 386)* и файл операционной системы .obj на внешнем запоминающем устройстве.
2. Вставьте внешнее запоминающее устройство в регистратор данных.
3. Цикл включения-выключения регистратора данных.

Преимущества и недостатки

Это отличный способ переустановить ОС без ноутбука в полевых условиях. Его недостатком является то, что, если вы хотите выполнить больше одного действия с «powerup.ini», например, изменить ОС и загрузить новую программу, требуется использовать отдельные карты памяти или изменить файл «.ini» между двумя задачами, которые вы хотите выполнить.

Такая загрузка операционной системы позволит:

1. Сохраните всех настройки регистратора данных.
2. Удаление всех данных в итоговом хранилище.
3. Сохраните USR-диск и сохраненные на нем данные.
4. Поддержка опций выполнения программы.
5. Удаление данных запускается с помощью команд **CardOut()** или **TableFile()**.

Вкладка DevConfig **Send OS** (Отправить ОС):

- Если у вас возникли проблемы связи на CR1000.
- Если необходимо CR1000 можно вернуть к исходной конфигурации.

Команда Send Program (Отправка программы) *(стр. 524)* или **Send New...** (Отправить новый...):

- Если вы хотите отправить ОС удаленно.
- Если вас не слишком беспокоят последствия.

вкладка: File Control (Управление файлами):

- Если вы хотите удаленно обновить ОС.
- Если подключение к CR1000 осуществляется только по IP.
- Если у вас есть доступ к IP, и вы хотите изменить ОС для тестирования.

Внешняя память и файл «PowerUp.ini»:

- Если вы хотите изменить ОС без компьютера

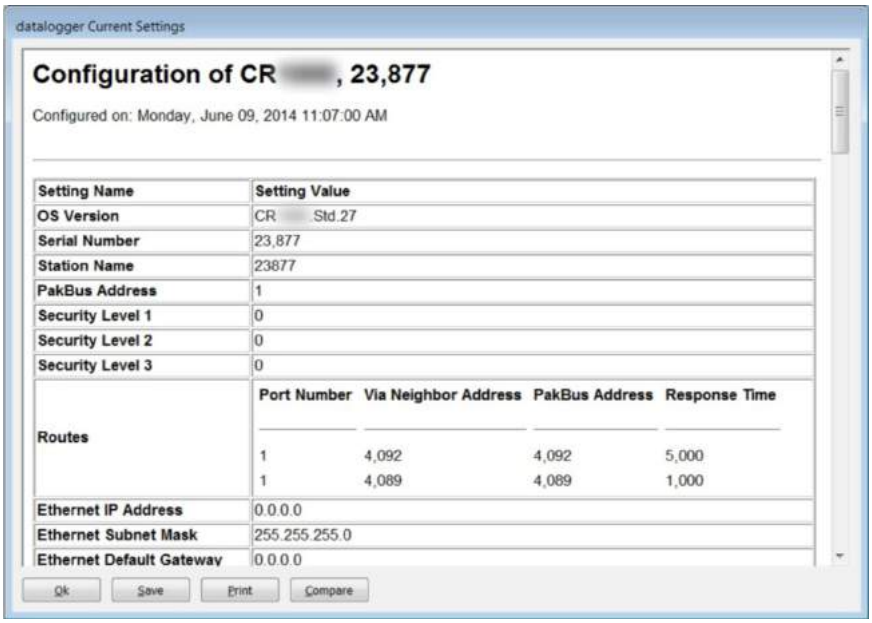
7.7.2.2. Восстановление заводских настроек по умолчанию

В *DevConfig* нажатие кнопки **Factory Defaults** (Значения по умолчанию) в нижней части вкладки **Settings Editor** (Редактор настроек) посылает в CR1000 команду на возврат к заводским настройкам по умолчанию. Возвращенные значения не вступят в силу, пока изменения не будут применены.

7.7.2.3. Сохранение и восстановление конфигурации

В *DevConfig* нажатие **Save** (Сохранить) на общем экране сохраняет конфигурацию в файл XML. Этот файл можно использовать для загрузки сохраненной конфигурации в CR1000, нажав **Read File** (Считать файла) и **Apply** (Применить).

Рисунок 38. Дополнительная информация о конфигурации CR1000



7.8. Программирование CRBasic — подробная информация

Связанные темы:

- *Программирование CRBasic — общие сведения* *(стр. 86)*
- *Программирование CRBasic — подробная информация* *(стр. 122)*
- *Программирование CRBasic — команды* *(стр. 537)*
- *Библиотека программных ресурсов* *(стр. 169)*
- *Справка для редактора CRBasic*

Программы создаются с помощью *Short Cut* *(стр. 528)* или *Редактора CRBasic* *(стр. 125)*. Старые программы CR10X и CR23X можно преобразовать в код CRBasic с помощью Transformer.exe (исполняемый файл *LoggerNet*). Максимальный размер программы составляет 490 Кбайт; однако большинство программ имеет значительно меньший размер.

7.8.1. Структура программы

Основные элементы программы CRBasic перечислены в таблице *Структура программы CRBasic* (стр. 123), а также показаны на примере CRBasic *Структура программы* (стр. 123).

Таблица 8. Структура программы CRBasic

Объявления	Определение использования памяти CR1000. Объявление констант, переменных, псевдонимов, единиц и таблиц данных.
Объявление констант	Список фиксированных констант.
Объявление переменных **Public**	Список/размер переменных, видимых на экране во время выполнения программы.
Объявление переменных **Dim**	Список/размер переменных, невидимых на экране во время выполнения программы.
Определение **псевдонимов**	Определение псевдонимов переменных.
Определение **единиц измерения**	Определение технических единиц измерения для переменной (по желанию). Единицы измерения строго соответствуют документации. CR1000 не использует единиц измерения и не ведет проверку их точности.
Определение таблиц данных.	Определение сохраненных таблиц данных.
Триггер процесса/хранения	Набор триггеров того, когда необходимо сохранить данные. Триггеры могут иметь фиксированный интервал времени, срабатывать при выполнении условия или использовать оба этих параметра.
Размер таблицы	Установить размер таблицы данных.
Другие устройства хранения данных онлайн	Можно использовать передачу данных на запоминающее устройство или карту памяти Campbell Scientific.
Обработка данных	Данные списка, которые должны хранится в таблице данных, например, образцы, средние, максимальные и минимальные значения и т. д.
	Процессы или расчеты, которые повторяются во время выполнения программы, можно оформить в виде подпрограммы, которая вызывается в случае необходимости, без постоянного построения кода.
Начало программы	Начало программы определяет начало операторов, определяющих действия CR1000.
Установить интервал сканирования	Сканирование имеет интервал для серии измерений.
Измерения	Введите измерения, которые необходимо выполнить.
Обработка	Введите любую дополнительную обработку.
Вызов таблицы данных	Таблицы объявленных данных должны вызываться для обработки и хранения.
Инициируйте элементы управления.	Проверьте измерения и инициируйте элементы управления, если это необходимо.
NextScan	Возврат к началу цикла для определения вида сканирования и ожидания следующего сканирования.
Завершение программы	Завершение программы указывает на окончание операторов, определяющих действия CR1000.

Структура программы CRBasic

```
Объявления

 Определение констант
Const RevDiff = 1
Const Del = 0 'default
Const Integ = 250
Const Mult = 1
Const Offset = 0

 Определение переменных типа Public
Public RefTemp
Public TC(6)

 Определение единиц измерения
Units RefTemp = degC
Units TC = DegC


 Определение таблиц данных
DataTable(Temp,1,2000)
  DataInterval(0,10,min,10)
  Average(1,RefTemp,FP2,0)
  Average(6,TC() ,FP2, 0)
EndTable


 Начало программы
BeginProg

  Установить интервал сканирования
 Scan(1,Sec,3,0)

    Измерения
    PanelTemp(RefTemp,250)
    TCDiff(TC()…Offset)

    Обработка (отсутствует в данном
    примере)


    Вызов таблицы данных
    CallTable Temp

    Контроль (отсутствует в данном
    примере)

    Переход к следующему сканированию
    NextScan

 Завершение программы
EndProg
```

Объявление констант

Объявление общих переменных, размерности массива и единиц измерения.

Определение таблицы данных

Объявления

Измерение

Вызов таблицы данных

Цикл сканирования

7.8.2. Написание и редактирование программ

7.8.2.1. Мастер программирования Short Cut

Short Cut является простой, управляемой в режиме меню программой, в которой приведены списки предварительно определенных измерений, вариантов обработки и алгоритмов управления, из которых пользователь может выбрать подходящий вариант. Вы выбираете, а *Short Cut* создает код CRBasic для выполнения этой задачи. *Short Cut* создает схему соединений, которая позволяет упростить подключение датчиков и внешних устройств. *Краткое руководство* *(стр. 41)* использует в работе примеры измерений *Short Cut*.

Для многих сложных приложений *Short Cut* удобно использовать на начальном этапе работы. После ввода максимально большого количества информации *Short Cut* создает шаблон программы, где указано форматирование большей части структур, процедур измерения и переменных, с которым уже можно работать. Затем эту программу можно дополнительно отредактировать с помощью программы *Редактора программ CRBasic*.

7.8.2.2. Редактор CRBasic

Прикладные программы CR1000 записываются в формате компьютерного языка BASIC (основной универсальный символический код команд), CRBasic (Campbell Recorder BASIC). *Редактор CRBasic* представляет собой текстовый редактор, который облегчает создание и изменение текстового файла ASCII, в котором содержится прикладная программа CR1000. *Редактор CRBasic* является компонентом *LoggerNet* *(стр. 655)*, *RTDAQ* и *ПО поддержки регистратора данных PC400* *(стр. 95)*.

К основным элементам CRBasic относятся:

- Переменные — названные пакеты памяти CR1000, в которых хранятся значения, которые обычно изменяются во время выполнения программы. Переменные, как правило, являются результатом измерений и обработки данных. Переменным присваивается буквенно-цифровое названия, они также могут быть отсортированы в массивы связанных данных.
- Константы — дискретные пакеты памяти CR1000, в которых хранятся определенные значения, которые не изменяются во время выполнения программы. Константам присваиваются буквенно-цифровые имена и значения на этапе начального объявления программы CRBasic.

Примечание. Ключевые слова и предварительно определенные константы зарезервированы для внутреннего использования CR1000. Если запрограммированная пользователем переменная окажется ключевым словом или предварительно определенной константой, произойдет ошибка выполнения или компиляции. Чтобы исправить эту ошибку, просто измените название переменной, добавив или удалив одну или несколько букв, цифр или подчеркивание (_) в ее имени, а затем повторно компилируйте и отправьте программу. В *Справке к редактору CRBasic* приведен список ключевых слов и предварительно определенных констант.

- Общие команды — команды и операторы, которые используются в большинстве языков BASIC, включая операторы управления программы, а также логические и математические операторы.
- Специальные команды — команды, уникальные для CRBasic, включая команды на проведение измерений и обработки, которые включают множество общих вычислений, используемых регистратором данных CR1000.

Перечисленные четыре элемента необходимо правильно разместить внутри структуры программы.

7.8.2.2.1. Добавление комментариев

Комментарии представляют собой неисполняемый текст, который добавлен в основную часть программы для оформления или уточнения программных алгоритмов.

Как показано в примере CRBasic *Добавление комментариев* *(стр. 126)*, перед комментарием добавляется одинарная кавычка ('). Комментарии можно добавить либо в виде отдельной строки, либо внутри кода CR1000. Когда компилятор CR1000 доходит до одинарной кавычки ('), он игнорирует остаток строки.

CRBasic: пример 2. Добавление комментариев

```
'В этом примере показан процесс добавление в программу комментариев. Комментарии размещены
'в двух местах: в виде отдельной сроки, как данное пояснение, или после
'оператора.

'Здесь начинается объявление переменных.
Public Start(6)                                 'Объявление начала массива времени

BeginProg
EndProg
```

7.8.2.2.2. Экономия памяти программы

В тех редких случаях, когда программы достигает максимального размера, можно использовать следующие методы экономии памяти:

- Объявление переменных в виде **DIM** вместо **Public.** Переменные **DIM** не требуют буферной памяти для сбора данных.
- Уменьшение массива до минимально необходимого размера. Массивы позволяют экономить память благодаря использованию скаляров, поскольку для них требуется меньше «мета-данных» на одно значение. Тем не менее, в грубом приближении, переменная 192 000 (4 КБ памяти) или 87 000 (2 КБ памяти) заполняет доступную память.
- Используйте массивы переменных с псевдонимами вместо отдельных переменных с уникальными именами. Псевдонимы требуют меньше памяти, чем уникальные имена переменных.
- Конкатенации строк к переменным **DIM** .
- Размер строчных переменных только до требуемого размера.

Читать далее. Раздел *Строковые операции* *(стр. 282)* *содержит дополнительную информацию об использовании и экономии памяти благодаря строковым переменным.*

7.8.3. Отправка программ CRBasic

CR1000 требует, чтобы программный файл CRBasic был отправлен в память для выполнения непосредственных измерений, обработки и операций хранения данных. Программный файл может иметь расширение «.crl» или «.dld», перед отправкой в CR1000 его можно архивировать с помощью алгоритма GZip. При получении файла CR1000 автоматически распаковывает его и использует так же, как и любой другой файл программы. Для получения дополнительной информации см. приложение *Программа и сжатие ОС* *(стр. 463)*.

Варианты отправки программы:

- Команда **Program Send** (Отправка программы) *(стр. 524)* в *ПО поддержки регистратора данных* *(стр. 95)*.
- **Программа** отправляет команду в *Служебную программу конфигурации устройства (DevConfig)* *(стр. 111)*.
- *Запоминающие устройство* Campbell Scientific или карта памяти *(стр. 653)*.

Желательно всегда извлекать данные CR1000 перед отправкой программы; иначе данные могут быть потеряны.

Читать далее. См. раздел *Управление файлами* *(стр. 382)*, документацию о запоминающем устройстве или карте памяти Campbell Scientific можно найти на сайте *www.campbellsci.com.*

7.8.3.1. Сохранение данных при отправке программы

При отправке программ в CR1000 с помощью программ, перечисленных в таблице *Варианты отправки программы, при которых происходит сброс памяти* *(стр. 127)*, стираются все данные из памяти.

Если желательно сохранить данные, отправлять программу лучше с помощью команды **File Control Send** (Отправка управления файлами) *(стр. 515)* или команды *Редактора CRBasic* **Compile, Save, Send** (Компилировать, сохранить, передать) в меню **Compile** (Компиляция). Перед отправкой программы отображается окно, показанное на рисунке *Окно Редактора CRBasic для отправки контрольного файла* *(стр. 127)*. Выберите **Run Now, Run On Power-up** (Выполнить сейчас, Выполнить при включении электропитания) и **Preserve data if no table changed** (Сохранять данные, если таблица не изменились) перед нажатием **Send Program** (Отправить программу).

Примечание. Для сохранения данных необходимо выбрать пункт **Preserve data if no table changed** (Сохранять данные, если таблица не изменились), независимо от того, подключено запоминающее устройство или карта памяти Campbell Scientific или нет.

Вне зависимости от используемого инструмента загрузки программы, если происходит изменение структур таблицы данных, перечисленных в таблице *Структуры таблицы данных* *(стр. 128)*, данные будут удалены при отправке новой программы.

Таблица 9. Варианты отправки программы, при которых происходит сброс памяти*

LoggerNet | *Connect (Подключение)* | **Program Send (Отправка программы)**
PC400 | **Clock/Program (Часы/программа)** | **Program Send (Отправка программы)**
PC200W | **Clock/Program (Часы/программа)** | **Program Send (Отправка программы)**
RTDAQ | **Clock/Program (Часы/программа)** | **Program Send (Отправка программы)**
DevConfig | **Logger Control (Управление регистратором** | **Program Send (Отправка программы)**
*Сброс памяти и установка атрибута программы **Run Always (Выполнять всегда)**

Рисунок 39. Окно Редактора CRBasic для отправки контрольного файла

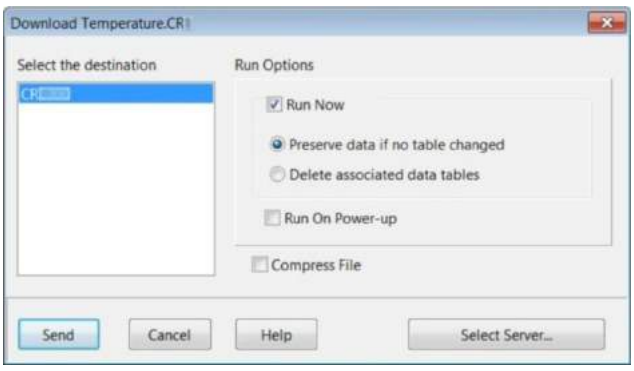


Таблица 10. Структуры таблицы данных
- Имя таблицы данных
- Выходной интервал или смещение данных
- Количество полей на одну запись
- Количество байт в поле
- Тип, размер, имя или положение поля
- Количество записей в таблице

7.8.4. Синтаксис программирования

7.8.4.1. Операторы программы

Программы CRBasic составлены из серии операторов. Каждый оператор обычно занимает одну строку текста в файле программы. Операторы состоят из команд, переменных, констант, выражений или их сочетания. «Команды» являются команды CRBasic. Как правило, оператор включает только одну команду. Однако некоторые команды, например, **If** (Если) и **Then** (То), могут входить в один оператор.

Списки команд и операторов приведены в приложении *Команды программирования на CRBasic* *(стр. 537)*. Полная обработка каждой команды и оператора описана в файлах *Справки* для *Редактора CRBasic* *(стр. 125)*.

7.8.4.1.1. Несколько операторов в одной строке

В одной строке можно разместить несколько операторов, если они разделены двоеточием (**:**). Это удобно для некоторых программ. Однако в целом, программы, где строки ограничены отдельными операторами, удобнее читать человеку.

В большинстве случаев операторы разделены: поскольку отдельные строки являются более безопасными. Однако, в случае подразумеваемой команды **EndIf**, CRBasic может вести себя неожиданным образом. В случае оператора **If ... Then ... Else ... EndIf**, где **EndIf** только подразумевается, он должен располагаться после последнего оператора в строке. Например:

```
If A then B : C : D
```

это не означает, что:

```
If A then B (implied EndIf) : C : D
```

скорее всего, это означает:

```
If A then B : C : D (implied EndIf)
```

7.8.4.1.2. Один оператор на нескольких строках

Длинные операторы, длина которые превышает ширину страницы *CRBasic Editor,* можно переносить на следующую строку, если разрыв оператора включает пробел и нижнее подчеркивание (_). Подчеркивание должно быть последним символом в текстовой строке, кроме дополнительного пробела.

Примечание. Длина оператора CRBasic ограничена 512 символами, независимо от переноса на другую строку.

Например:

```
Public A, B, _
   C,D, E, F

If (A And B) _
   Or (C And D) _
   Or (E And F) then ExitScan
```

7.8.4.2. Объявления с одним оператором

Объявления с одним оператором используются для переменных, констант, элементов, связанных с переменными и константами, а также названия устройства. Следующие команды, как правило, используются перед командой **BeginProg** (Начало программы):

- **Public;**
- **Dim;**
- **Constant (Константа);**
- **Units (Единица измерения);**
- **Alias (Псевдоним);**
- **StationName (Имя устройства).**

В таблице *Правила для имен* *(стр. 159)* перечислены объявленные имена и разрешенная длина. Дополнительную информацию об ограничениях для имен см. в разделе *Предварительно определенные константы* *(стр. 138)*.

7.8.4.3. Объявление переменных

Переменная представляет собой пакет памяти, которому присваивается буквенно-цифровое имя. Во время выполнения программы через переменные проходят измерения и результаты обработки. Переменные объявляются как **Public** или **Dim**. Переменные **Public** можно просматривать с помощью *цифровых мониторов* *(стр. 521)*. Переменные **Dim** нельзя просмотреть. Общим переменным можно присвоить свойство только для чтения с помощь команды **ReadOnly** (Только для чтения), в этом случае их нельзя изменить с помощью цифрового монитора. Однако программа имеет к такой переменной доступ для чтения/записи.

Объявленные переменные инициализируются один раз при запуске программы. Кроме того, переменные, которые используются в операторах **Function()** или **Sub()** или объявляются в теле функции или подпрограммы, являются локальными для этой функции или подпрограммы.

Имена переменных могут содержать до 39 символов, однако для большинства переменных оно не должно превышать 35 символов. Это позволяет использовать четыре дополнительных символа, которые добавляются в качестве суффикса к имени переменной, когда она выводится в таблицу данных. Имена переменных могут содержать следующие символы:

- от A до Z
- от a до z
- от 0 до 9
- _ (символ подчеркивания)
- $

Имена должны начинаться с буквы, подчеркивания или знака доллара. Пробелы и кавычки не используются. Имена переменных не чувствительны к регистру.

Несколько переменных могут быть объявлены в одной строке, через запятую:

```
Public RefTemp, AirTemp2, Batt_Volt
```

В объявлении переменным может быть присвоено начальное значение. Ниже приведен пример объявления переменой и присвоения ей начального значения.

```
Public SetTemp = {35}
```

В строковых переменных размер строки по умолчанию составляет 24 символов (в апреле 2013 года в версии 26 ОС, это параметр был изменен с 16 на 24).

7.8.4.3.1. Объявление типов данных

Переменные и значения данных, сохраненные в конечной памяти, могут быть настроены на различные типы данных для оптимизации исполнения программы и использования памяти.

Объявление переменных **Dim** или **Public** позволяет добавить необязательную характеристику **As** (Как), которая определяет тип данных. Тип данных по умолчанию (при объявлении без характеристики) — ***IEEE4*** с плавающей точкой, что эквивалентно объявлению формата **Float** (Плавающее значение). Типы переменных перечислены в таблице *Типы данных в памяти переменных (стр. 131, 130)*. Типы данных в памяти окончательных данных приведены в таблице *Типы данных в памяти окончательных данных (стр. 131)*. В примере CRBasic *Объявление типа данных (стр. 132)* показаны различные типы данных, которые используются в объявлениях и выходных разделах программы.

CRBasic позволяет смешивать типы данных в пределах одного массива переменных; однако такая практика может привести возникновению, по меньшей мере, одной проблемы. ПО поддержки регистратора данных не способно эффективно обрабатывать различных типов данных для одного имени поля. Следовательно, оно портит имена полей в заголовках файлов данных.

Таблица 11. Типы данных в памяти переменных

Название	*Команда*	*Описание*	*Размер слова (байт)*	*Комментарии*	*Разрешение/диапазон*
Плавающее	***Как плавающее значение или как IEEE4***	IEEE с плавающей точкой	4	Тип данных для всех переменных, если не указано иное. Стандарт IEEE 754	От ±1,4E-45 до ±3,4E38
Длинный	***As Long***	Целое число	4	Используется для сохранения данных в диапазоне ±2 147 483 648 Скорость: обработка целых чисел выполняется быстрее, чем чисел с плавающей запятой. Разрешение: 32 бита. По сравнению в 24 бита в IEEE4. Подходит для хранения целых чисел, натуральных чисел и целочисленных переменных в памяти окончательных данных. При хранении нецелочисленных данных, дробная часть значения теряется.	От –2 147 483 648 до +2 147 483 647
Логический	***As Boolean (Как логический)***	Целое число	4	Подходит для хранения значения истина или ложь, например, состояния флагов и портов управления. 0 всегда ложь. –1 всегда истина. В зависимости от сферы применения, любое другое число может быть интерпретировано как истинное или ложное. См. раздел *Истина = –1, ложь = 0 (стр 164)*.	Истина = –1 или любое число ≥1 Ложь = любое число ≥0 и <1

Таблица 11. Типы данных в памяти переменных

Название	Команда	Описание	Размер слова (байт)	Комментарии	Разрешение/диапазон
Строчный	***As Sitting (как строка)***	Строка ASCII	Минимум: 3 (4 с нулевым ограничителем) По умолчанию: 24 Максимум: ограничено только объемом доступной памяти CR1000.	См. предупреждения.[1] Размер строки определяется операционной системой CR1000 и программой CRBasic. При преобразовании из **STRING** (Строчный) во **FLOAT** (плавающий) формат, числа в начале строки преобразуются, но преобразование останавливается, когда встречается нечисловое значение. Если строка начинается с нечислового значения, то **FLOAT** (Плавающий) получает значение **NAN**. Если строка содержит несколько числовых значений, разделенных нечисловыми, то для синтаксического анализа числовых значений можно использовать команду **SplitStr()**. См. раздел *Операции со строками (стр. 282)* и *Последовательный интерфейс ввода/вывода (стр. 245)*.	Если не указано иное, размер строки составляет 24 байта или символа. Размер строки выделяется в значениях, кратных четырем байтам; например, для **строки * 25, строки * 26, строки * 27** и **строки * 28** выделяется 28 байт (27 можно использовать). Минимальный размер строки равен 4 (3 можно использовать). Для получения дополнительной информации см. разделе *Справка по редактору CRBasic*. Максимальная длина ограничена только доступной памятью CR1000.

[1] ВНИМАНИЕ! При использовании очень длинной строки для переменной, определенной как **Public** (Общая), операции *ПО поддержки регистратора данных (стр. 654)* будут часто передавать всю строку по каналу связи. Если пропускная способность канала ограничена или если оплата зависит от объема переданной информации, возможно, следует объявить переменную **Dim**.

Таблица 12. Типы данных в памяти окончательных данных

Название	Аргумент	Описание	Размер слова (байт)	Комментарии	Разрешение/диапазон
FP2	***FP2***	Campbell Scientific с плавающей точкой	2	Тип окончательных данных в памяти по умолчанию. Используйте ***FP2*** для хранимых данных, требующих 3 или 4-значащих цифр. Если требуется большее количество цифр, используйте тип ***IEEE4*** или смещение.	**Нуль** / **Минимум** / **Максимум**: 0,000 / ±0,001 / ±7999, **Абсолютная величина** / **Десятичный разряд**: 0–7,999 / X,XXX; 8–79,99 / XX,XX; 80–799,9 / XXX,X; 800–7999, / XXXX,
IEEE4	***IEEE4 или плавающий***	IEEE с плавающей точкой	4	Стандарт IEEE 754	От ±1,4E-45 до ±3,4E38
Длинный	***Длинный***	Целое число	4	Используется для сохранения данных в диапазоне ±2 147 483 648 Скорость: обработка целых чисел выполняется быстрее, чем чисел с плавающей запятой. Разрешение: 32 бита. По сравнению в 24 бита в IEEE4. Подходит для хранения целых чисел, натуральных чисел и целочисленных переменных в памяти окончательных данных. При хранении нецелочисленных данных, дробная часть значения теряется.	От –2 147 483 648 до +2 147 483 647

Таблица 12. Типы данных в памяти окончательных данных

Название	Аргумент	Описание	Размер слова (байт)	Комментарии	Разрешение/диапазон
UINT2	***UINT2***	Целое число без знака	2	Используется для хранения положительных данных ≤+65535. Используется для хранения порта или флага состояния. См. пример CRBasic *Загрузка двоичную информацию в переменную (стр. 139)*. Когда **Public FLOAT** (Общие с плавающим значением) преобразуется в *UINT2* в памяти окончательных данных, значения вне диапазона 0—65535 приводят к непригодности данных. **INF** преобразуется в **65535. NAN** преобразуется в 0.	От –0 до +65535
UINT4	***UINT4***	Целое число без знака	4	Используется для хранения положительных данных ≤+2147483647. Прочие виды использования, включают хранение длинных идентификационных номеров (например, для считывания со сканера штрих-кода), серийные номера или адреса. Может это может требоваться для использования в некоторых устройствах Modbus.	От 0 до 2147483647
Логический	***Логический***	Целое число	4	Подходит для хранения значения истина или ложь, например, состояния флагов и портов управления. 0 всегда ложь. –1 всегда истина. В зависимости от сферы применения, любое другое число может быть интерпретировано как истинное или ложное. См. раздел *Истина = –1, ложь = 0 (стр. 164)*. Для экономии памяти можно использовать *UINT2* или *BOOL8*.	Истина = –1 или любое число ≥1 Ложь = любое число ≥0 и <1
Bool8	***Bool8***	Целое число	1	8 бит (0 или 1) информации. Использует меньше места, чем 32-битный логический формат. Для хранения той же информации в логическом формате потребует 256 бит. См. раздел *Тип данных Bool8 (стр. 198)*.	Истина = –1, ложь = 0
NSEC	***NSEC***	Временная метка	8	Тип данных, состоящий из четырех байтов секунд времени, протекшего начиная с 1990 года, и четырех байтов наносекунд в секунде. Используется для записи и обработки данных времени. См. раздел *Тип данных NSEC (стр. 202)*.	1 наносекунда
Строчный	***Строчный***	Строка ASCII	Минимум: 3 (4 с нулевым ограничителем) По умолчанию: 24 Максимум: ограничено только объемом доступной памяти CR1000.	См. предупреждения.[1] Размер строки определяется операционной системой CR1000 и программой CRBasic. При преобразовании из **STRING** (Строчный) во **FLOAT** (плавающий) формат, числа в начале строки преобразуются, но преобразование останавливается, когда встречается нечисловое значение. Если строка начинается с нечислового значения, то **FLOAT** (Плавающий) получает значение **NAN**. Если строка содержит несколько числовых значений, разделенных нечисловыми, то для синтаксического анализа числовых значений можно использовать команду **SplitStr()**. См. раздел *Операции со строками (стр. 282)* и *Последовательный интерфейс ввода/вывода (стр. 245)*.	Если не указано иное, размер строки составляет 24 байта или символа. Размер строки выделяется в значениях, кратных четырем байтам; например, для **строки * 25, строки * 26, строки * 27** и **строки * 28** выделяется 28 байт (27 можно использовать). Минимальный размер строки равен 4 (3 можно использовать). Для получения дополнительной информации см. разделе *Справка по редактору CRBasic*. Максимальная длина ограничена только доступной памятью CR1000.

CRBasic: пример 3. Объявление типа данных

```
'В этом примере показаны различные объявления типа данных.

'Объявление типа данных, связанных с какой-либо одной переменной, происходит дважды: сначала как
'Public или Dim формата, а затем в сегменте DataTable/EndTable (Таблица данных/Конец таблицы).
'Если не указано иное, по умолчанию используется тип данных с плавающей запятой: As Float (Как с
'плавающей точкой) в объявлении переменной Public или Dim или FP2 в объявлении таблицы данных.

'Примеры переменной с плавающей точкой
Public Z
Public X As Float

'Пример длинной переменной
Public CR1000Time As Long
Public PosCounter As Long
Public PosNegCounter As Long

'Примеры логических переменных
Public Switches(8) As Boolean
Public FLAGS(16) As Boolean

'Пример строчной переменной
Public FirstName As String * 16 'позволяет использовать строку до 16 символов

DataTable(TableName,True,-1)
  'Пример хранения данных FP2
 Sample(1,Z,FP2)

  'Пример хранения данных IEEE4/Float
 Sample(1,X,IEEE4)

  'Пример хранения данных UINT2
 Sample(1,PosCounter,UINT2)

  'Пример хранения данных LONG
 Sample(1,PosNegCounter,Long)

  'Пример хранения данных STRING
 Sample(1, FirstName, String)

  'Пример хранения данных BOOLEAN
 Sample(8,Switches(),Boolean)

  'Пример хранения данных BOOL8
 Sample(2,FLAGS(),Boo18)

  'Пример хранения данных NSEC
 Sample(1,CR1000Time,Nsec)
EndTable

BeginProg
 'Здесь идет описание логики программы.
EndProg
```

7.8.4.3.2. Размерность числовых переменных

В некоторых приложениях требуются многомерные массивы. Размеры массива аналогичны пространственных измерениям (расстояние, площадь и объем). Одномерный массив определяется как

```
Public VariableName(x)
```

где (x) — индекс, обозначает x число переменных в виде серии.

Двумерный массив определяется как

```
Public VariableName(x,y)
```

где (x, y) — индексы, которые означают (x • y) количество переменных в квадрате в матрице x–y.

Трехмерный массив определяется как

```
Public VariableName (x,y,z)
```

где x,y,z) — индексы, имеющие (x • y • z) переменных в трехмерной матрице x–y–z. Размерность больше трех в CRBasic не используется.

При использовании переменных вместо целых чисел в качестве индексов измерений (в примере CRBasic *Использование индексов размера массива переменных* *(стр. 134)*) приведены рекомендации по объявлению индексов переменных **As Long**. Это позволяет более эффективно использовать ресурсы CR1000.

CRBasic: пример 4. Использование индексов размера массива переменных

```
' В этом примере программы демонстрируется использование индексов измерений в массивах. Переменная
' VariableName (Имя переменной) определяется тремя измерениями и 4 значениями в каждом индексе.
' Это указывает на то, массив имеет 64 элемента. Элемент 24 загружается со значением 2,718.
'
Dim aaa As Long
Dim bbb As Long
Dim ccc As Long
Public VariableName(4,4,4) As Float

BeginProg
  Scan(1,sec,0,0)
    aaa = 3
    bbb = 2
    ccc = 4
    VariableName(aaa,bbb,ccc) = 2.718
  NextScan
EndProg
```

7.8.4.3.3. Размеры строчных переменных

Строки могут быть иметь не более двух измерений. Третье «измерение» используется для указания символа в строке. См. раздел *Строковые операции* *(стр. 282)***.** Длину строки также можно объявить. См. таблицу *Типы данных в памяти переменных* *(стр. 131, стр. 130)***.**

Одномерный массив строк называется **StringVar**, он имеет пять элементов в массиве, каждый элемент длиной 36 символов объявляется следующим образом

```
Public StringVar(5) As String * 36
```

Объявлены пять переменных, каждая имеет 36 символов:

```
StringVar(1)
StringVar(2)
StringVar(3)
StringVar(4)
StringVar(5)
```

7.8.4.3.4. Объявление переменных Flag

Флаг является переменной, как правило, объявляется **As Boolean** *(стр. 508)*, он может иметь значение истина или ложь, выключен или выключен, двигаться или не двигаться и т. д. Программа может ветвиться исходя из значения флага. Раньше флаги просто использовались для того, чтобы информировать наблюдателя, что событие происходит или произошло. Хотя любая переменная любого типа данных может использоваться в качестве флага, на практике лучше всего работает использование булевых логических переменных, особенно переменных с именем «Флаг». В примере CRBasic *Объявление и использование флага* *(стр. 135)* показано изменение слов в строке на основании флага.

CRBasic: пример 5. Объявление и использование флага

```
'В этом примере программа демонстрирует и объявляет использование флагов как логических
'переменных и использование строк для отчета о состоянии флага. Для запуска демонстрации
'отправьте эту программу в CR1000, а затем измените значение переменных Flag(1) и Flag(2) на
'верное или ложное, чтобы увидеть, как логическая программа задает слова High (Высокий) или Low
'(Низкий) в переменных FlagReport(1) и FlagReport(2).
'Для того, чтобы присвоить флагу значение «Истина» при использовании LoggerNet подключения к
'цифровому монитору, просто нажмите на зеленую точка рядом с словом «Ложь». При использовании
'клавиатуры можно выбрать варианты «Истина» и «Ложь».

Public Flag(2) As Boolean
Public FlagReport(2) As String

BeginProg
  Scan(1,Sec,0,0)

   If Flag(1) = True Then
      FlagReport(l) = "High"
   Else
      FlagReport(l) = "Low"
   EndIf

   If Flag(2) = True Then
      FlagReport(2) = "High"
   Else
      FlagReport(2) = "Low"
   EndIf

  NextScan
EndProg
```

7.8.4.4. Объявление массивов

Связанные темы:

- *Объявление массивов* *(стр. 135)*
- Массивы мультипликаторов и смещений
- *VarOutOfBounds (Выход переменной за пределы)* *(стр. 488)*

Можно объявить несколько переменных одного и того же корневого имени. Полученная в результате серия одноименных переменных называется массивом. Массив создается путем добавления суффикса **(x)** к имени переменной. Количество создаваемых переменных X, которые отличаются только по названию, с помощью увеличения номера суффикса. Например, четыре оператора

```
Public TempC1
Public TempC2
Public TempC3
Public TempC4
```

можно просто сжать до

```
Public TempC(4).
```

Этот оператор создает в памяти четыре переменные ***TempC(1), TempC(2), TempC(3)*** и ***TempC(4).***

Массив переменных полезен в программных операциях, которые аналогично влияют на множество переменных. В примере CRBasic *Использование переменной массива в расчетах* *(стр. 136)* показан компактный код, который преобразует четыре температуры (°C) в °F.

В этом примере структура **For/Next** (Для/следующий) с приращением переменной используется для определения, к каким элементам массива будет применяться логическая операция. Функция CRBasic **For/Next** (Для/следующий) будет работать только для элементов массива, которые четко определены, и проигнорирует остальные. Если на определенный элемент массива нет ссылки, как в случае объявления

```
Dim TempC()
```

CRBasic делают ссылку только первый элемент массива **TempC (1).**

В примере CRBasic *Конкатенация чисел и строк* *(стр. 284)* показан вариант использования оператора присвоения += *(стр. 565)* при работе с массивами.

CRBasic: пример 6. Использование массива переменных в расчетах

```
В этом примере программы демонстрируется использование массива переменных для упрощения кода. В
этом примере два массива переменных используются для преобразования четырех измерений
температуры из градусов Цельсия в градусов по Фаренгейту.

Public TempC(4)
Public TempF(4)
Dim T

BeginProg
  Scan(1,Sec,0,0)

    Therm107(TempC(),1,1,Vx1,0,250,1.0,0)
    Therm107(TempC(),1,2,Vx1,0,250,1.0,0)
    Therm107(TempC(),1,3,Vx1,0,250,1.0,0)
    Therm107(TempC(),1,4,Vx1,0,250,1.0,0)

    For T = 1 To 4
       TempF(T) = TempC(T) * 1.8 + 32
    Next T

  NextScan
EndProg
```

7.8.4.5. Объявление локальных и глобальных переменных

Расширенные программы могут использовать *подпрограммы* *(стр. 288)* или *функции* *(стр. 602)*, каждая из которых может иметь множество переменных **Dim**, привязанных к этой подпрограмме или функции. Такие переменные называются *локальными* переменными. Имена локальных переменных могут быть идентичны именам *глобальных переменных* *(стр. 517)*, а также именам локальных переменных, объявленных в других подпрограммах и функциях.

Эта функция позволяет создавать в CRBasic библиотеки многократно используемых подпрограмм и функций, которые не будут создавать конфликты имен переменных. Если программа с локальными переменными **Dim** пытается использовать их в глобальном масштабе, произойдет ошибка компиляции **необъявленной переменной**.

Чтобы отобразить локальную переменную, в тех случаях, когда ее перевод в общий формат Public приведет к конфликту имен, создайте образец местной переменной в таблице данных и отобразите таблицу элементов данных в *цифровом мониторе* *(стр. 521)*.

При обмене содержимым глобальных и локальных переменных, объявить каждую проходящую/получаемую пару с одинаковыми типами данных и длинами строк.

7.8.4.6. Инициализация переменных

По умолчанию на момент компиляции программы регистратора данных переменные устанавливаются равными нулю. Переменным при объявлении можно присвоить значение, отличное от нуля. Примеры синтаксиса показаны в примере CRBasic *Инициализация переменных* *(стр. 137)*.

CRBasic: пример 7. Инициализация переменных

```
'В этом примере программы показано, как можно объявить переменные в виде данных определенного
'типа.
'Переменные не объявляются как данные определенного типа по умолчанию для типа данных Float.
'Также показана загрузка значений в объявленные переменные.

Public aaa As Long = 1'Объявление одной переменной As Long и загрузка значения 1.
Public bbb(2) As String *20 = {"String_1", "String_2"}     'Объявление массива As String и
                                                           'загрузка строк в каждый элемент.
Public ccc As Boolean = True 'Объявление переменной As Boolean и загрузка значения True.

'Инициализировать переменную ddd, элементы 1,1 1,2 1,3 и 2,1.
'Элементы (2,2) и (2,3) по умолчанию к нулю.
Dim ddd(2,3)= {1.1, 1.2, 1.3, 2.1}

'Инициализировать переменную eee
Dim eee = 1.5

BeginProg
EndProg
```

7.8.4.7. Объявление констант

См. пример CRBasic *Использование объявления константы* *(стр. 137)*, где показано использование объявления констант. Константа может быть объявлена в начале программы, где ей присваивается буквенно-цифровое имя, которое будет использоваться вместо значения, чтобы программа могла ссылаться на имя, а не на само значение. Использование констант вместо значений упрощает чтение и изменение программы, а также делает ее более защищенной от непреднамеренных изменений. При объявлении с помощью **ConstTable/EndConstTable** константы можно изменить, пока программа выполняется с помощью меню экранной клавиатуры CR1000KD (**Configure, Settings | Constant Table** (Настройка, параметры | Таблица констант)) или команды **C** в эмуляторе терминала (см. раздел *Поиск и устранение неисправностей —эмулятор терминала* *(стр. 501)*).

Примечание. Использование всех заглавных буквы для имен констант облегчает их поиск в тексте.

CRBasic: пример 8. Использование объявления констант

```
 В этом примере программы демонстрируется использование объявления констант.

 Объявление переменных
Public PTempC
Public PTempF

 Объявление констант
Const CtoF_Mu1t = 1.8
Const CtoF_Offset = 32

BeginProg
  Scan(1,Sec,0,0)
    PanelTemp(PTempC,250)
    PTempF = PTempC * CtoF_Mult + CtoF_Offset
  NextScan
EndProg
```

7.8.4.7.1. Предварительно определенные константы

Многие слова зарезервированы для использования CRBasic. Эти слова не могут использоваться в программе в качестве имен переменных или таблиц. Предварительно определенные константы включают имена команд и действительные буквенно-цифровые имена параметров команд. Поскольку список предварительно определенных констант достаточно большой и часто расширяется по ходу разработки операционной системы, лучше всего во время разработки программы CRBasic часто компилировать программы. Компилятор отслеживает использование любых зарезервированных слов. Далее указаны предварительно определенные константы, которым присвоены значения:

- LoggerType = 1000 (как в CR1000).

Они могут быть полезны в программировании.

7.8.4.8. Объявление псевдонимов и единиц измерения

В программе CRBasic переменной может быть присвоено второе имя или псевдоним. Псевдонимы особенно полезны при использовании массивов. Массивы являются мощным инструментом комплексного программирования, но они размещают рядом несколько переменных с одинаковыми именами. Псевдонимы позволяют соединить возможности массива и использование уникальных имен.

Объявленное имя переменной может пересекаться с объявленным псевдонимом в теле программы CRBasic. Тем не менее, когда значение сохраняется в конечной памяти, оно будет иметь прикрепленное к нему имя псевдонима. Так что, если программе CRBasic требуется доступ к этому значению, она должна использовать имя, которое является псевдонимом.

Переменным в одномерном, двумерном и трехмерном массивах можно назначить единицы измерения. Единицы измерения не используются в других областях программирования, но добавляют смысл заголовкам таблиц полученных данных. Для разных элементов массива необходимо использовать разные единицы измерения, необходимо сначала назначить псевдонимы элементам массива, а затем единицы измерения для каждого псевдонима. Например:

```
Alias var_array(1) = solar_radiation
Alias var_array(2) = quanta

Units solar_radiation = Wm-2
Units variable2 = moles_m-2_s-1
```

7.8.4.9. Числовые форматы

CRBasic поддерживает четыре числовых формата. Наиболее распространенным среди них является использование базовых чисел. Также могут использоваться экспоненциальный, двоичный и шестнадцатеричные форматы, как показано в таблице *Форматы ввода чисел в CRBasic* *(стр. 139)*. Программные и аппаратные экраны Campbell Scientific поддерживают отображение только стандартных десятичных чисел.

Таблица 13. Форматы ввода чисел в CRBasic

Формат	*Пример*	*Десятичный эквивалент*
Стандартный	6.832	6,832
Научное представление	5,67E-8	5,67 X 10^{-8}
Двоичный	&B1101	13
Шестнадцатеричный	&HFF	255

Двоичный формат (1 = высокий, 0 = низкий) полезен при загрузке состояния нескольких флагов или портов в одну переменную. Например, сохранение двоичного числа &B 11100000 сохраняет состояние флагов от 8 до 1: флаги от 1 до 5 относятся к низким, а от 6 до 8 к высоким. В примере CRBasic *Загрузка двоичных данных в переменную* *(стр. 139)* показан алгоритм, который загружает двоичное состояние флагов в целую переменную LONG.

CRBasic: пример 9. Загрузка двоичных данных в переменную

```
' В этом примере программы показано, как двоичные данные загружаются в переменную. Двоичный
' формат (1 = высокий, 0 = низкий) полезен при загрузке состояния нескольких флагов
' или портов в одну переменную. Например, сохранение двоичного числа &B11100000
' сохраняет состояние флагов от 8 до 1: флаги от 1 до 5 относятся к низким, а от 6 до 8 к
' высоким.
' В этом примере показан алгоритм, который загружает двоичное состояние флагов в
' целую переменную LONG.

Public FlagInt As Long

Public Flag(8) As Boolean
Public I

DataTable(FlagOut,True, -1)
  Sample(1, FlagInt, UINT2)
EndTable

BeginProg
  Scan(1,Sec,3,0)

    FlagInt = 0
    For I = 1 To 8
      If Flag(I) = true Then
        FlagInt = FlagInt + 2 ^ (I - 1)
      EndIf
    Next I
    CallTable FlagOut

  NextScan
EndProg
```

7.8.4.10. Объявления с несколькими операторами

Объявления с несколькими операторами используются для объявления таблиц данных, подпрограмм, функций и происшествий. Связанные команды включают:

- **DataTable()/ EndTable**
- **Sub() / EndSub**
- **Function() / EndFunction**
- **ShutDown / ShutdownEnd**
- **DialSequence() / EndDialSequence**
- **ModemHangup() / EndModemHangup**
- **WebPageBegin() / WebPageEnd**

Объявления с несколькими операторами могут быть расположены следующим образом:

- До **BeginProg** (Начало программы).
- После **EndSequence** (Конечная последовательность) или бесконечной **Scan() / NextScan** (Сканирование() / Следующее сканирование) и перед **EndProg** (Конец программы) или **SlowSequence (Медленная последовательность).**
- Сразу после **SlowSequence (Медленная последовательность). Код SlowSequence** (Медленная последовательность) начинает выполнение после объявления любой последовательности. Только объявление последовательности может произойти после **EndSequence** (Конечная последовательность) и перед **SlowSequence** (Медленная последовательность) или **EndProg** (Конец программы).

7.8.4.10.1. Объявление таблицы данных

Данные хранятся в таблицах в соответствии с требованиями программы CRBasic. Таблица данных создается серией команд CRBasic, которые вводятся после объявления переменных, но до команды **BeginProg** (Начало программы). Эти команды включают:

```
DataTable()
    Условия выходного триггера
    Выход команд обработки
EndTable
```

Таблица данных по существу представляет собой файл, который находится в памяти CR1000. Этот файл записывается каждый раз, когда в него направляются данные. Пусковой механизм, который инициирует хранение данных, отключается либо часами CR1000, либо событием, например появлением высокой температуры. Количество объявленных таблиц данных ограничено только объемом доступной памяти CR1000 (для ОС ранее 28 версии предел составлял 30 таблиц данных). Таблицы данных могут хранить отдельные измерения, отдельные расчетные значения или сводные данные, например, средние, максимальные и минимальные значения в таблицах данных.

Каждая таблица данных связана с дополнительной служебной информацией, которая становится частью заголовка ASCII-файла (первые несколько строк файла) при загрузке данных на компьютер. Служебная информация включает:

- формат таблицы;
- тип регистратора данных и версию операционной системы;
- название программы CRBasic, запущенной на регистраторе данных;
- имя таблицы данных (не более 20 символов);
- буквенно-цифровые имена полей, прикрепленных к заголовку столбцов данных.

Эта информация называется «определение таблицы».

Таблица 14. Обычная таблица данных							
TOA5	CR1000	CR1000	1048	CR1000.Std.13.06	Процессор: Data.cr1	35723	Одна минута
TIMESTAMP	RECORD	BattVolt_Avg	PTempC_Avg	Темп C_Avg (1)	TempC_Avg (2)		
TS	RN	Вольт	Град. C	Град. C	Град. C		
		Средн.	Средн.	Средн.	Средн.		
7/11/2007 16:10	0	13,18	23,5	23,54	25,12		
7/11/2007 16:20	1	13,18	23,5	23,54	25,51		
7/11/2007 16:30	2	13,19	23,51	23,05	25,73		
7/11/2007 16:40	3	13,19	23,54	23,61	25,95		
7/11/2007 16:50	4	13,19	23,55	23,09	26,05		
7/11/2007 17:00	5	13,19	23,55	23,05	26,05		
7/11/2007 17:10	6	13,18	23,55	23,06	25,04		

В таблице *Обычная таблица данных* *(стр. 140)* показан файл данных в том виде, в котором он появляется после того, как связанная таблица данных загружается из запрограммированного CR1000 с кодом в CRBasic, приведенном в примере *Определение и использование таблицы данных* *(стр. 142)***.** Файл данных состоит из пяти или более строк. Каждая строка имеет одно или несколько полей. Первые четыре строки образуют заголовок файла. Последующие строки содержат данные.

Примечание. Файлы дискретных данных (ASCII или двоичных) также могут быть записаны на запоминающий диск CR1000 с помощью команды **TableFile()**.

Первая строка заголовка является строкой среды. Она состоит из восьми полей, перечисленных в таблице *Строка среды TOA5* *(стр. 141)*.

Таблица 15. Строка среды TOA5		
Поле	***Описание***	***Изменено***
1	**TOA5**	
2	Имя устройства	*Программа DevConfig* или CRBasic, действующая согласно настройкам
3	Модель регистратора данных	
4	Серийный номер регистратора данных	
5	Версия ОС регистратора данных	Новая ОС
6	Название программы регистратора данных	Новая программа
7	Сигнатура программы регистратора данных	Новая или измененная программа
8	Имя таблицы	Измененная программа

Вторая строка заголовка сообщает имена полей. Эта строка состоит из набора разделенных запятыми строк, которые определяют имена отдельных полей, согласно тому, что указано в программе регистратора данных. Если поле является элементом массива, к имени добавляется список разделенных запятой индексов в скобках, которые идентифицируют индекс массива. Например, переменная с именем **Values**, которая в программе регистратора данных объявлена двухмерным массивом, будет представлена четырьмя именами полей: **Values(1,1), Values(1,2), Values(2,1)** и **Values(2,2).** У скалярных переменных не будет индекса массива. В этой строке будет одно значение для каждого скалярного значения, определяемого таблицей. Имена полей по умолчанию представляют собой сочетание имен (или псевдонимов) переменных, из которых получены данные, к которому прибавляется суффикс из трех букв. Суффикс — это аббревиатура процесса передачи данных, с помощью которого они выводятся в хранилище. Например, **Avg** это аббревиатура для процесса обработки данных, который вызывается командой **Average()** (Среднее). Если программисту не подходят имена полей по умолчанию, для их настройки можно воспользоваться командой **FieldNames()**

(Имена полей). **TIMESTAMP, RECORD, Batt_Volt_Avg, PTemp_C_Avg, TempC_Avg(1)** и **TempC_Avg(2)** являются именами полей по умолчанию в таблице *Обычная таблица данных (стр. 140)*.

В третьей строке заголовка указаны единицы измерения для этого поля данных. Эти единицы измерения объявляются в начале программы CRBasic, как показано в примере CRBasic *Объявление и использование таблицы данных (стр. 142)*. Единицы измерения строго соответствуют документации. CR1000 не использует объявленные единицы и не проверяет их точность.

В четвертой строке заголовка указаны аббревиатуры процесса обработки данных, который использовался для создания поля данных. См. таблицу *Сокращения обработки данных (стр. 168)*.

В остальных строках описаны данные и хранение связанных с ними записей. В первом поле указывается временная метка, а во втором — номер записи (строка данных).

Как показано в примере CRBasic *Объявление и использование таблицы данных (стр. 142)*, объявление таблицы данных декларации начинается с команды **DataTable()** (Таблица данных) и заканчивается командой **EndTable()** (Конец таблицы). Между **DataTable()** (Таблица данных) и **EndTable()** (Конец таблицы) размещаются команды, которые определяют, какие данные должны храниться и при каких условиях. Таблица данных вызывается программой CRBasic для выполнения обработки сохраненных данных. Как правило, данные таблицы вызываются командой **CallTable()** (Вызов таблица) один раз при каждом **сканировании**.

CRBasic: пример 10. Объявление и использование таблицы данных

```
'В этом примере показано определение и использование таблиц данных.

'Объявление переменных
Public Batt_Volt
Public PTemp_C
Public Temp_C(2)

'Определение единиц измерения
Units Batt_Volt=Volts
Units PTemp_C=Deg_C
Units Temp_C()=Deg_C

'Описываем таблицы данных
DataTable(OneMin,True,-1)          'Обязательное начало объявления таблицы данных
   DataInterval(0,1,Min,10)        'Дополнительная команда запуска таблицы с интервалом в одну
                                   минуту
   Average(1,Batt_Vo1t,FP2,False)  'Дополнительная команда для средней переменной Batt_Volt
   Average(1,PTemp_C,FP2,False)    'Дополнительная команда для средней переменной PTemp_C
   Average(2,Temp_C(),FP2,False)   'Дополнительная команда для средней переменной Temp_C
EndTable                           'Обязательный конец объявления таблицы данных

DataTable(Table1,True, -1)
   DataInterval(0,1440,Min,0)  'Дополнительная команда запуска таблицы с интервалом 24 часа
   Minimum(1,Batt_Volt,FP2,False,False)  'Дополнительная команда для определения минимума
                                         Batt_Volt
EndTable
```

```
 Основная программа
BeginProg
  Scan(5,Sec,1,0)

    Измерения напряжения аккумулятора регистратора данных по умолчанию Batt_Volt:
   Battery(Batt_Volt)

    Измерение температуры коммутационной панели PTemp_C:
   PanelTemp(PTemp_C,_60Hz)

    Измерения с помощью термопары типа T (медь-константановая) Temp_C:
   TCDiff(Temp_C(),2,mV2_5C,1,TypeT,PTemp_C,True,0,_60Hz,1,0)

    Вызов таблицы данных и хранение данных
   CallTable(OneMin)
   CallTable(Table1)

  NextScan
EndProg
```

Команды DataTable() / EndTable

Команда **DataTable()** имеет три параметра: заданное пользователем буквенно-цифровое имя таблицы, например, ***OneMin***, условие срабатывания (например, ***Истина***), и размер таблицы в памяти, например, ***–1*** (автоматическое размещение).

- ***Name*** (Имя) — именем таблицы может быть любое сочетание цифр, букв и подчеркиваний до 20 символов в длину. Первый символ должен быть буквой или подчеркиванием.

Примечание. Хотя предварительный компилятор может пропустить и другие символы, однако, если правила выбора имени не соблюдаются, может возникать сбой.

- ***TrigVar*** — определяет, будут ли записи данных записаны в хранилище. Записи данных записываются в хранилище, если параметр ***TrigVar*** имеет значение Истина, а также выполняются другие условия, например **DataInterval**() (Интервал данных). Значение по умолчанию — ***–1 (истина). TrigVar*** может быть переменной, выражением или константой. ***TrigVar*** не контролирует промежуточную обработку. Промежуточная обработка контролируется выключаемой переменной ***DisableVar***, которая используется в качестве параметра во всех командах по обработке выводных данных (см. раздел *Команды обработки выходных данных* *(стр. 145)*).

Читать далее В разделе *TrigVar и DisableVar — контроль и обработка выходных данных* *(стр. 195)* описано использование команд ***TrigVar*** и ***DisableVar*** в специальных приложениях.

- ***Size*** (Размер) — размер таблицы представляет собой количество записей, которые сохраняются в таблицу до того, как новые данные будут записываться на место старых. Значение ***10*** означает, что в таблице хранится 10 записей; одиннадцатое значение перезапишет первую запись. ***-1*** означает, что память для таблицы выделяется автоматически при компилировании программы. В большинстве случаев автоматическое выделение памяти является предпочтительным вариантом, поскольку CR1000 задает размер всех таблиц таким образом, чтобы они заполнялись (и начинали перезаписывать данных) примерно в одно время. Выделяется приблизительно 2 КБ дополнительного места для таблицы данных, чтобы минимизировать возможность перезаписи новых данных в кольцевой памяти, когда *ПО поддержки регистратора данных* *(стр. 654)* получает самые старые данные одновременно с записью новых. Эти дополнительные записи не выводятся в таблице **Status** (Состояние) и не направляются в ПО поддержки регистратора, соответственно, не собираются.

Правила определения размера таблицы меняются, если в объявление таблицы входит команда **CardOut()** или команда **TableFile()** с параметром ***Option 64***. Эти команды поддерживают запись данных на карту памяти. Запись данных на карту требует выделения дополнительной памяти в качестве буфера копирования данных. CR1000 автоматически определяет нужный размер буфера (см. раздел *Карты памяти и номера записей* *(стр. 466)*).

В примере CRBasic *Объявление и использование таблицы данных* *(стр. 142)* показано создание таблицы данных с именем **OneMin**, сохранение данных выполняется один раз в минуту, согласно параметру **DataInterval(),** при этом самые последние записи сохраняются в SRAM-память. Записи **DataRecordSize** в таблице **DataTableInformation** сообщают о выделенной памяти с точки зрения количества записей, содержащихся в таблице.

Команда DataInterval()

DataInterval() дает команду CR1000 делать записи данных в заданном интервале и определять, когда запись была пропущена. Этот интервал не зависит от интервала **Scan() / NextScan**; тем не менее он должен быть кратным интервалу **Scan() / NextScan.**

Иногда, обычно из-за ошибки синхронизации, логическая программа не позволяет сделать запись. Если запись не сделана, CR1000 определяет бездействие как «пропуск» и увеличивает значение счетчика **SkippedRecord** (Пропущенные записи) в таблице **Status** (Состояние). Пропуски занимают значительную часть памяти в таблице данных, и этом может привести к тому, что таблица данных будет заполнена раньше, чем ожидалось. Параметр **Lapses** (Пропуска) команды ***DataInterval*** (Интервал данных) определяет реакцию CR1000 на пропуски. Для получения дополнительной информации см. таблицу *Варианты параметров пропуска в команде DataInterval()* *(стр. 145)*.

Примечание. Программная логика приводит к тому, что пропуски включают разные по длине интервалы сканирования (пропущенные циклы сканирования), использование **DataInterval** в управляемых событиями таблицах данных, и логика, которая направляет выполнение программы, согласно команде **CallTable()** (Вызов таблицы).

Таблица данных состоит из последовательных блоков данных 1 КБ. Каждый блок данных содержит временную метку, номер блока, а также одну или несколько записей. По умолчанию временная метка и номер записи для каждой записи не сохраняются. Вместо этого процедура извлечения данных ПО поддержки регистратора данных использует временную метку блока, номер блока во временной метке и номер сохранения записи в памяти компьютера. Этот метод позволяет меньше загружать пропускную способность сети и сберечь 16 байт памяти CR1000 на одну запись. Однако, когда при пропуске одной или нескольких последовательных записей возникает пропуск параметр **SkippedRecords** (Пропущенные записи) увеличивается, в блок данных добавляется 16-байтный подзаголовок с меткой времени и номером записи, до момента следующей записи. Следовательно, программы, где часто возникают пропуски, занимают значительный объем памяти без реальной пользы.

Если ***Lapses*** (Пропуск) присвоено значение ***20***, память, выделенная для таблицы данных, увеличивается на объем, достаточный для размещения 20 подзаголовков (320 байт). При возникновении более 20 пропусков фактическое количество записей, сделанное до перезаписи (кольцевая память), может быть меньше, чем было указано в команде **DataTable()**, **CardOut()** для CF-карты или команде **TableFile()** с параметром ***Option 64***.

Если в программе предполагается множество пропусков и не учитывается пропускная способность канала связи, параметру ***Lapses*** (Пропуски) необходимо присвоить значение ***0***, чтобы гарантировать, что CR1000 выделит достаточно памяти для каждой таблицы данных.

Таблица 16. Варианты параметров пропуска в команде DataInterval()	
Значение параметра Lapse (Пропуск) в DataInterval() ***Аргумент***	***Результат***
Lapse > 0	Номер записи в таблице фиксированный, в таблицу данных добавляется X блоков данных (1 Кбайт на один блок данных), если память доступна. Если номер записи присваивается автоматически, дополнительная память для таблицу не выделяется.
Lapse = 0	Для каждый записи всегда сохраняется временная метка и номер записи.
Lapse <0	При возникновении пропуска новый блок данных не создается. Временные метки записей, рассчитанные на основе извлечения данных, могут быть ошибочными.

Время сканирования и системное время

В некоторых приложениях требуется системное время (см. *Системное время (стр. 530)*), а не время сканирования (см. *Время сканирования (стр. 527)*). Для получения системного времени команда **CallTable()** (Вызов таблицы) должна выполняться за пределами цикла **Scan()** (Сканирование). См. раздел *Метки времени (стр. 303)*.

Команда OpenInterval()

По умолчанию CR1000 использует закрытые интервалы. Данные, выводимые в данных в таблицу командой **DataInterval** (Интервал данных), включают измерения, полученные только в течение текущего интервала. Промежуточная память, которая содержит измерения, очищается при следующем вызове таблицы данных, независимо от того, была сделана запись в таблицу данных или нет.

Как правило, данные временных рядов (средние и итоговые данные, максимумы и т. д.), которые выводятся в таблицу данных на основании интервала, включают только измерения на основании текущего интервала. После каждого интервала вывода данных память, содержащая измерения для данных временных рядов, очищается. Если интервал вывода данных пропущен (поскольку не выполнены все критерии вывода), память очищается при следующем вызове таблицы данных. Если команда **OpenInterval** (Открытый интервал) содержится в объявлении **DataTable()**, память не очищается. Это приводит к тому, что все измерения включаются в данные временных рядов, поскольку сохраняются последние данные о времени (хотя данные могут охватывать несколько интервалов вывода данных).

Примечание. Регистраторы данных на основе массивов, например, CR10X и CR23X, используют только открытые интервалы.

Команды по обработке выходных данных

Команды по обработке сохраненных данных (так называемые команды «обработки выходных данных) определяют, какие данные сохраняются в таблице данных. Когда таблица данных вызывается в программе CRBasic, команды по обработке сохраненных данных обрабатывают переменные, содержащие текущие входные данные или вычисления. Если условия запуска выполняются, например, если интервал выходных данных вывод превышен, обработанные значения сохраняются в таблице данных. В примере CRBasic *Объявление и использование таблицы данных (стр. 142)* сохраняются три средние значения.

Рассмотрим команду **Average()** (Среднее) в качестве примера процесса обработки сохраненных данных. **Average()** (Среднее) сохраняет среднее значение переменной в интервале выходных данных. Ее параметры:

- ***Reps*** — количество последовательных элементов в массиве переменных, для которого вычисляются средние значения. Параметру ***Reps*** присваивается значение ***1*** для усреднения ***PTemp***, и значение ***2*** для усреднения температуры двух термопар, оба значения находятся в массиве переменных ***Temp_C***.
- ***Source*** (Источник) — массив переменных для усреднения. Используются массивы переменных ***PTemp_C*** (массив 1) и ***Temp_C()*** (массив 2).
- ***DataType*** (Тип данных) — тип данных для сохраненного среднего значения (в данном примере используется тип данных ***FP2*** *(стр. 641)*).

Читать далее Для получения дополнительной информации о доступных типах данных см. *Объявление типов данных* *(стр. 130)*.

- ***DisableVar*** (Отключение переменной) — определяет, будет ли результат измерения или значение использоваться при обработке выводных данных. Результат измерения или значение не используются при обработке, если параметр ***DisableVar*** имеет значение ***истина (≠0).*** Например, если отключаемая переменная в команде **Average()** (Среднее) имеет значение ***Истина***, то текущее значение не включается в расчет среднего значения. В примере CRBasic *Использование отключаемой переменной* *(стр. 146)* и примере *Использование NAN для фильтрации данных* *(стр. 484)* показано, как можно использовать параметр ***DisableVar*** для исключения значений из процесса усреднения. В этих примерах ***DisableVar*** контролируется ***Flag1***. Когда ***Flag1*** имеет высокое значение или параметр ***True, DisableVar*** имеет значение ***Истина***. Когда параметр ***False, DisableVar*** имеет значение ***Ложь.*** Когда значение ***Ложь*** вводится в качестве аргумента ***DisableVar***, для усреднения используются все показания. Среднее значение переменной ***Oscillator*** не включает значения, возникшие, когда флаг ***Flag1*** имел высокое значение (***Истина***), в результате среднее значение равно ***2***; когда ***Flag1*** имеет низкое значение или ***Ложь*** (использованы все образцы), среднее значение составляет ***1,5***.

Читать далее. В разделах *TrigVar и DisableVar (стр 195) — контроль и обработка выходных данных* *(стр. 195)* и *Измерения и NAN* *(стр. 482)* описано использование ***TrigVar*** и ***DisableVar*** в специальных приложениях.

Читать далее. Для получения полного списка команд по обработке выходных данных см. раздел *Обработка окончательных данных (выход в память)* *(стр. 542)*.

CRBasic: пример 11. Использование отключаемой переменной

```
' В этом примере показано использование отключаемой переменной или параметра DisableVar, который
' применяется во многих командах по обработке выходных данных. Использование отключаемой
' переменной позволяет выборочно включать исходные данные при расчете среднего, максимального,
' минимального значений и т. д. Если отключаемая переменная имеет значение -1 или истина, данные
' не включаются; если 0 или ложь — данные включаются. Отключаемой переменной по умолчанию
' присваивается значение ложь.

' Объявление переменных и единиц измерения
Public Oscillator As Long
Public Flag(1) As Boolean
Public DisableVar As Boolean

' Описываем таблицы данных
DataTable(Os cAvgData,True,-1)
  DataInterval(0,1,Min,10)
  Average(1,Oscillator,FP2, DisableVar)
EndTable
```

```
 Основная программа
BeginProg
  Scan(1,Sec,1,0)

    Сброс и увеличение значения счетчика
   If Oscillator = 2 Then Oscillator = 0
   Oscillator = Oscillator + 1

    Процесс и контроль
   If Oscillator = 1
     If Flag(1) = True Then
       DisableVar = True
     EndIf
   Else
      DisableVar = False
   EndIf

    Вызов таблицы данных и хранение данных
   CallTable(OscAvgData)

  NextScan
EndProg
```

Количество записей

Точное количество записей, которые можно сохранить в таблице данных, определяется сложным набором правил, краткое описание которого можно найти в приложении *Количество записей в таблицах данных* *(стр. 466)*.

7.8.4.10.2. Объявление подпрограмм

Читать далее. Для получения дополнительной информации по программированию с помощью подпрограмм см. раздел *Подпрограммы* *(стр. 288)*.

Подпрограммы позволяют нескольким процессам в основной части программы вызывать один фрагмент кода. Объявление подпрограмм проводится до основного кода программы.

Примечание. Конкретную подпрограмму могут одновременно вызвать несколько программах последовательностей. Для сохранения целостности измерений и обработки данных, CR1000 определяет очередность вызова подпрограммы. Это позволяет выполнять вызовы последовательно в том порядке, в котором они были получены. Что может привести к неожиданным паузам из-за конфликта последовательности выполнения программ.

7.8.4.10.3. «Включаемый» файл

«Включаемый» файл является альтернативой использованию подпрограммы. «Включаемый» файл представляет собой файл программы CRBasic, расположенный на приводе ЦП CR1000, которые компилируется в качестве вставки в программу CRBasic. Он может также *выполняться самостоятельно* *(стр. 116)*. По существу, это подпрограмма в специальном файле, который хранится отдельно от основного файла программы. Он может быть использован один раз или несколько раз как основной программой, так и другими программами. Файл начинается с команды **SlowSequence** (Медленная последовательность) и может содержать любой код.

Процедура использования «включаемого» файла:

1. Запишите файл, который начинается с команды **SlowSequence**, и далее содержит любой код.

2. Отправьте файл в CR1000 с помощью инструментов меню **File Control** (Управление файлами) *ПО поддержки регистратора данных* *(стр. 95)*.
3. Введите путь и имя файла в параметр **Include File** (Включаемый файл) с помощью команд *DevConfig* или *PakBusGraph*.

На рисунках *Параметры «включаемого файла» в DevConfig* *(стр. 149)* и *Параметры «включаемого файла» в PakBusGraph* *(стр. 149)* показаны методы выбора параметров с помощью *DevConfig* или коммуникаций. Ограничения на длину файла отсутствуют. В примере CRBasic *Использование «включаемого файла» для контроля переключения 12 В* *(стр. 149)* показана программа, в которой файл управляет питанием модема; в примере CRBasic *«Включаемый файл» для контроля переключения 12 В* *(стр. 150)* приведен соответствующий код.

Рассмотрим в качестве примера «включаемого» файла CPU: «pakbus_broker.dld». При запуске CR1000 использует следующие правила:

1. Если регистратор запускается с помощью включения питания, любая программа с меткой «выполнять при включении электропитания» становится «текущей программой». В противном случае выбирается любой файл с меткой «выполнить сейчас». Такой порядок используется всегда и не зависит от этого параметра.
2. При наличии файла с этим параметром, он включается в выбранную выше программу.
3. Если текущий файл не выбран или если не удается скомпилировать текущий файл, регистратор данных запустит программу, заданную этим параметром в качестве текущей программы.
4. Если программу, указанную этим параметром, не удается запустить или если программа не указана, регистратор данных попробует запустить программу «default.cr1» с диска ЦП.
5. При отсутствии файла «default.cr1» или если его не удается скомпилировать, регистратор данных не будет запускать никаких программ.

Теперь CR1000 позволит оператору **SlowSequence** (Показать последовательность) занять место оператора **BeginProg** (Начало программы). Эта функция позволяет использовать указанный файл в качестве включаемого файла и в качестве программы по умолчанию.

Формальные синтаксические правила для данного параметра:

```
include-setting := device-name ":" file-name "." file-extension.
device-name     := "CPU" | "USR" | "CRD"
File-extension  := "did" | "crl"
```

Рисунок 40. Параметры «включаемого файла» в DevConfig

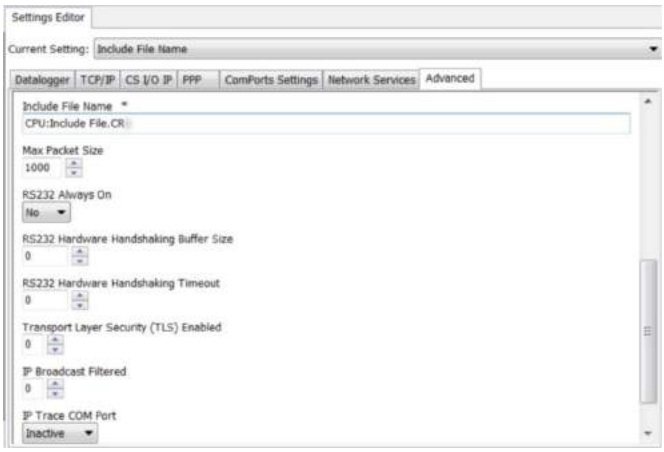
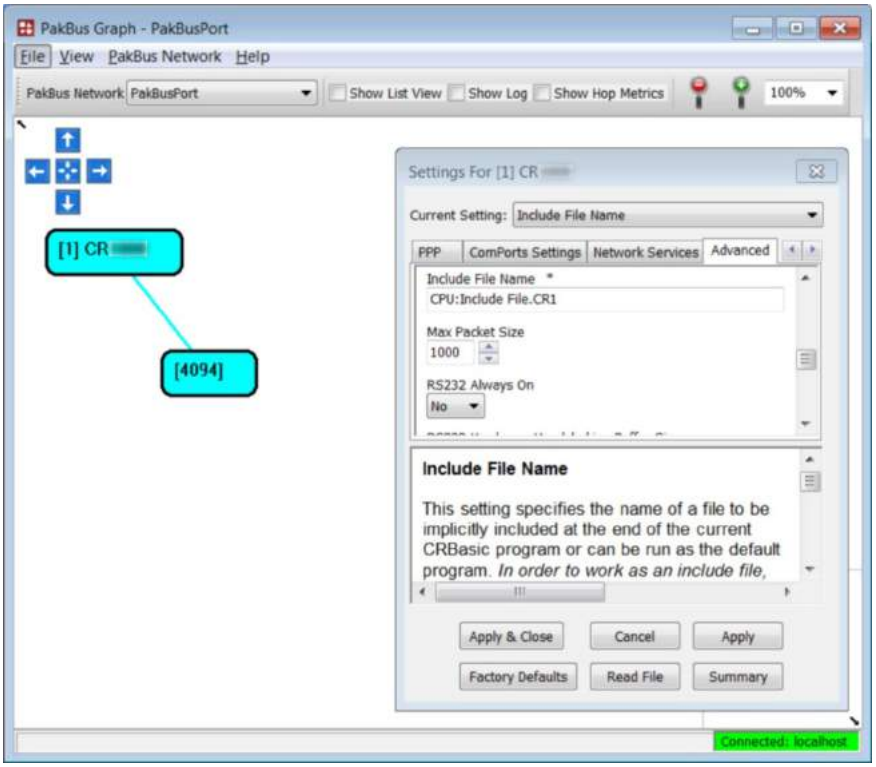


Рисунок 41. Параметры «включаемого файла» в PakBusGraph



CRBasic: пример 12. Использование «включаемого» файла

```
'В этом примере программы показано использование «включаемого» файла. «Включаемый» файл является
файлом CRBasic, который обычно
находится на приводе ЦП CR1000. Это, по существу, подпрограмма, которая
хранится в специальном файле отдельно от основной программы, но компилируется в качестве
элемента основной программы.
«Включаемый» файл может быть использован один или несколько раз и в нескольких программах.
«Включаемые» файлы начинаются с команды SlowSequence (Медленная последовательность) и могут
содержать любой код.
'
Процедура использования «включаемого» файла:
1. Скопируйте код из примера «Включаемый» файл для управления клеммой SW12 (стр. 150) в
```

```
   редактор CRBasic, назовите его «IncludeFile.cr1» и сохраните в ту же папку компьютера, где
   находится основной файл программы (это дозволяет проводить предварительную компиляцию). В том
   числе команду SlowSequence в качестве первого оператора, за которым может идти любой другой
   код.

2. Отправьте «включаемый» файл на диск ЦП CR1000 с помощью меню управления файлами
   ПО поддержки регистратора данных (стр. 654). Убедитесь, что параметры Run Now (Выполнить
   сейчас) и Run On (Выполнить на) не выбраны.
   Варианты включения, указанные в ПО при передаче файла.
3. Добавьте команду Include (Включить) к основной программе CRBasic в том месте, где
   вызывается «включаемый» файл (см. следующий код).
4. Введите путь в файловой системе CR1000 и имя файла после команды Include() (Включить), как
   показано на следующем примере кода.

Файл «IncludeFile.cr1» содержит код для управления электропитанием мобильного модем.

Сотовый телефон + кабель для подключения к клемме      Отрицательный провод (-)
SW12.
к G.

Public PTemp, batt_volt

DataTable(Test,1,-1)
  DataInterval(0,15,Sec,10)
  Minimum(1,batt_volt,FP2,0,False)
  Sample(1,PTemp,FP2)
EndTable

BeginProg
  Scan(1,Sec,0,0)
   PanelTemp(PTemp,250)
   Battery(Batt_volt)
   CallTable Test
  NextScan
  Include "CPU:IncludeFile.CR1" <<<<<<<<<<<<<<<< Код «включаемого» файла выполняется здесь
EndProg
```

CRBasic: пример 13. «Включаемый» файл для управления клеммой SW12.

```
В этом примере программы показано использование «включаемого» файла. См. документацию в примере
CRBasic.
Использование «включаемого» файла (стр. 149)
'
<<<<<<<<<<<<<<<<<<<<<<<<ПРИМЕЧАНИЕ. Команда BeginProg (Начало программы) отсутствует
SlowSequence <<<<<<<<<<ПРИМЕЧАНИЕ. Начинается с SlowSequence
  Scan(1,Sec,0,0)
   If TimeIntoInterval(9,24,Hr) Then SW12(1)     Модем включен в 9:00 (900 часов)
   If TimeIntoInterval(17,24,Hr) Then SW12(0)    Модем выключен в 17:00 вечера (1700 часов)
  NextScan
'
<<<<<<<<<<<<<<<<<<<<<<<<ПРИМЕЧАНИЕ. Отсутствует команда EndProg (Конец программы)
```

7.8.4.10.4. Объявление подпрограмм

Команды Function() / EndFunction позволяют создавать индивидуальные команды CRBasic. Их объявление проводится аналогично объявлению подпрограммы.

7.8.4.10.5. Объявление вспомогательных последовательностей

Последовательность состоит из двух или более операторов кода. Последовательности таблиц данных являются важной особенностью почти всех программ. Подпрограммы и функциональные последовательности, хотя и используются реже, имеют общую цель. В отличие от этого, приведенные далее последовательности используются только в определенных приложениях.

Последовательности отключения

Команды **ShutDownBegin/ShutDownEnd** используются для определения кода, который будет выполняться при каждом отключении программы выключается с помощью предписанных средств. Дополнительная информация приведена в разделе *Справка к редактору CRBasic.*

Последовательности набора

Команды **DialSequence/EndDialSequence** используются для определения кода маршрутизации пакетов на устройстве PakBus®. Дополнительная информация приведена в разделе *Справка к редактору CRBasic*.

Последовательности при зависании модема

Команды **ModemHangup/EndModemHangup** используются для вставки кода, который запускается при отключении COM-порта. Дополнительная информация приведена в разделе *Справка к редактору CRBasic*.

Последовательности для веб-страниц

Команды **WebPageBegin/WebPageEnd** задают веб-страницу, которая отображается, когда от внешнего источника приходит запрос на определенную страницу HTML. Дополнительная информация приведена в разделе *Справка к редактору CRBasic.*

7.8.4.11. Выполнение и приоритет задачи

Выполнение программных команд используется для выполнения трех следующих задач:

- Задача измерения — четко оговоренное по времени измерение с помощью датчиков, подключенных непосредственно к CR1000.
- Задача CDM — четко оговоренное по времени измерение и управление периферийными устройствами CDM *(стр. 509)*.
- Задача SDM — четко оговоренное по времени измерение и управление периферийными устройствами SDM *(стр. 527)*.
- Обработка — преобразование измеренных значений в используемые технические единицы измерения, выполнение вычислений, хранение данных, принятие решения о запуске управления и поддержание связи через последовательный интерфейс ввода/вывода.

Команды, которые обрабатываются каждой задачей, приведены в таблице *Задачи программы* *(стр. 152)*.

Эти задачи выполняются как в поточном, так и последовательном режиме обработки. В поточном режим задачи выполняются более или менее параллельно. А в последовательном режиме задачи выполняются более или менее последовательно. Когда программа скомпилирована, CR1000 проводит ее оценку и автоматически определяет, какой режим лучше использовать. При использовании команд **PipelineMode** (Поточный режим) или **SequentialMode** (Последовательный режим) в начале программы она переходит в тот или другой режим. Режим информация включается в сообщение, возвращаемое регистратором данных, которое отображается в *ПО поддержки регистратора данных* *(стр. 654)*. Компилятор предварительного прохода *Редактора CRBasic* возвращает аналогичное сообщение.

Примечание. Программу можно специально перевести в последовательный или поточный режим с помощью команд **SequentialMode** (Последовательный режим) или **PipelineMode**(Поточный режим) в разделе объявления.

Некоторые задачи в программе могут иметь более высокие приоритеты. Измерение, как правило, имеет приоритет над всеми остальными задачами. Приоритеты задач в последовательном и поточном режиме отличаются.

Таблица 17. Задачи программы

Измерение	*Цифровая задача*	*Обработка*
• Аналоговые измерения • Ток возбуждения • Считывание счетчиков импульсов • Считывание портов управления (**GetPort()**) • Выбор портов управления (SetPort()) • **VibratingWire()** • **PeriodAvg()** • **CS616()** • **Calibrate()**	• Команды SDM, за исключением **SDMSI04()** и **SDMI016()** • Команды CDM / CPI-устройств	• Обработка • Выходной параметр • Последовательный интерфейс ввода/вывода • **SDMSIO4()** • **SDMIO160** • **ReadIO()** • **WriteIO()** • Оценка выражения и параметров переменной при измерении и команд SDM

7.8.4.11.1. Поточный режим

Поточный режим обрабатывает измерения, а также большинство цифровых задач и задач обработки отдельно и, возможно, одновременно. Измерения выполняются в определенное время и с наивысшим приоритетом, что позволяет точно задавать время измерения, и, как правило, более эффективно обрабатывать данные и расходовать энергию.

Планирование в поточном режиме требует, чтобы программа была написана таким образом, чтобы измерения выполнялись в каждом цикле. Поскольку одновременно выполняется несколько задач, последовательность, в которой выполняются команды может не совпадать с порядком из появления в программе. Таким образом, в поточном режиме условные измерения не допускаются. Из-за точного выполнения команд измерения, обработка данных текущего сканирования (включая обновление общих переменных и хранение данных) откладывается до момента завершения всех измерений. Некоторые виды обработки, например, передача переменных в такие команды управления, как **PortSet()** и **ExciteV(),** не могут быть завершены до следующего сканирования.

При выполнении условия начала выполнения задачи, она помещается в очередь. Поскольку все задачи имеют одинаковый приоритет, задача ставится в конец очереди. Каждые 10 мс (или быстрее при запуске новой задач) текущая задача приостанавливается и переносится в конец очереди, и запускается следующая задача из очереди. Таким образом, все задачи получают рваное времени обработки в CR1000.

При этом все задачи имеют одинаковый приоритет. Однако, при возникновении конфликта задач, выполнение программ определяется графиком очередности в таблице *Приоритеты задач в поточном режиме (стр. 153)*.

Таблица 18. Приоритеты задач в поточном режиме
1. Измерения в основной программе.
2. Фоновая калибровка.
3. Измерения в медленных последовательностях.
4. Обработка.

7.8.4.11.2. Последовательный режим

При последовательном режиме команды выполняются в той последовательности, в которой они записаны в программе. Последовательный режим может работать медленнее, чем поточный режим, поскольку он выполняет только одну строку кода за раз. После выполнения измерения его результат преобразуется в значение, задаваемое аргументами обработки данных для этого измерения, и выполнение программы переходит к следующей команде. Построчное выполнение команд позволяет записывать в программу условные измерения.

Примечание. Точное время выполнения измерения в последовательном режиме может меняться, если есть другие измерения или виды обработки, которые должны выполняться при определенных условиях, а также при наличии сложных коммуникационных задач или других прерываний, например, доступ к запоминающему устройству или карте памяти Campbell Scientific.

При работе в последовательном режиме регистратор данных использует систему очередности обработки, аналогичную той, которая используется в поточном режиме. Основное различие в исполнении программы в последовательном режиме, заключается в отсутствии предварительного планирования измерений; вместо этого все команды выполняются в порядке, указанном программой.

Система приоритетов позволяет избежать конфликтов использования измерительного оборудования. Основное сканирование имеет наивысший приоритет и не дает выполнять другие последовательности действий на измерительном оборудовании, до момента завершения основного сканирования, включая обработку данных. Другие задачи, такие как обработка данных от других последовательностей команд и коммуникаций, могут выполняться одновременно с главной последовательностью. После завершения основного сканирования другие последовательности получают доступ к измерительному оборудованию в порядке их приоритета. После последовательности фоновой калибровки медленные последовательности выполняются в том порядке, в котором они были объявлены в программе.

Примечание. Измерение имеет приоритет над другими задачами, например обработкой данных и коммуникациями, чтобы обеспечить точный промежуток времени, необходимый для выполнения большинства команд измерения.

При инициализации переменных, когда в программе используется несколько последовательностей, необходимо соблюдать осторожность. Если какая-либо последовательность опирается на какой-то элемент (переменная, порт и т. д.), который инициализируется в другой последовательности, необходимо создать схему их взаимодействия и добавить ее в программу CRBasic, чтобы убедиться, что последовательность инициализации была завершена до момента продолжения обработки зависимой задачи. Это может быть сделано с помощью простой переменной или даже введения задержки, однако

необходимо понять, что операционная система CR1000 не будет сама получать подтверждение для двух независимых задач.

Аналогичные проблемы могут возникнуть при повторном использовании одной переменной в нескольких задачах. Отсутствие обмена сообщениями между этими двумя задачами в программе CRBasic может привести к непредсказуемым результатам. Пара команд **SemaphoreGet() и SemaphoreRelease()** позволяет предотвратить нежелательный доступ к объекту (переменной, COM-порту и т. д.) со стороны другой задачи, пока этот объект используется. Для получения информации об использовании *SemaphoreGet()* и **SemaphoreRelease(),** см. раздел *Справка к редактору CRBasic.*

7.8.4.12. Время выполнения

Время выполнения программы регулируется согласно времени выполнения команд, которое указано в следующей таблице.

Таблица 19. Времени выполнения команд

Команды	*Общие рекомендации*	*Синтаксис*

Scan() / NextScan	Используется в большинстве программ. Начинает/ заканчивает основное сканирование.	`BeginProg`
		`  Scan()`
		`  '.`
		`  '.`
		`  '.`
		`  NextScan`
		`EndProg`

SlowSequence / EndSequence	Используется, когда измерение или обработка данных должны выполняться при более медленных частотах, чем основная программа.	`BeginProg`
		`  Scan()`
		`  '.`
		`  '.`
		`  '.`
		`  NextScan`
		`  SlowSequence`
		`  Scan()`
		`  '.`
		`  '.`
		`  '.`
		`  NextScan`
		`  EndSequence`
		`EndProg`

SubScan / NextSubScan	Используется, когда измерение или обработка данных должны выполняться при более высоких частотах, чем основная программа.	`BeginProg`
		`  Scan()`
		`  '.`
		`  '.`
		`  '.`
		`  SubScan()`
		`  '.`
		`  '.`
		`  '.`
		`  NextSubScan`
		`  NextScan`
		`EndProg`

7.8.4.12.1. Команда Scan() / NextScan

Простые программы CR1000 часто полностью строятся на одной структуре **Scan() / NextScan**, при этом за пределами цикла сканирования остаются только объявление переменных и таблица данных. **Scan() / NextScan** создает бесконечный цикл; каждое выполнение которого синхронизировано с часами CR1000. Параметры **Scan()** позволяют менять период с шагом от 10 мс до 24 часов. Как показано в примере CRBasic *Синтаксис команд BeginProg / Scan() / NextScan / EndProg* *(стр. 155)*, программа CRBasic может быть достаточно короткой.

Пример CRBasic 14. Синтаксис команд BeginProg / Scan() / NextScan / EndProg

```
 В этом примере показано использование синтаксиса команд BeginProg/EndProg и Scan()/NextScan.

Public PanelTemp_

DataTable(PanelTempData,True,-1)
  DataInterval(0,1,Min,10)
  Sample(1,PanelTemp_,FP2)
EndTable

BeginProg '                    <<<<<<<Начало программы
  Scan(1,Sec,3,0) '            <<<<<<< Сканирование
    PanelTemp(PanelTemp_,250)
    CallTable PanelTempData
  NextScan '                   <<<<<<< Следующее сканирование
EndProg '                      <<<<<<<Завершение программы
```

Команда Scan() определяет частоту выполнение команды в программе, как показано в следующем фрагменте кода CRBasic:

```
Сканирование (интервал, единицы измерения, размер буфера,
счетчик)
Scan(1,Sec,3,0)
      Здесь указываются команды CRBasic.
ExitScan
```

Команда Scan() имеет четыре параметра:

- ***Interval*** (Интервал) —интервал между циклами сканирования. Интервал составляет: 10 мс ≤ ***интервал*** ≤ 1 день.
- ***Units*** (Единицы измерения) — единица измерения времени для данного интервала.
- ***BufferSize*** (Размер буфера) — размер (количество циклов сканирования) буфера в оперативной памяти, где хранятся необработанные результаты измерений. При работе в поточном режиме буфер позволяет проводить обработку при сканировании, чтобы откладывать проведение измерения на момент, когда это не затронет измерение. Использование размера по умолчанию *Редактора CRBasic* является нормальным. Рекомендации по устранению неисправностей см. в разделе *Пропущенное сканирование (стр. 487)*.
- ***Count*** (Счетчик) — количество циклов сканирования, выполняемых до перехода к команде **NextScan** (Следующее сканирование). Значение счетчика ***0*** означает постоянное зацикливание (или до появления команды **ExitScan** (Следующее сканирование)). В примере в CRBasic показан синтаксис программы сканирования, время сканирования составляет одну секунду, три сканирования буферизуются, измерение и хранение данных продолжаться неопределенное время.

7.8.4.12.2. Команды SlowSequence / EndSequence

Медленные последовательности включают автоматические и введенные пользователем последовательности. Фоновая калибровка представляет собой автоматическую медленную последовательность.

Вводимые пользователем медленные последовательности объявляются командой **SlowSequence** (Медленная последовательность) и выполняются вне основного сканирования программы. Медленные последовательности обычно выполняются с меньшей скоростью, чем основное сканирование. В программе может быть определено до четырех медленных последовательностей сканирования.

Команды медленных последовательностях сканирования выполняются в том момент, когда не запущено основное сканирование. При работе в поточном режиме измерения в медленных последовательностях ставятся о очередь после измерений основной программы, если это позволяет время. Поэтому измерения в медленной последовательности могут охватывать несколько интервалов сканирования в основной программе. Когда никакие измерения не

необходимо соединить, медленная последовательность сканирования выполняется независимо от основного сканирования, поэтому медленные последовательности без измерений могут выполняться с интервалом, который меньше и равен интервалу основного сканирования (по-прежнему с шагом 10 мс), без пропуска сканирования. При соединении измерений проверка пропущенных медленных последовательностей выполняется после завершения первого соединения, а не сразу после срабатывания интервала.

В последовательном режиме все команды медленных последовательностей выполняются тогда, когда указаны в программе в соответствии с приоритетом задачи.

Фоновая калибровка представляет собой автоматическую медленную последовательность, поскольку это сторожевая задача.

Читать далее. См. раздел *Автоматическая калибровка CR1000 — общие сведения* *(стр. 92)*.

7.8.4.12.3. Команды SubScan() / NextSubScan

Команды SubScan() / NextSubScan используются в контроле аналоговых мультиплексоров (для получения информации о доступных аналоговых мультиплексорах см. приложение *Аналоговые мультиплексоры* *(стр. 646)*) или для измерения аналоговых входов с большей скоростью, чем программа сканирования. **SubScan() / NextSubScan** можно использовать в **SlowSequence / EndSequence** с интервалом **0. Команда SubScan** не может быть вложенной. В **PulseCount** или измерение SDM не могут использоваться в субсканировании.

7.8.4.12.4. Приоритеты сканирования в последовательном режиме

Примечание. Измерение имеет приоритет над другими задачами, например обработкой данных и коммуникациями, чтобы обеспечить точный промежуток времени, необходимый для выполнения большинства команд измерения.

Чтобы избежать конфликтов использования измерительного оборудования, в последовательном режиме используется система приоритетов. Как показано на рисунке *Функциональная схема приоритета сканирования в последовательном режиме* *(стр. 158)*, основная последовательность сканирования имеет наивысший приоритет. Другие последовательности, например медленные последовательности и калибровочные сканирования, должны ожидать, пока измерительное оборудование освободиться от основного сканирования, включая само измерение и обработку.

Основное сканирование

Выполнение основного сканирования обычно происходит быстро, поэтому большую часть времени процессор может простаивать. Например, программа измерения погодных условий может проводить сканирование один раз в секунду, но выполнение программы может занимать лишь 250 мс, в результате 75 % доступного времени сканирования остаются неиспользованными. CR1000 может обеспечить эффективное использование времени сканирования для оптимизации программы и управления связью. Если это не отключено и не вытеснена из-за слишком напряженного графика, самокалибровка (см. раздел *Автоматическая калибровка CR1000 — общие сведения* *(стр. 92)*) имеет приоритет и использует некоторые промежутки времени сканирования. Если самокалибровка вытеснена, компилятор предварительного прохода CRBasic создает предупреждающее сообщение. Остальные приоритеты включают сканирования в медленных последовательностях в том порядке, в котором они запрограммированы, а также цифровые триггеры. Далее приведено краткое введение в правила и приоритеты, регулирующие использование времени между сканированием в последовательном режиме. Правила и приоритеты, регулирующие поточный режим, несколько сложнее.

Разрешение продолжать измерения предоставляется с помощью измерения *семафор (стр. 527)*. Основные сканирования с измерениями имеют приоритет на получение семафора перед измерениями при калибровке или сканировании в медленной последовательности. Семафор включается с помощью основного сканирования в начале, сканирование включает измерения. Семафор отключается только после окончания выполнения последней команды основного сканирования.

Сканирования в медленной последовательности

Сканирования в медленной последовательности начинаются после команды **SlowSequence**. Они начинают обработку задач до проведения измерения, но делают остановку, если необходимо дождаться семафора для измерения. Медленные последовательности отключают семафор *(стр. 527)* после полного выполнения каждой команды измерения, чтобы позволить основному сканированию получить семафор, когда это необходимо. Если семафор измерения устанавливается с помощью сканирования в медленной последовательности и начало основного сканирования попадает в начало очереди, основное сканирование не начинается, пока не получит семафор; то есть ожидает, пока медленная последовательность отключит семафор. Сканирование в медленной последовательности не удерживает семафор для всего сканирования. Он освобождает семафор после каждого использования оборудования.

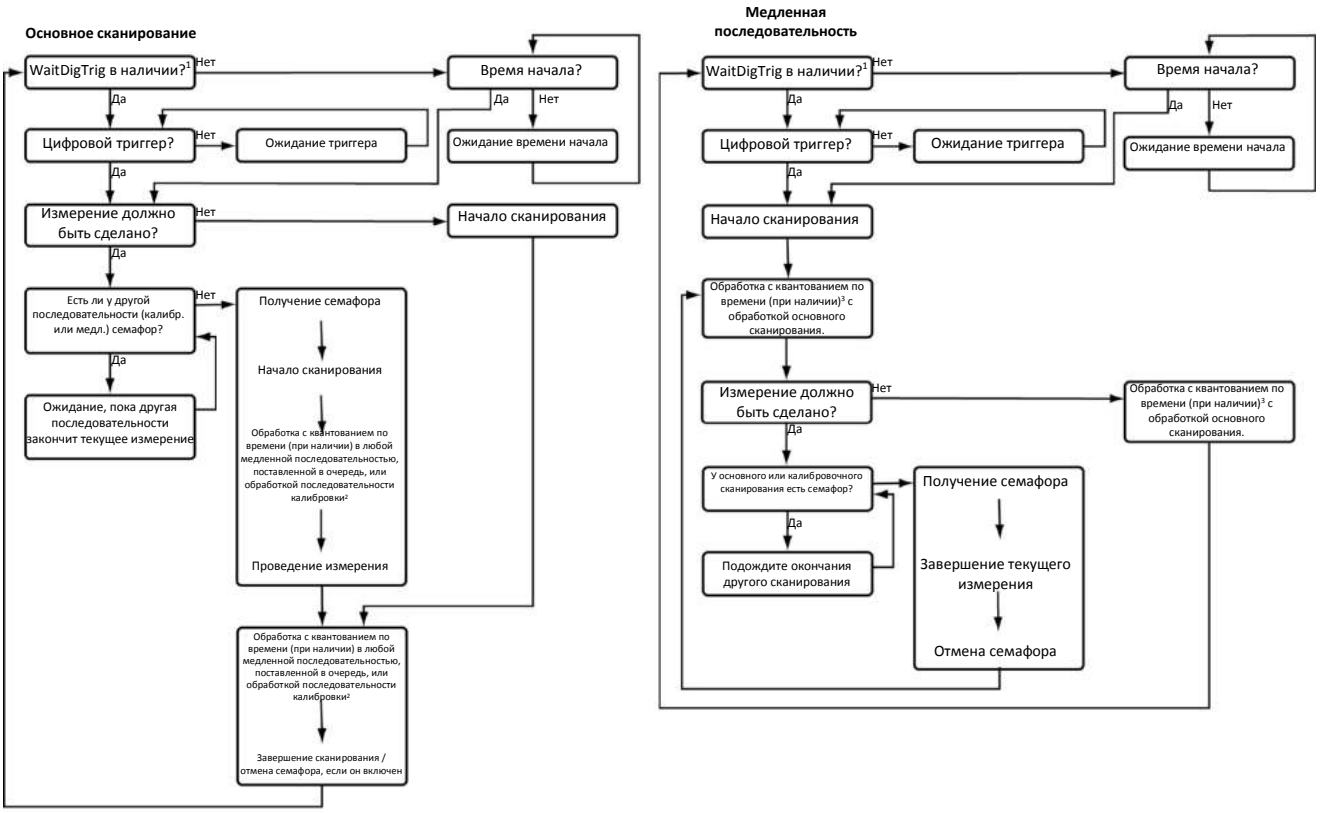
Сканирование WaitDigTrig

Читать далее. См. раздел *Синхронизация измерений (стр. 365)*.

Основные сканирования и медленные последовательности, как правило, вызываются в интервалах времени, определенных командой **Scan()**. Некоторые приложения, однако, требуют запуска сканирования в основной или медленной последовательности с помощью внешнего цифрового триггера, например, импульса 5 В пост. тока на порт управления. Команда **WaitDigTrig()** активирует программу при обнаружении внешнего триггера. **WaitDigTrig()** отдает приоритет начала сканирования, но сканирование выполняется и приобретает *семафор (стр. 527)* в соответствии с правилами, указанными в *Основное сканирование (стр. 156)* и *Сканирования в медленной последовательности (стр. 157)*. Любая обработка ведется с квантованием по времени с обработкой других последовательностей. Каждый раз, когда программа встречает команду **WaitDigTrig()**, она останавливается и ожидает запуска.

Примечание. Команда WaitDigTrig() может использоваться для того, чтобы запрограммировать CR1000 для управления другим CR1000.

Рисунок 42. Функциональная схема приоритета сканирования в последовательном режиме



1 — Программа с WaitDigTrig() сразу после Scan().

2 — Обработка с квантованием по времени (при наличии) с медленной последовательностью только при отсутствии измерений в основном сканировании.

3 — Обработка с квантованием по времени (при наличии) с основным сканированием, если в основном сканировании отсутствуют измерения, иначе квантование повремени с полным основным сканированием.

7.8.4.13. Команды программирования

В дополнение к основному синтаксису CRBasic включает дополнительные команды для облегчения измерений и сохранения данных. В разделе *Команды программирования CRBasic* *(стр. 537)* приведен исчерпывающий перечень этих команд.

7.8.4.13.1. Измерение и обработка сохраненных данных

Команды CRBasic созданы для проведения измерений и хранения данных. Команды для проведения измерений позволяют оборудованию CR1000 выполнять измерения и сохранять полученные результаты в виде переменных. Команды для сохранения данных обрабатывают измерения для получения среднего, максимального, минимального значений, стандартное отклонения, FFT и т. д.

Каждая команда представляет собой ключевое слово, за которым следует серия информационных параметров, необходимых для завершения процедуры. Например, команда для измерения температуры панели CR1000:

```
PanelTemp(Dest,Integ)
```

PanelTemp является ключевым словом. Два параметра: ***Dest*** имя назначенной переменной, в которой хранится значение температуры; и ***Integ*** промежуток времени на интеграцию измерения. Для того, чтобы поместить измерение температуры панели в переменную ***RefTemp***, используя время интеграции 250 мкс, используется код, показанный в примере CRBasic *Синтаксис команды для проведения измерений* *(стр. 159)*.

CRBasic: пример 15. Синтаксис команды для проведения измерений

```
 В Этом примере программы показано использование команды однократного измерения. В этом
 случае программа измеряет температуру коммутационной панели CR1000.

Public RefTemp Объявление переменной для получения команды.

BeginProg
  Scan(1,Sec,3,0)
    PanelTemp(RefTemp, 250) <<<<<<Команда для выполнения измерения
  NextScan
EndProg
```

7.8.4.13.2. Типы аргументов

Большинство CRBasic команд или команд имеют субкоманды или параметры. Параметры заполняются программистом с помощью аргументов. Многие команды имеют параметры, которые позволяют использовать различные типы аргументов. Далее перечислены наиболее распространенные типы аргументов. Разрешенные типы аргументов указаны в описании каждой команды в *Справке к редактору CRBasic*.

- константа или выражение, которое считается константой;
- переменная;
- переменная или массив;
- константа, переменная или выражение;
- константа, переменная, массив или выражение;
- название;
- имя или список имен;
- переменная или выражение;
- переменная, массив или выражение.

7.8.4.13.3. Имена в качестве аргументов

В таблице *Правила для имен* *(стр. 159)* приведены максимальные значения длины и допустимые символы для имен переменных, массивов, констант и т. д. Компилятор предварительного прохода *Редактор CRBasic* определяет слишком длинные или неправильно отформатированные имена.

Внимание! Использование в именах символов, не указанных в таблице *Правила для имен*, зависит от конкретной операционной системы. Однако они могут не поддерживаться в будущих операционных систем.

Таблица 20. Правила для имен

Категория имени[1]	***Максимальная длина (количество символов)***	***Допустимые символы***
Переменная или массив	39	Буквы от A до Z, от a до z, (подчеркивание) и цифры от 0 до 9. Имена должны начинаться с буквы или подчеркивания. CRBasic не чувствителен к регистру букв.
Постоянная	38	

Таблица 20. Правила для имен		
Категория имени[1]	***Максимальная длина (количество символов)***	***Допустимые символы***
Единица измерения	38	
Псевдоним	39	
Имя устройства	64	
Имя таблицы данных	20	
Имя поля	39	
Описание имени поля	64	

[1] Переменные, константы, единицы измерения, псевдонимы, названия устройств, имена полей, имена таблиц данных, а также имена файлов используют одинаковые имена; то есть, после использования имени, оно резервируется только в этой категории. Дополнительную информацию об ограничениях для имен см. в разделе *Предварительно определенные константы* *(стр. 138)*.

7.8.4.14. Выражения в качестве аргументов

Читать далее. Для получения дополнительной информации о выражениях см. раздел *Типы программируемых выражений* *(стр. 160)***.**

Многие параметры команд CRBasic позволяют ввод аргументов в виде выражений. Если выражение является сравнением, оно возвращает значение **–1**, если оно истинно, и **0** если ложно. (См. раздел *Логические выражения* *(стр. 164)*). Следующий пример кода демонстрирует использование выражений в качестве аргумента в параметре ***TrigVar*** команды **DataTable()**:

```
  Таблица данных (имя, TrigVar, размер)
DataTable(Temp, TC > 100, 5000)
```

Когда триггер составляет ***TC > 100*** , при температуре термопары выше 100 триггеру присваивается значение ***Истина,*** и данные сохраняются.

7.8.4.15. Типы программируемых выражений

Выражение представляет собой последовательность слов, операторов и чисел, которая дает значение или результат. Выражения вычисляются слева направо, с учетом правил старшинства. Результатом каждого этапа оценки является тип Long (целое, 32 бита), если переменные типа Long (константы являются целыми числами) и функции дают целочисленные результаты, например, как это происходит в команде **INTDV().** Если часть уравнения содержи переменную с плавающей запятой, константу (24 бит) или функцию, результатом которой становится значение с плавающей точкой, остальная часть выражения вычисляется с помощью чисел с плавающей точкой и 24-битной математики, даже если итоговая функция преобразует результат в целое число, это может привести к потере точности; например, **INT((rtYear-1993)*.25).** Это важная особенность, на которую стоит обратить внимание в следующих случаях: 1) попытка использовать математику целых чисел для сохранения численного разрешения вне ограничений для переменных с плавающей запятой, 2) если результат должен быть проверен на эквивалентность в отношении другого значения. Ограничения указаны в разделе *Арифметика чисел с плавающей запятой* *(стр. 161)*.

В CRBasic используется два типа выражений: математические и программируемые. Полезное свойство выражений в CRBasic заключается в том, что они эквивалентны и часто взаимозаменяемы с результатами.

Рассмотрим выражения:

```
x = (z * 1.8) + 32 (математическое выражение)
If x = 23 then y = 5(программируемое выражение)
```

Переменная x может быть опущена, а выражения объединены и записаны в виде:

```
If (z * 1.8 + 32 = 23) then y = 5
```

Замена результата выражением должна выполняться обоснованно с осознанием того, что это может значительно усложнить чтение программного кода.

7.8.4.15.1. Арифметика чисел с плавающей запятой

Переменные и вычисления выполняются внутри с одинарной точностью для четырбайтного значения с плавающей запятой IEEE, при этом вычисление некоторых операции проводится с двойной точностью.

Примечание. Переменная с плавающей запятой и одинарной точностью имеет 24 бита дробной части. Двойная точность для 32-битного расширение дробной части приводит к точности 56 бит. Команды, которые используют двойной точностью: **AddPrecise(), Average(), AvgRun(), AvgSpa(), CovSpa(), MovePrecise(), RMSSpa(), StdDev(), StdDevSpa(), Totalize()** и **TotRun().**

Арифметика чисел с плавающей запятой является обычным явлением во многих электронных и вычислительных системах, но у нее есть определенные подводные камни, о которых должны знать программисты высокого уровня. В нескольких источниках арифметика чисел с плавающей запятой обсуждается достаточно подробно. Одним из таких источников является тема *Число с плавающей точкой* на сайте в www.wikipedia.org. Поэтому программисты CR1000 должны, по меньшей мере, учесть следующее:

- Числа с плавающей точкой не идеально имитируют реальные числа.
- Арифметика чисел с плавающей запятой не всегда идеально имитирует истинную арифметику.
- Следует избегать использования равенства условных операторов. Вместо этого желательно использовать >= и <=. Например, используйте команды **If X >= Y then do** вместо **If X = Y then do.**
- При программировании суммирования нецелых чисел по длинному циклу используйте команду **AddPrecise()**. В противном случае при увеличении размера дробные слагаемые будут иметь все меньше влияние на величину суммы, так как обычные числа с плавающей запятой ограничены примерно 7 цифрами после запятой.

7.8.4.15.2. Математические операции

Математические операции описаны достаточно подробно, поскольку они выполняются в алгебраической форме. Например, для преобразования температуры в градусах по Цельсию в градусы по Фаренгейту, синтаксическая конструкция может быть следующей:

```
TempF = TempC * 1.8 + 32
```

Читать далее. Пространство кода можно сохранить при заполнении массива или частичного массива теми же значениями. Описание того, как это делается в CRBasic, приведено в примере *Использование команды Move()* для экономии пространства кода. В примере CRBasic *Использование массивов переменных для экономии пространства кода* *(стр. 162)* показан пример кода для преобразования двадцати показаний температуры в виде массива переменных из °C в °F.

CRBasic: пример 16. Использование команды Move() для экономии пространства кода

```
Move(counter(1),6,0,1)                  Сброс шести счетчиков до нуля. Удержание массива
                                        заполненным десятью самых последними показаниями.
Move(TempC(2),9,TempC(1),9)             Сдвиг предыдущих девяти показаний, чтобы освободить
                                        место для нового измерения.

 Новое измерение:
TCDiff(TempC(1),1,mV2_5C,8,TypeT,PTemp,True,0,_60Hz,1.0,0)
```

CRBasic: пример 17. Использование массивов переменных для экономии пространства кода

```
For I = 1 to 20
   TCTemp(I) = TCTemp(I) * 1.8 + 32
Next I
```

7.8.4.15.3. Выражения с типами числовых данных

Переменные типа FLOAT, **LONG** и **Boolean** можно преобразовать в другие типы данных, например **FP2**, с помощью '='.

Получение логической переменной из типа FLOAT или LONG

Когда тип **FLOAT** или **LONG** преобразуется в **Boolean**, как показано на примере CRBasic *Преобразование переменной FLOAT/LONG в логическое значение* *(стр. 162)*, ноль приобретает значение ложь (**0**), отличные от нуля значения истинными (**–1**).

CRBasic: пример 18. Конвертация переменной FLOAT/LONG в логическое значение

```
 В этом примере программы показано преобразование типов данных Long в логически
 тип данных.

Public Fa As Float
Public Fb As Float
Public L As Long
Public Ba As Boolean
Public Bb As Boolean
Public Bc As Boolean

BeginProg
  Fa = 0
  Fb = 0.125
  L = 126
  Ba = Fa                   Это позволит установить Ba = False (0)
  Bb = Fb                   Это позволит установить Bb = True (-1)
  Bc = L                    Это позволит установить Bc = True (-1)
EndProg
```

Получение переменной FLOAT из типа LONG или логического типа

Когда тип **LONG** или **Boolean** преобразуется в **FLOAT**, целое значение загружается в **FLOAT**. Логические переменные преобразуются в **–1** или **0**. Целые числа типа **LONG** больше 24 бит (16 777 215; размер дробной части для **FLOAT**) теряют разрешение при преобразовании в тип **FLOAT**.

Получение переменной LONG из типа FLOAT или логического типа

При преобразовании в тип **Long или логический** присваивается значение **–1** или **0**. Когда переменная **FLOAT** преобразуется в **LONG**, она обрезается. Это преобразование проводится аналогично функции **INT** (*Арифметические функции* *(стр. 568)*). Преобразование ведется в целое число, которое равно или меньшее значения Float; например, **4,6** преобразуется в **4**, а **–4,6** — **–5**).

Если значение **FLOAT** больше предельно допустимого значения для **LONG** (+2 147 483 647), целому числу присваивается максимальное значение. Если значение **FLOAT** меньше предельно допустимого значения для **LONG** (–2 147 483 648), целому числу присваивается минимальное значение.

Целые числа в выражениях

Переменные LONG, когда это возможно, рассматриваются в выражениях, как целые числа. В примере CRBasic *Оценка целых чисел (стр. 163)* показано присвоение целым числам типа **LONG** или **FLOAT**.

CRBasic: пример 19. Оценка целых чисел

```
' В этом примере программы показана оценка целых чисел.

Public I As Long
Public X As Float

BeginProg
  I = 126
  X = (I+3) * 3.4
  'I+3 оценивается как целое число, а затем преобразуется в тип данных Float перед
  'умножением на 3,4.
EndProg
```

Преобразования констант

Константы не объявляются с типом данных, поэтому CR1000 присваивает им тип данных по мере необходимости. Если константа (как введенная в виде числа, так и объявленная с помощью команды **CONST**) может быть правильно выражена как целое число, компилятор будет использовать тот тип данных, который является наиболее эффективным для каждого конкретного выражения. Если это возможно, используется целочисленный вариант, например, если выражение еще не обозначено как **FLOAT**. В примере CRBasic *Преобразование констант в LONG или FLOAT (стр. 163)* указан вариант, где значение, которое считается целом числом (10) назначается CR1000 в качестве **As FLOAT.**

CRBasic: пример 20. Преобразование констант в LONG или FLOAT

```
' В этом примере программы показано преобразование констант в типов данных Long или Float.

Public L As Long
Public F1 As Float
Public F2 As Float
Const ID = 10

BeginProg
  F1 = F2 + ID
  L = ID * 5
EndProg
```

В примере CRBasic *Преобразование констант в LONG или FLOAT (стр. 163)*, **I** является целым числом. **A1** и **A2** имеют тип **FLOATS**. Цифра 5 загружается **As FLOAT**, чтобы эффективно добавить идентификатор константы, который представляет собой тип **As FLOAT** для предыдущего выражения, и чтобы избежать неэффективного преобразование из **LONG** в **FLOAT**, перед каждой плавающей запятой.

7.8.4.15.4. Логические выражения

Измерения могут указывать на отсутствие или наличие события. Например, результат измерения относительной влажности 100 % указывает на событие, связанное с конденсацией, например туман, дождь или роса. CR1000 может представить состояние события в двоичной форме для дальнейшей обработки, то есть событие или происходит (истина), или не происходит (ложь).

Истина = –1, ложь = 0

Во всех случаях аргумент **0** в логических выражениях имеет значение **FALSE** (ЛОЖЬ); если обобщить, то любое ненулевое число имеет значение «non-FALSE» (не ложь). Однако аргумент **TRUE** (ИСТИНА) в операционной системе CR1000 предварительно задан равным **–1**, поэтому только аргумент **–1** *всегда* переводится как **TRUE** (ИСТИНА). Рассмотрим выражение

```
If Condition(l) = TRUE Then...
```

Это условие является истинным, только если Условие(1) = **–1**. Если условие Condition(1) имеет любое значение, кроме «не равен нулю», то условие не будет считаться выполненным, потому что константа **TRUE** (ИСТИНА) в системной памяти CR1000 имеет предварительно заданное значение **–1**. При вводе = **TRUE** (ИСТИНА) выполняется литерное сравнение. Таким образом, для абсолютной уверенности в том, что функция верна, она должна иметь значение **TRUE** (ИСТИНА) или **–1.**

Примечание. TRUE (ИСТИНА) составляет **–1**, поэтому каждому биту присваивается высокое значение (–1 это &B 11111111 для всех четырех байтов). Это позволяет правильно выполнить операцию **AND** (И). Операция **AND** (И) выполняет логическую функцию AND (И) для каждого бита, поэтому значение выражения **TRUE AND X** (ИСТИНА И X) будет не равно нулю, если хотя бы один из битов в X отличен от нуля (если X не равен нулю). Когда переменной типа BOOLEAN (ЛОГИЧЕСКИЙ) присваивается любое ненулевое число, CR1000 внутренне преобразует его в **–1.**

CR1000 способен преобразовывать условия, указанные в таблице *Двоичные условия ИСТИНА и ЛОЖЬ* *(стр. 164)* в двоичную форму (–1 или 0), используя перечисленные команды, и сохранить в двоичной форме в указанную ячейку памяти. В таблице *Примеры логических выражений* *(стр. 165)* показаны некоторые логические выражения.

Non-Zero (Не равно нулю) = True (Sometimes) (Истина (иногда))

Любой аргумент, кроме ***0*** и ***–1,*** будет преобразован в некоторых случаях как ***TRUE*** (ИСТИНА), а в остальных случаях — как ***FALSE*** (ЛОЖЬ). Однако использование только ***–1*** как числового представления значения ***TRUE*** (ИСТИНА) может быть лучшим решением. Рассмотрим выражение

```
If Condition(l) then...
```

Поскольку = **True** (Истина) в выражении пропущено, условие **Condition(1)** считается истинным, если оно равно любому ненулевому значению.

Таблица 21. Двоичные условия TRUE (ИСТИНА) и FALSE (ЛОЖЬ)		
Условие	***Используется команда CRBasic***	***Размещение двоичного результата в памяти***
Время	**TimeIntoInterval()**	Переменная, система
	IfTime()	Переменная, система
	TimeIsBetween()	Переменная, система
Триггер порта управления	**WaitDigTrig()**	Система
Коммуникации	**VoiceBeg()**	Система
	ComPortIsActive()	Переменная
	PPPClose()	Переменная
Измерение	**DataEvent()**	Система

При использовании условий TRUE (ИСТИНА) или FALSE (ЛОЖЬ) с логическими операторами, например AND (И) и OR (ИЛИ), можно кодировать логические выражения для выполнения одной из следующих трех общих логических функций. Это облегчает условную обработку и управление приложениями:

1. Определите выражение, указав один путь или действие, если выражение истинно (= –1), и/или другой путь или действие, если выражение ложно (= 0).
2. Определите множественные выражения, связанные с операторами **AND** (И) или **OR** (ИЛИ).
3. Определите множественные ссылки **AND** (И) или **OR** (ИЛИ).

Для построения логических выражений используются следующие команды и логические операторы. В примере CRBasic *Примеры логических выражений (стр. 165)* показаны некоторые логические выражения.

- IF (ЕСЛИ);
- AND
- OR
- NOT
- XOR
- IMP
- IIF

Таблица 22. Примеры логических выражений
`If X >= 5 then Y = 0` Устанавливает переменную Y равной 0, если выражение «X> =5» истинно, то есть, если X больше или равно 5. CR1000 оценивает выражение (X > = 5) и вносит в системную память значение **–1**, если выражение истинно, или **0**, если выражение ложно.
`If X >= 5 OR Z = 2 then Y = 0` Устанавливает Y = 0, если одно из выражений X >= 5 или Z = 2 является истинным.
`If X >= 5 AND Z = 2 then Y = 0` Устанавливает Y = 0 только если оба выражения X >= 5 и Z = 2 являются истинными.
`If 6 then Y = 0.` **Поскольку 6** является истинным, так как возвращается число **6** (которое на равно нолю), Y присваивается значение **0** при каждом выполнении оператора.
`If 0 then Y = 0.` **Поскольку 0** является ложным, так как возвращается число **0**, этот оператор никогда не присвоит Y значение **0**.
`Z = (X > Y).` Z равно **–1**, если X > Y, или Z равно **0**, если X <= Y.

Таблица 22. Примеры логических выражений

Оператор **NOT** (НЕ) дополняет каждый бит в слове. Логическое значение может быть FALSE (ЛОЖЬ) (0 или все биты 0) или TRUE (ИСТИНА) (–1 или все биты 1). «Дополненное» логическое значение превращает TRUE (ИСТИНА) в FALSE (ЛОЖЬ) (все биты дополняются до 0).

Пример программы

```
'(a AND b) = (26 AND 26) = (&b11010 AND &b11010) =
'&b11010. NOT (&b11010) yields &b00101.

'Значение не равно нулю, поэтому при преобразовании в
'ЛОГИЧЕСКИЙ формат получает значение TRUE (ИСТИНА).
Public a As LONG
Public b As LONG
Public is_true As Boolean
Public not_is_true As Boolean
Public not_a_and_b As Boolean
BeginProg
  a = 26
  b = a
  Scan (1,Sec,0,0)
    is_true = a AND b               'Устанавливает значение TRUE (ИСТИНА).
    not_is_true = NOT (is_true)     'Устанавливает значение (ЛОЖЬ).
    not_a_and_b = NOT (a AND b)     'Устанавливает значение TRUE (ИСТИНА)!
  NextScan
EndProg
```

7.8.4.15.5. Строковые выражения

CRBasic облегчает конкатенацию строковых переменных в переменных всех типов данных с помощью операторов **&** и + . Для получения стабильных результатов при конкатенации строк используйте **&**. Используйте + при конкатенации строк в другие типы переменных. В примере CRBasic *Конкатенация строки и переменной* *(стр. 166)* показан код CRBasic код для конкатенации строк и целых чисел. Для получения дополнительной информации о программировании строки см. раздел *Строковые операции* *(стр. 282)* в *Библиотеке программных ресурсов* *(стр. 169)*.

CRBasic: пример 21. Конкатенация строк и переменных

```
'В этом примере программы показана конкатенация переменных, объявленных As String в
'другие строки, и переменных, объявленных другими типами данных.
'
'Объявление переменных
Dim PhraseNum(2) As Long
Dim Word(15) As String * 10
Public Phrase(2) As String * 80

'Объявление таблицы данных
DataTable(HAL, 1, -1)
   DataInterval(0,15,Sec,10)

   'Запись фразы в таблицу данных «Тест»
  Sample(2, Phrase, String)
EndTable
```

```
'Программа
BeginProg
  Scan(1,Sec,0,0)

    'Определение строк в качестве строковых переменных
    Word(1) = "Good"
    Word(2) = "morning"
    Word(3) = "Dave"
    Word(4) = "I'm"
    Word(5) = "sorry"
    Word(6) = "afraid"
    Word(7) = "I"
    Word(8) = "can't"
    Word(9) = "do"
    Word(10) = "that"
    Word(11) = " "
    Word(12) = ","
    Word(13) = ";"
    Word(14) = "."
    Word(15) = Chr(34)

    'Определение целых чисел в качестве переменных типа Long
    PhraseNum(1) = 1
    PhraseNum(2) = 2

    'Конкатенация строки «1. Доброе утро, Дейв».
    Phrase(1) = PhraseNum(1)+Word(14)+Word(11)&Word(15)&Word(1)&Word(11)&Word(2)& _
                Word(12)&Word(11)&Word(3)&Word (14)&Word(15)

    'Конкатенация строки «2. К сожалению, Дейв, я не могу это сделать».
    Phrase(2) = PhraseNum(2)+Word(14)&Word(11)&Word(15)&Word(4)&Word(11)&Word(6)&Word(11)& _
                Word(7)&Word(11)&Word(8)&Word(11)&Word(9)&Word(11)&Word(10)&Word(12)& _
                Word(11)&Word(3)&Word(14)&Word (15)

    CallTable HAL

  NextScan
EndProg
```

7.8.4.16. Программирование доступа к таблицам данных

Таблица данных представляет собой область памяти в которой хранятся записи данных. Иногда в программе CRBasic необходимо использовать сохраненные данные. Например, программа может быть предназначена для получения средней температуры за последние пять дней для дальнейшей обработки. CRBasic содержит синтаксические выражения, которые облегчают доступ к этим данным таблицы или мета-данным, относящимся к таблице данных. За исключением случаев использования команды **GetRecord()** (*Доступ и управление таблицами данных* *(стр. 592)*), синтаксическое выражение вводится непосредственно в программу CRBasic с помощью имени переменной. Общая форма:

```
TableName.Fie1dName_Prc(Fie1dname Index, Records Back)
```

где

- **TableName** (Имя таблицы данных) — имя таблицы данных.
- **FieldName** (Имя поля) — имя переменной, из которого получают обработанное значение.
- **Prc** — сокращение имени процесса, который использует данные. Полный список сокращений см. в таблице *Сокращения обработки данных* *(стр. 168)*. Для этого не требуются значения из таблиц **Status** (Состояние) или **Public** (Общая).

- **Fieldname Index** (Индекс имени поля) — номер элемента массива в полях, которые относятся к массивам (дополнительно).
- **Records Back** (Возврат записей) — как далеко может зайти возврат в таблицу, чтобы получить значение (дополнительно). Если оставить поле пустым, используется самая последняя запись.

Таблица 23. Сокращения обработки данных

Сокращение	*Название процесса*
Tot	Суммировать
Средн.	Среднее
Макс.	Максимум
Min	Минимум
SMM	Значение мин. или макс.
Std	Стандартное отклонение
MMT	Момент
Нет аббревиатура	Образец
Hst	Гистограмма[1]
H4D	Histogram4D
FFT	FFT
Cov	Ковариантa
RFH	«Дождевая» гистограмма
LCr	Пересечение уровня
WVc	Вектор ветра
Med	Медиана
ETsz	ET
RSo	Солнечная радиация (от ET)
TMx	Макс. время
TMn	Мин. время

[1] **Hst** передается в виде **Hst,20,1.0000e+00,0.0000e+00,1.0000e+01**, где **Hst** обозначает гистограмма, 20 = 20 двоичных значений, 1 = весовой коэффициент, 0 = нижняя граница, 10 = верхняя граница.

Например, для доступа к количеству ошибок схемы безопасности, используйте оператор

```
wderr = status.watchdogerrors
```

где **wderr** — объявленная переменная, **status** — имя таблицы, а **watchdogerrors** — ключевое слово для поля ошибки схемы безопасности.

Для доступа к информации о таблице используется семь специальных имен переменных.

- **EventCount;**
- **EventEnd;**
- **Output;**
- **Record;**
- **TableFull;**
- **TableSize;**
- **TimeStamp.**

Полное описание см. в *Справке к редактору CRBasic* в теме *Доступ к таблице данных)*.

Таблица **DataTableInformation** (Информация о таблице данных) также содержит эту информацию. См. *Состояние, настройки и информация о таблицах данных (Status/Settings/DTI)* *(стр. 603)*.

7.8.4.17. Программирование использования сигнатур

Сигнатуры помогают обеспечить целостность и безопасность системы. Информация об использовании сигнатур содержится на следующих ресурсах.

- Команда **Signature()** (Сигнатура) в меню *Diagnostics* (Диагностика) *(стр. 550)*.
- Запись **RunSignature** (Рабочая сигнатура) в таблице *Параметры состояния/настроек/таблиц* *(стр. 603)*.
- Запись **ProgSignature** (Сигнатура программы) в таблице *Параметры состояния/настроек/таблиц* *(стр. 603)*.
- Запись **OSSignature** (Сигнатура ОС) в таблице *Параметры состояния/настроек/таблиц* *(стр. 603)*.
- *Безопасность* *(стр. 92)*.

Многие сигнатуры фиксируются в таблице **Status** (Состояние), которая содержит тип данных таблицы. Сигнатуры, записанные в таблице **Status** (Состояние), можно скопировать в переменную с помощью метода, описанного в разделе *Программирование доступа к таблицам данных* *(стр. 167)*. После формирования переменной можно выбрать сигнатуры как часть другой таблицы данных для архивирования.

7.9. Библиотека программных ресурсов

Эта библиотека комментариев и кода CRBasic для выбора определенного узкого круга приложения CR1000. Если вам потребуется дополнительная информация, свяжитесь со специалистом по применению Campbell Scientific.

7.9.1. Дополнительные методы программирования

7.9.1.1. Считывание событий

В примере CRBasic *Считывание событий* *(стр. 169)* показано программирование вывода данных в таблицу при наступлении события.

CRBasic: пример 22. Синтаксис команд BeginProg /Scan() /NextScan /EndProg

```
'В этом примере программы показаны обнаружение и запись события. Событие имеет
'начало и конец. Эта программа записывает события в момент их окончания.
'Записываемое событие — разность температур превышает 3 градуса. Событие
'записывается, когда разность температур становится меньше 3 градусов.

'Команда DataEvent делает запись о событии в таблице данных после
'каждого завершения события. Количество событий записывается в зарезервированную переменную
'EventCount(1,1). В этой программе, EventCount(1,1)записывается в таблицу
'OneMinute.

'Примечание. Команда DataEvent должна использоваться в таблице данных с
'меньшим интервалом записи, чем ожидаемая частота события.

'Объявление переменных
Public PTemp_C, AirTemp_C, De1taT_C
Public EventCounter
```

```
 Объявление таблицы данных, управляемой событием
DataTable(Event,True,1000)
  DataEvent(0,DeltaT_C>=3,DeltaT_C<3,0)
  Sample(1,PTemp_C, FP2)
  Sample(1,AirTemp_C, FP2)
  Sample(1,DeltaT_C, FP2)
EndTable

 Объявление таблицы данных, управляемой временем
DataTable(OneMi n,True,-1)
  DataInterval(0,1,Min,10)
  Sample(1,EventCounter, FP2)
EndTable

BeginProg
  Scan(1,Sec,1,0)

     Температура коммутационной панели
     PanelTemp(PTemp_C,_60Hz)

     Измерения с помощью термопары типа T:
     TCDiff(AirTemp_C,1,mV2_5C,1,TypeT,PTemp_C,True,0,_60Hz,1,0)

     Расчет разницы между температурой воздуха и панели
     DeltaT_C = AirTemp_C - PTemp_C

     Обновление счетчика событий (используется специальный синтаксис команды
     Event.EventCount(1,1))
     EventCounter = Event.EventCount(1,1)

     Вызов таблицы данных
     CallTable(Event)
     CallTable(OneMin)

  NextScan
EndProg
```

7.9.1.2. Условный вывод

В примере CRBasic *Условный вывод* *(стр. 170)* показана программа для вывода данных в таблицу, с условием срабатывания, которое не относится ко времени.

CRBasic: пример 23. Условный вывод

```
 В этом примере программы показана условная запись данных в таблицу. Также
 здесь показано использование команд StationName() (Имя устройства) и Units (Единицы измерения).

 Объявление имени устройства (сохраняется в таблице состояния)
StationName(Delta_Temp_Station)

 Объявление переменных
Public PTemp_C, AirTemp_C, DeltaT_C
```

```
 Объявление единицы измерения
Units PTemp_C = deg C
Units AirTemp_C = deg C
Units De1taT_C = deg C

 Объявление таблицы вывода — Условный вывод при Delta T> = 3.
 Таблица сохраняет данные со скоростью сканирования (один раз в секунду), если условие
 выполняется,
  поскольку команда DataInterval (Интервал данных) не включен в объявление таблицы.
DataTable(De1taT,De1taT_C >= 3,-1)
  Sample(1,Status.StationName,String)
  Sample(1,DeltaT_C,FP2)
  Sample(1,PTemp_C,FP2)
  Sample(1,Ai rTemp_C,FP2)
EndTable

BeginProg
  Scan(1,Sec,1,0)
     Измерение температуры коммутационной панели
    PanelTemp(PTemp_C,_60Hz)

     Термопара типа T.
    TCDiff(AirTemp_C,1,mV2_5C,1,TypeT,PTemp_C,True,0, _60Hz,1,0)

     Расчет разницы между температурой воздуха и панели
    DeltaT_C = AirTemp_C - PTemp_C

     Вызов таблицы данных
    CallTable(DeltaT)

  NextScan
EndProg
```

7.9.1.3. Тестирование насоса для грунтовых вод

В примере CRBasic *Тестирование насоса для грунтовых вод* *(стр. 171)* описано:

- Как создать несколько интервалов в одной таблице данных.
- Использование команд контроля программы вне структуры **(Scan)/ NextScan.**
- Один из способов выполнения условного кода.
- Использование нескольких последовательных сканирований, каждое со счетчиком сканирования.

CRBasic: пример 24. Тестирование насоса для грунтовых вод

```
' В этом примере программы показано использование нескольких циклов сканирования в программе для
' запуска насоса для грунтовых вод. Обратите внимание, что в данном примере для команды Scan()
' единицы времени были изменены с секунд на миллисекунды, это позволяет выполнить программу за
' короткий промежуток времени. Для того, чтобы использовать эту функцию программы для реального
' испытания насоса, измените аргумент единиц измерения для команды Scan() с миллисекунд на
' секунды. Также необходимо указать уровень измерения в подпрограмме MeasureLevel.

' Испытание насоса для грунтовых вод требует измерения и записи уровня воды,
' согласно следующему графику:

' Количество минут      Интервал вывода данных
'-----------------      ----------------
' 0-10                  10 секунд
' 10-30                 30 секунд
' 30-100                60 секунд
' 100-300               120 секунд
' 300-1000              300 секунд
' 1000+                 600 секунд

' Объявление переменных
Public PTemp
Public Batt_Volt
Public Level
Public LevelMeasureCount As Long
Public ScanCounter(6) As Long

' Объявление таблицы данных
DataTable(LogTable, 1, -1)
  Minimum(1,Batt_Volt,FP2,0,False)
  Sample(1,PTemp,FP2)
  Sample(1,Level,FP2)
EndTable

' Объявление подпрограммы измерения уровня
Sub MeasureLevel
  LevelMeasureCount = LevelMeasureCount + 1 'Included to show passes through sub-routine
  Здесь указываются команды для измерения уровня.
EndSub

' Основная программа
BeginProg

  Минуты испытания с 0 по 10: 10-секундный выходной интервал.
  Scan(10,mSec,0,60) 'There are 60 10-second scans in 10 minutes
    ScanCounter(1) = ScanCounter(1) + 1 'Included to show passes through this scan
    Battery(Batt_volt)
    PanelTemp(PTemp,250)
    Call MeasureLevel

    Формируем таблицы выходных данных
    CallTable LogTable
NextScan
```

```
  Минуты испытания с 10 по 30: 30-секундный выходной интервал.
  Scan(30,mSec,0,40) 40 30-секундных сканирований за 20 минут.
     ScanCounter(2) = ScanCounter(2) + 1 Включено для демонстрации процесса сканирования.
     Battery(Batt_volt)
     PanelTemp(PTemp,250)
     Call MeasureLevel

     Формируем таблицы выходных данных
     CallTable LogTable
  NextScan

  Минуты испытания с 30 по 100: 60-секундный выходной интервал.
  Scan(60,mSec,0,70) 70 60-секундных сканирований за 70 минут.
     ScanCounter(3) = ScanCounter(3) + 1 Включено для демонстрации процесса сканирования.
     Battery(Batt_vo1t)
     PanelTemp(PTemp,250)
     Call MeasureLevel

     Формируем таблицы выходных данных
     CallTable LogTable
  NextScan

  Минуты испытания с 100 по 300: 120-секундный выходной интервал.
  Scan(120,mSec,0,200) 200 120-секундных сканирований за 10 минут.
     ScanCounter(4) = ScanCounter(4) + 1 Включено для демонстрации процесса сканирования.
     Battery(Batt_volt)
     PanelTemp(PTemp,250)
     Call MeasureLevel

     Формируем таблицы выходных данных
     CallTable LogTable
  NextScan

  Минуты испытания с 300 по 1000: 300-секундный выходной интервал.
  Scan(300,mSec,0,140) 140 300-секундных сканирований за 700 минут.
     ScanCounter(5) = ScanCounter(5) + 1 Включено для демонстрации процесса сканирования.
     Battery(Batt_volt)
     PanelTemp(PTemp,250)
     Call MeasureLevel

     Формируем таблицы выходных данных
     CallTable LogTable
  NextScan

  Минуты испытания после 1000: 600-секундный выходной интервал.
  Scan(600,mSec,0,0) После 1000 минуты продолжается сканирование с 600-секундным интервалом в
  течение неопределенного времени.
     ScanCounter(6) = ScanCounter(6) + 1 Включено для демонстрации процесса сканирования.
     Battery(Batt_volt)
     PanelTemp(PTemp,250)
     Call MeasureLevel

     Формируем таблицы выходных данных
     CallTable LogTable
  NextScan

EndProg
```

7.9.1.4. Прочие возможности

В примере CRBasic *Прочие возможности* *(стр 174)* показано использование нескольких функций CRBasic: тип данных, единицы измерения, имена, счетчики событий, флаги, интервалы вывода и контрольные элементы.

CRBasic: пример 25. Прочие возможности программы

```
'Этот пример программы показано использование команды однократного измерения. В этом
'случае программа измеряет температуру коммутационной панели CR1000.

Public RefTemp 'Объявление переменной для получения команды.

BeginProg
  Scan(1,Sec,3,0)
    PanelTemp(RefTemp, 250) 'Команда для выполнения измерения
  NextScan
EndProg

'Программа может (и должна быть!) хорошо документирована. Любой текст предваряемый
'апострофом игнорируется компилятором CRBasic.

'Одна термопара проводит измерение дважды, используя температуру коммутационной панели в качестве
'эталона
'температуры. Первое измерение проводится в градусах Цельсия, второе — в градусах по Фаренгейту.
'Затем первое измерение преобразуется из градусов Цельсия в градусы F в последующей
'строке, а результат помещаются в другую переменную. Вычисляется разница между эталонной
'температурой
'панели и первым измерением, затем она
'используется для управления состоянием флага, контролируемого программой. Затем управление
'программы
'переходит в управление устройством, поскольку состояние флага используется для определения
'состояние порта управления, который контролирует работу светодиода.

'Напряжение батареи измеряется и сохраняется только потому, что рекомендует общепринятая практика
'программирования.

'Создаются две таблицы для хранения данных. Таблица «OneMin», управляемая интервалом,
'в которой данные сохраняются каждую минуту, определяемую по часам CR1000. Таблица «Event» —
'таблица, управляемая временем, которая сохраняет данные только при соблюдении определенных
'условий.

'Объявление переменных Public (видимых).
Public Batt_Volt As FLOAT                    'Объявление в качестве типа Float
Public PTemp_C                               'Тип Float по умолчанию
Public AirTemp_C                             'Тип Float по умолчанию
Public AirTemp_F                             'Тип Float по умолчанию
Public AirTemp2_F                            'Тип Float по умолчанию
Public DeltaT_C                              'Тип Float по умолчанию
Public HowMany                               'Тип Float по умолчанию
Public Counter As Long                       'Объявление в качестве типа Long, чтобы в счетчике
                                             'отсутствовала
                                             'ошибка округления.
Public SiteName As String * 16               'Объявление в качестве строки с 16 символами для
                                             'имени сайта (необязательно).

'Объявление условий и флагов для управления программой. Установка значений «High» (Высокий) и
'«Low» (Низкий) равными «TRUE» (ИСТИНА) и «FALSE» (ЛОЖЬ) соответственно.
Public Flag(1) As Boolean
Const High = True
Const Low = False
```

```
 Дополнительно — объявление названия устройства в таблице состояния.
StationName(CR1000_on_desk)

 Дополнительно — объявление единиц измерения. Единицы измерения не используются в
 программировании, и указываются только в заголовке файла данных.
Units Batt_Volt = Volts
Units PTemp = deg C
Units AirTemp = deg C
Units AirTempF2 = deg F
Units DeltaT_C = deg C

 Объявление таблицы, управляемой выходным интервалом.
DataTable(OneMin,True,-1)                    Сохранение данных в зависимости от времени
  DataInterval(0,1,Min,0)                    контролирует интервал.
  Average(1,AirTemp_C,IEEE4,0)               Сохраняет среднюю температура в высоком
                                               формате разрешения.
  Maximum(1,AirTemp_C,IEEE4,0,False)         Сохраняет максимальную температуру в высоком
                                               формате разрешения.
  Minimum(1,AirTemp_C,FP2,0,False)           Сохраняет минимальную температуру в низком
                                               формате разрешения.
  Minimum(1,Batt_Volt,FP2,0,False)           Сохраняет минимальное напряжение батареи в низком
                                               формате разрешения.
  Sample(1,Counter,Long)                     Сохраняет счетчик в целочисленном формате.
  Sample(1, SiteName, String)                Сохраняет название сайта в виде строки.
  Sample(1,HowMany, FP2)                       Сохраняет сколько событий данных находится в
                                               низком формате разрешения.

EndTable

 'Declare an event driven data output table
  DataTable(Event,True,1000)                   Таблица данных — управляемая событием
  DataInterval(0,5,Sec,10)                     -AND (И) управляемая интервалом.
  DataEvent(0,DeltaT_C >= 3,DeltaT_C < 3,0)    -AND (И) управляемая диапазоном события.
  Maximum(1,AirTemp_C,FP2,0,False)             Сохраняет максимальную температуру в низком
                                                 формате разрешения.
  Minimum(1,AirTemp_C,FP2,0,False)             Сохраняет минимальную температуру в низком
                                                 формате разрешения.
  Sample(1,DeltaT_C, FP2)                      Сохраняет разницу температур в низком
                                                 формате разрешения.
  Sample(1,HowMany, FP2)                       Сохраняет сколько событий данных находится в
                                                 низком формате разрешения.
EndTable

BeginProg

 Второй способ присвоить название устройству заключается в загрузке имени в строчную переменную.
 Оно размещается здесь, и поскольку выполняется только один раз, экономит немного времени
 выполнения программы.

SiteName = "CR1000SiteName"
```

```
  Scan(1,Sec,1,0)

    Измерения

    Напряжение батареи
  Battery(Batt_Volt)

  Температура коммутационной панели
  PanelTemp(PTemp_C,_60Hz)

  Измерения с помощью термопары типа T:
  TCDiff(AirTemp_C,1,mV2_5C,1,TypeT,PTemp_C,True,0,_60Hz,1,0)
  TCDiff(AirTemp_F,1,mV2_5C,1,TypeT,PTemp_C,True,0,_60Hz,1.8,32)

  Преобразование из градусов по Целью в градусы по Фаренгейту.
  AirTemp2_F = AirTemp_C * 1.8 + 32

  Подсчет количества проходов программы. Здесь показано использование
  целочисленной переменной Long в счетчиках.
  Counter = Counter + 1

  Расчет разницы между температурой воздуха и панели
  DeltaT_C = AirTemp_C - PTemp_C

  Контроль флаг на основании разницы температур. Если DeltaT>= 3, то
  задать высокое значение флага Flag 1, в противном случае — низкое.
  If DeltaT_C >= 3 Then
    Flag(1) = high
  Else
    Flag(1) = low
  EndIf

  Включить светодиод, подключенный к порту 1, когда флаг 1 имеет высокое значение.
  If Flag(1) = high Then
    PortSet(1,1)                                  Альтернативное выражение: PortSet(1,high)
  Else
    PortSet(1,0)                                  Альтернативное выражение: PortSet(1,low)
  EndIf

  Подсчитайте, сколько раз произошло событие данных №DeltaT_C>= 3». Синтаксическое выражение
  TableName.EventCount используется для возврата количества событий сохранения данных,
  которые были выполнены для таблицы, управляемой событиями. Этот пример показан в таблице
  данных «Event», которая объявлена ранее, и сообщает данные счетчика событий. После команды
  EventCount необходимо указать (1,1).
  HowMany = Event.EventCount(1,1)

  Вызов таблиц данных
  CallTable(OneMin)
  CallTable(Event)

  NextScan
EndProg
```

7.9.1.5. Команда PulseCountReset

Команда PulseCountReset используется в редких случаях, для сброса или обнуление импульсных аккумуляторов CR1000 (см. *Измерения — общие сведения* *(стр. 62)*).

Команда PulseCountReset требуется в приложениях, где две отдельные команды **PulseCount()** используются для проведения разного вида сканирования на одной клемме импульсного входа. Хотя компилятор не допускает несколько команд **PulseCount()** для сканирования через одну клемму, выполнение нескольких разных видов сканирования через одну клемму допускается. Информация **PulseCount()** не сохраняется глобально, а только для каждой отдельной команды. Таким образом, если в программе чередуется быстрое и медленное сканирование, можно задать два отдельных сканирования с определенной с логикой переключения между ними. Если команда **PulseCount()** используется в обоих видах сканирования, перед началом каждого сканирования используется команда **PulseCountReset** .

7.9.1.6. Масштабирование массива

В примере CRBasic *Масштабирование массива* *(стр. 177)* показана программа для создания и использования масштабирования массива. Несколько мультипликаторов и смещений вводятся в начале программы, а затем используются несколькими командами в разных частях программы.

CRBasic: пример 26. Масштабирование массива

```
'В этом примере программы показано использование массива с масштабированием. Массив для трех
'измеряемых температур. Первая указана в градусах по Цельсию, вторая
'в градусах по Кельвину, а третья — по Фаренгейту.

'Объявление видимых переменных.
Public PTemp_C
Public Temp_C(3)
Public Count

'Объявление масштабирования массивов, как невидимых переменных.
Dim Mult(3)
Dim Offset(3)

'Объявление выходной таблицы.
  DataTable(Min_5 ,True, -1)
  DataInterval(0,5,Min,0)
  Average(1,PTemp_C,FP2,0)
  Maximum(1,PTemp_C,FP2,0,0)
  Minimum(1,PTemp_C,FP2,0,0)
  Average(3,Temp_C(),FP2,0)
  Minimum(3,Temp_C(1),FP2,0,0)
  Maximum(3,Temp_C(1),FP2,0,0)
EndTable

'Начало программы
BeginProg

  'Загрузка масштабирования массива
  Mult(1) = 1.0 : Offset(1) = 0       'Масштабирование температуры 1-й термопары в градусы по
                                      'Цельсию.
  Mult(2) = 1.0 : Offset(2) = 273.15  'Масштабирование температуры 2-й термопары в градусы по
                                      'Кельвину.
  Mult(3) = 1.8 : Offset(3) = 32      'Масштабирование температуры 3-й термопары в градусы по
                                      'Фаренгейту.
```

```
  Scan(5,Sec,1,0)

    Эталонная температура измерения
    PanelTemp(PTemp_C,_60Hz)

    Измерение тремя термопарами и масштабирование всех полученных результатов. Коэффициенты
    масштабирования массива
    применяются для каждого измерения, так как синтаксис использует аргумент 3 для параметра
    Reps
    команды TCDiff(), а масштабирование массивов переменных в качестве аргументов для
    параметров Multiplier (Множитель) и Offset (Смещение).
    TCDiff(Temp_C(), 3, mV2_5C,1,TypeT,PTemp_C,True,0,250,Mult(),Offset())

    CallTable(Min_5)

  NextScan
EndProg
```

7.9.1.7. Сигнатуры: примеры программ

Сигнатура программы представляет собой уникальное целое число, которое вычисляется на основании всех символов в заданном наборе кода. При изменении кода меняется и сигнатура. Включение данных сигнатур в наборы данных CR1000 позволяет системным администраторам отслеживать изменения программ и обеспечивать качество данных. Доступны следующие сигнатуры программы.

- текстовая сигнатура;
- двоичная сигнатура времени выполнения;
- сигнатуры исполняемого кода.

7.9.1.7.1. Текстовая сигнатура

Текстовая сигнатура является наиболее распространенным видом сигнатур для программы. Эта сигнатура вычисляется по всему тексту программы, включая пустые строки и комментарии. Сигнатура текста программы содержится в таблице **Status** (Состояние) как переменная ***ProgSignature.*** См. пример CRBasic *Сигнатуры программы* *(стр. 178)***.**

7.9.1.7.2. Сигнатура времени выполнения

Двоичная сигнатура времени выполнения вычисляется на основе только программного кода. В нее не входят комментарии или пустые строки. См. пример CRBasic *Сигнатура программы* *(стр. 178)***.**

7.9.1.7.3. Сигнатуры исполняемого кода

Сигнатуры исполняемого кода позволяет рассчитывать сигнатуры для дискретных участков исполняемого кода. Исполняемый код является кодом, который находится между командами **BeginProg** (Начало программы) и **EndProg** (Конец программы). См. пример CRBasic *Сигнатура программы* *(стр. 178)***.**

CRBasic: пример 27. Сигнатуры программ

```
В этом примере программы показано, как запросить текст сигнатуры программы (ProgSig =
Status.ProgSignature) и
двоичную сигнатуру времени выполнения (RunSig = Status.RunSignature). Здесь также рассчитываются
две сигнатуры сегмента исполняемого кода (ExeSig(1), ExeSig(2)).

Определение переменных типа Public
Public RunSig, ProgSig, ExeSig(2),x,y
```

```
Определение таблицы данных
DataTable(Signatures,1,1000)
  DataInterval(0,1,Day,10)
  Sample(1,ProgSig,FP2)
  Sample(1,RunSig,FP2)
  Sample(2 , ExeSig() , FP2)
EndTable

Программа
BeginProg
  ExeSig() = Signature                    Инициализация функции сигнатуры исполняемого
                                          кода
  Scan(1,Sec,0,0)
    ProgSig = Status.ProgSignature        Определение переменной для записи таблицы Status
                                          (Состояние)
                                          «ProgSignature» (Сигнатура программы).
    RunSig = Status.RunSignature          Определение переменной для записи таблицы Status
                                          (Состояние)
                                          «RunSignature» (Рабочая сигнатура).
    x = 24
    ExeSig(1) = Signature                 Сигнатура включает код с момента первоначальной
                                          команды для сигнатуры.
    y = 43
    ExeSig(2) = Signature                 Сигнатура включает весь код с момента
                                          ExeSig(1) = сигнатура
  CallTable Signatures
NextScan
```

7.9.1.8. Использование многократного сканирования

В примере CRBasic *Использование многократного сканирования* *(стр. 179)* показано использование нескольких циклов сканирования. Некоторые приложения требуют проведения измерений или обработки данных в промежутке, который отличается от промежутка основного сканирования. Перед вторичной или медленной последовательностью сканирование добавляется команда **SlowSequence**.

CRBasic: пример 28. Использование многократного сканирования

```
В этом примере программы показано использование множественного сканирования. Некоторые
приложения требуют
выполнения измерения или обработки в течение промежутка времени, которые отличается от основной
программы сканирования. Повторное сканирование размещается с помощью команды SlowSequence.

Определение общих переменных
Public PTemp
Public Counter1

   Объявление таблицы данных 1
   DataTable(DataTable1,1, -1)            Таблица DataTable1 регулируется событиями.
                                          Событие — сканирование.
      Sample(1,PTemp, FP2)
      Sample(1, Counter1, fp2)
   EndTable


Основная программа
BeginProg                                 Начало исполняемой части программы.
   Scan(1,Sec,0,0)                        Начало основного сканирования.
      PanelTemp(PTemp,250)
      Counter1 = Counter1 + 1
      CallTable DataTable1                Вызов DataTable1
   NextScan                               Завершение основного сканирования.
```

```
  SlowSequence                                  Начало медленной последовательности.
    Объявление переменных Public для вторичного сканирования (может быть объявлено в заголовке
    программы)
    Public Batt_Volt
    Public Counter2

    Объявление таблицы данных
    DataTable(DataTable2 ,1,-1)                 Таблица DataTable2 регулируется событиями.
                                                Событие — сканирование.
      Sample(1,Batt_Volt,FP2)
      Sample(1,Counter2,FP2)
    EndTable

    Scan(5,Sec,0,0)                             Начало 1-го вторичного сканирования
      Counter2 = Counter2 + 1
      Battery(Batt_Volt)
      CallTable DataTable2                      Вызов DataTable2
    NextScan                                    Конец медленной последовательности сканирования
EndProg                                         Конец исполняемой части программы.
```

7.9.2. Компиляция: условный код

Когда в CR1000 отправляется пользовательская программы CRBasic, создается точная копия программы в виде файла на диске *ЦП* *(стр. 371)*. Двоичная версия программы или «операционная программа» создается компилятором CR1000 и записывается в *оперативную память* *(стр. 372)***.** Именно эту версию программы запускает CR1000.

CRBasic позволяет определить условный код, перед которым в программе CRBasic размещен хэш-символ (#), который компилируется в операционную программу в зависимости от условных параметров. Кроме того, все регистраторы данных Campbell Scientific (за исключением CR200) принимают программные файлы или файлы команды **Include()** с расширениями «.DLD». Эта функция позволяет обойти системные фильтры, которые следят за расширениям файлов определенных регистраторов; это позволяет создавать один файла кода для запуска на нескольких моделях регистраторов данных CRBasic.

Примечание. Не следует путать файлы CRBasic с расширениями «.DLD» с файлами «.DLD», которые использовались на старых версиях регистраторов данных Campbell Scientific.

В качестве примера, псевдокод, использующий данную функции, может быть записан в следующем виде:

```
#Const Destination = LoggerType
#If Destination = 3000 Then
  <code specific to the CR3000>
#ElseIf Destination = 1000 Then
  <code specific to the CR1000>
#ElseIf Destination = 800 Then
  <code specific to the CR800>
#ElseIf Destination = 6 Then
  <code specific to the CR6>
#Else
  <code to include otherwise>
#EndIf
```

Такая структура позволяет легко менять константы, например, для измерения включенных команд.

Редактор CRBasic теперь использует опцию предварительной компиляции, что позволяет создавать текстовый файл CRBasic только с необходимыми условными операторами из более объемной основной программы. Эта опция также может использоваться в командной строке компилятора предварительного прохода с помощью команды -p <outfile name>. Эта функция позволяет отправлять в CR1000 наименьший по размеру файл программы, чтобы сократить затраты на передачу по коммуникационным каналам.

В примере CRBasic *Условный код* *(стр. 181)* показан пример программы, демонстрирующий использование условной компиляции в CRBasic. В программе содержатся также примеры использование предварительно определенной контрасты **LoggerType** и связанных с ней предварительно определенных констант регистратора (**6, 800, 1000** и **3000**).

CRBasic: пример 29. условный код

```
'В этом примере программы показана компиляции программы, которая зависит от модели регистратора
'данных и скорости работы программы. Основные команды: #If, #ElseIf, #Else и #EndIf.

'Установка параметров программы на основании:
'LoggerType – это предварительно определенная константа в операционной системе CR1000
'ProgramSpeed, которая определяется в следующем операторе:
Const ProgramSpeed = 2

#If ProgramSpeed = 1
  Const ScanRate = 1                        '1 секунда
  Const Speed = "1 Second"
#ElseIf ProgramSpeed = 2
  Const ScanRate = 10                       '10 секунд
  Const Speed = "10 Second"
#ElseIf ProgramSpeed = 3
  Const ScanRate = 30                       '30 секунд
  Const Speed = "30 Second"
#Else
  Const ScanRate = 5                        '5 секунд
  Const Speed = "5 Second"
#EndIf
'Переменные типа Public
Public ValueRead, SelectedSpeed As String * 50

'Основная программа
BeginProg

  'Возвращение к выбранной скорости и отображение типа регистратора.
  #If LoggerType = 3000
    SelectedSpeed = "CR3000 running at " & Speed & " intervals."
  #ElseIf LoggerType = 1000
    SelectedSpeed = "CR1000 running at " & Speed & " intervals."
  #ElseIf LoggerType = 800
    SelectedSpeed = "CR800 running at " & Speed & " intervals."
  #ElseIf LoggerType = 6
    SelectedSpeed = "CR6 running at " & Speed & " intervals."
  #Else
    SelectedSpeed = "Unknown Logger " & Speed & " intervals."
  #EndIf

  'Открытие последовательного порта.
  Serial Open(ComC1,9600,10,0,10000)

  'Основное сканирование
  Scan(ScanRate,Sec,0,0)
    'Измерение с использованием различных параметров и другого канала SE, который зависит от
    'типа регистратора данных на котором запущена программа.
```

```
  #If LoggerType = 3000
    Эта команда используется для регистратора данных CR3000.
    VoltSe(ValueRead,1,mV1000,22,0,0,_50Hz,0.1,-30)
  #ElseIf LoggerType = 1000
    Эта команда используется для регистратора данных CR1000.
    VoltSe(ValueRead,1,mV2500,12,0,0,_50Hz,0.1,-30)
  #ElseIf LoggerType = 800
    Эта команда используется для регистратора данных серии CR800.
    VoltSe(ValueRead,1,mV2500,3,0,0,_50Hz,0.1,-30)
  #ElseIf LoggerType = 6
    Эта команда используется для регистратора данных серии CR6.
    VoltSe(ValueRead,1,mV1000,U3,0,0,50,0.1,-30)
  #Else
    ValueRead = NAN
  #EndIf
NextScan

EndProg
```

7.9.3. Отображение данных: пользовательские меню — подробная информация

Связанные темы:

- *Пользовательские меню — общие сведения* *(стр. 84, стр. 581)*
- *Экран данных: пользовательские меню — подробная информация* *(стр. 182)*
- *Пользовательские меню — набор команд* *(стр. 581)*
- *Экранная клавиатура — общие сведения* *(стр. 83)*
- *Справка к редактору CRBasic* для **DisplayMenu()**

Меню для отображения экранной клавиатуры CR1000KD можно настроить так, чтобы упростить выполнение рутинных операций. Просмотр данных, переключение функций управления, а также добавление примечаний являются приложениями, которые используются чаще всего. Отдельные экраны меню поддерживают до восьми строк текста, которые содержат до семи переменных.

Используйте следующие команды CRBasic. Для получения дополнительной информации см. разделе *Справка по редактору CRBasic*.

DisplayMenu()

Отмечает начало и конец пользовательского меню. Разрешено использовать в программе только один раз.

Примечание. Метка должна включать не менее шести символов, чтобы перекрыть отображение часов по умолчанию .

EndMenu

Отмечает конец пользовательского меню. Разрешено использовать в программе только один раз.

DisplayValue()

Определяет метку и отображает значения (переменная или данные таблицы значений), которые не подвергаются редактированию, такие как измерения.

MenuItem()

Определяет метку и отображает переменную для редактирования путем ввода или выбора из списка, определенного MenuPick().

MenuPick()

Создает список выбора, согласно которому рекрутируется переменная **MenuItem()**. Идет сразу же после **MenuItem().** Если переменная объявлена с типом **As Boolean, MenuPick()** допускается использование значений True (Истина) и False (Ложь), а также объявленных

эквивалентов. Иначе в списке выбора окажется много элементов. Порядок элементов в списке определяется порядком выполнения команд; однако элемент, первоначально отображенный в **MenuItem**, определяется значением элемента.

SubMenu() / EndSubMenu

Определение начала и конца меню второго уровня.

Примечание. Метка SubMenu() должна быть не менее шести символов, чтобы замаскировать отображение часов по умолчанию.

В примере CRBasic *Пользовательские меню* *(стр. 185)* описано программирование CRBasic для пользовательского меню, которое облегчает просмотр данных, ввод примечаний и управления устройством. Ниже приведен список данных, которые показывают организацию пользовательского меню, созданного в примере CRBasic *Пользовательские меню* *(стр. 185)*.

Пример пользовательского меню — главный экран (стр. 183).
Пример пользовательского меню — окно просмотра данных (стр. 183).
Пример пользовательского меню — подменю создания примечаний (стр. 184).
Пример пользовательского меню — список выбора предварительно определенных примечаний (стр. 184).
Пример пользовательского меню — окно для свободного ввода примечаний (стр. 184).
Пример пользовательского меню — окно подтверждения/удаления примечаний (стр. 184).
Пример пользовательского меню — управление подменю (стр. 185).
Пример пользовательского меню — список выбора контрольных светодиодов (стр 185).
Пример пользовательского меню — список выбора логических контрольных светодиодов (стр 185).

Рисунок 43. Пример пользовательского меню — главный экран

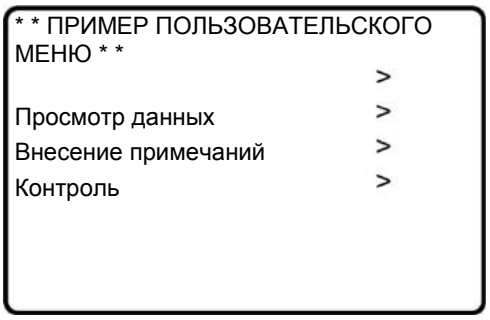


Рисунок 44. Пример пользовательского меню — окно просмотра данных

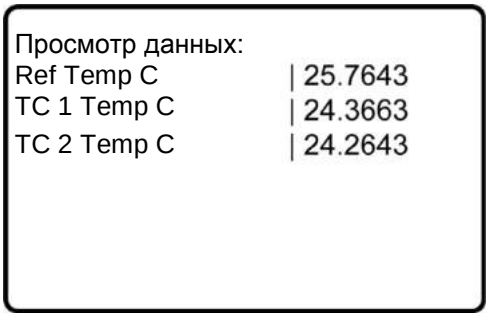
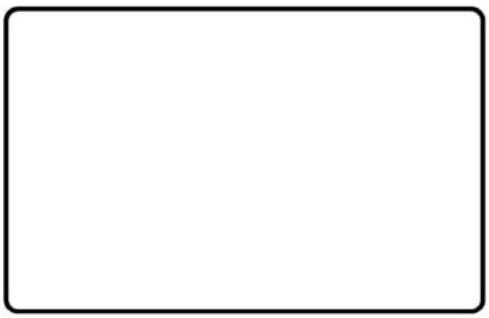
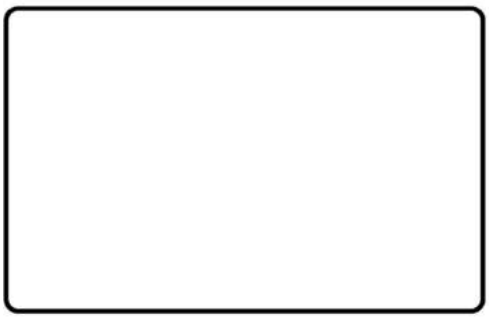


Рисунок 45. Пример пользовательского меню — подменю создания примечаний

о меню — список выбора ıаний



о меню — окно для свободного



о меню — окно подтверждения/

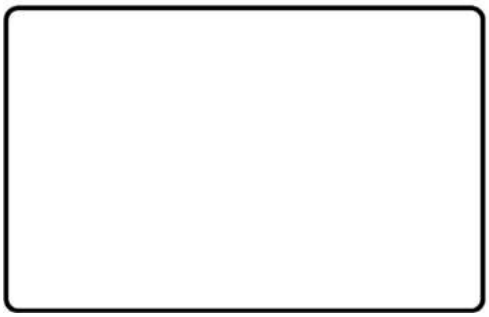
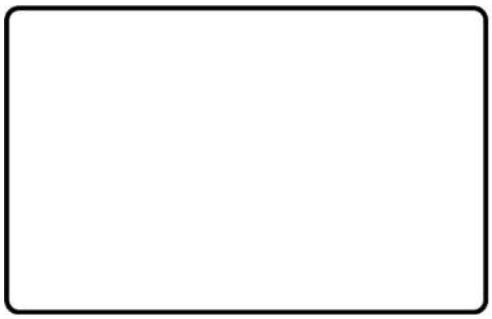
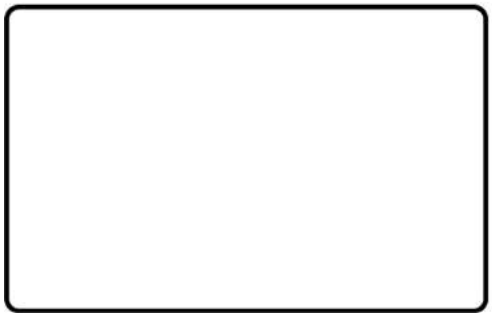


Рисунок 49. Пример пользовательского меню — управление подменю



меню — список выбора

меню — список выбора



Примечание. Рисунки от *Пример пользовательского меню — главный экран (стр. 183)* до *Пример пользовательского меню — список выбора логических контрольных светодиодов (стр. 185)* можно использовать в качестве справочного материала для следующего примера CRBasic *Пользовательские меню (стр. 84, стр. 581)*.

CRBasic: пример 30. Пользовательские меню

```
'В этом примере программы показано построение пользовательского меню экранной клавиатуры
'CR1000KD.

'Объявления, поддерживающие пункт меню View Data (Просмотр данных)
Public RefTemp                              'Переменная эталонной температуры
Public TCTemp(2)                            'Массив температуры от термопары

'Объявления, поддерживающие пункт меню с пустой строкой
Const Escape = "Hit Esc”                    'Слово указывает на действие для выхода из
                                            'тупиковой ситуации

'Объявления, поддерживающие пункт меню Enter Notes (Ввод примечаний)
Public SelectNote As String * 20            'Содержит примечание из предварительно
                                            'определенного списка
```

```
Const Cal_Done = ”Cal Done”                  Слово сохраняется при выборе Cal_Don
Const Offst_Chgd = ”Offset Changed”          Слово сохраняется при выборе Offst_Chgd
Const Blank =                                Слово сохраняется при выборе пустого значения
Public EnterNote As String * 30              Переменная для сохранения пустой записи
Public CycleNotes As String * 20             Переменная для сохранения контрольного слова
                                             примечания
Const Accept = ’’Accept”                     Контрольное слова примечания
Const Clear = ”Clear”                        Контрольное слова примечания

Объявления, поддерживающие пункт меню Control (Управление)
Const On = true                              Присвоение «Вкл.» логического значения истина
Const Off = false                            Присвоение «Выкл.» логического значения ложь
Public StartFlag As Boolean                  Переменная процесса управления светодиодом
Public CountDown As Long                     Переменная обратного отсчета светодиода
Public ToggleLED As Boolean                  Переменная управления светодиодом

Определение применения для таблицы данных    Настройка примечаний к таблице данных, записанной
DataTable(Notes,1,-1)                        в момент подтверждения применения
   Sample(1, SelectNote, String)             Примечание для списка выбора измерений
   Sample(1,EnterNote,String)                Примечание для измерения без записи
EndTable

Определение температуры в таблице данных     Настройка таблицы данных температуры.
DataTable(TempC, 1, -1)                      Запись каждые 60 секунд:
   DataInterval(0,60,Sec,10)
   Sample(1,RefTemp,FP2)                     Измерение эталонной температуры
   Sample(1,TCTemp(1),FP2)                   Измерение термопарой 1
   Sample(1,TCTemp(2),FP2)                   Измерение термопарой 2
EndTable

Объявления пользовательского меню
DisplayMenu(”**CUSTOM MENU DEMO**”,-3)       Создание меню; после включения питания появляется
                                             пользовательское меню. Системное меню скрыто
                                             от пользователя.

   SubMenu(””)                               Макет подменю для записи в пустую строку
     DisplayValue(”” , Escape)               пустая строка
   EndSubMenu                                Конец макета подменю

   SubMenu(”View Data ”)                     Создание подменю с именем PanelTemps
     DisplayValue(”Ref Temp C”,RefTemp)      Элемент для подменю от Public
     DisplayValue(”TC 1 Temp C”,TCTemp(1))   Элемент для подменю — TCTemps(1)
     DisplayValue(”TC 2 Temp C”,TCTemp(2))   Элемент для подменю — TCTemps(2)
   EndSubMenu                                Конец подменю

   SubMenu(”Make Notes ”)                    Создание подменю с именем PanelTemps
     MenuItem(”Predefined”,SelectNote)       Выбор предварительно определенных примечаний для
                                             пункта меню.
     MenuPick(Cal_Done,Offset_Changed)       Создание списка выбора предварительно определенных
                                             примечаний.
     MenuItem(”Free Entry”,EnterNote)        Введенное пользователем примечание к пункту меню
     MenuItem(”Accept/Clear”,CycleNotes)
     MenuPick(Accept, Clear)
   EndSubMenu
```

```
   SubMenu("Control")                          Создание подменю с именем PanelTemps
    MenuItem("Count to LED",CountDown)         Создание пункта меню CountDown (Обратный отсчет)
    MenuPick(15,30,45, 60)                     Создание списка выбора для CountDown (Обратный
                                               отсчет)
    MenuItem("Manual LED",toggleLED)           Пункт меню ручного управления светодиодом
    MenuPick(On,Off)
    EndSubMenu
EndMenu                                        Создание пользовательского меню

Основная программа
BeginProg

   CycleNotes = "??????"                       Инициализация примечаний в подменю,
                                               запись ????? с нулевым значением
   Scan(1,Sec,3,0)

    Измерения
    PanelTemp(RefTemp,250)                     Эталонная температура измерения
                                               Измерение с помощью двух термопар
    TCDiff(TCTemp(),2,mV2_5C,1,TypeT,RefTemp,True,0,_60Hz,1.0,0)
    CalTable TempC                             Вызов таблицы данных

    Код поддержки для пункта меню «Внесение примечаний»
    If CycleNotes = "Accept" Then
     CalTable Notes                            Внесение данных в примечания таблицы данных
     CycleNotes = "Accepted"                   Внесение «принято» после записи
     Delay(1,500,mSec)                         Пауза, чтобы пользователь мог прочитать
                                               «принятое»
     SelectNote = ""                           Очистить примечание списка выбора
     EnterNote = ""                            Очистить примечание свободной записи
     CycleNotes = "??????"                     Записать ?????в качестве нулевой строки
    EndIf
    If CycleNotes = "Clear" Then               Очистить примечания по запросу
     SelectNote = ""                           Очистить примечание списка выбора
     EnterNote = ""                            Очистить примечание свободной записи
     CycleNotes = "??????"                     Записать ?????в качестве нулевой строки
    EndIf

    Пункт меню Control (Управление), код поддержки меню
    CountDown = CountDown - 1                  Обратный отсчет до 1
    If CountDown <= 0                          остановка обратного отсчета при достижении 0
     CountDown = 0
    EndIf
    If CountDown > 0 Then
     StartFlag = True                          Указывает начало обратного отсчета
    EndIf
    If StartFlag = True AND CountDown = 0 Then Межпроцессный обратный отсчет
                                               и ручное управление светодиодом
     ToggleLED = True
     StartFlag = False
    EndIf
    If StartFlag = True AND CountDown <> 0     Межпроцессный обратный отсчет и ручное управление
    Then                                       светодиодом
     ToggleLED = False
    EndIf
    PortSet(4,ToggleLED)                       Выбор порта управления в соответствии с
                                               результатами обработки
   NextScan
EndProg
```

7.9.4. Входные данные: загрузка больших наборов данных

Большие наборы данных, например таблицы просмотра или номера тегов, можно загрузить в CR1000 для использования программой CRBasic. Это можно выполнить с помощью команд **Data, DataLong** и **Read**, как показано в примере CRBasic *Загрузка больших наборов данных* *(стр. 188).*

CRBasic: пример 31. Загрузка больших наборов данных

```
В этом примере программы показана загрузка набора данных в переменные. Двадцать значений
загружаются в два массива: один объявляется Float, второй — As Long. Отдельные строки данных
могут быть значительно длиннее, чем указано (длина в данном случае ограничена только
максимальной длиной оператора), и можно добавить любое количество строк. Таким образом можно
загрузить тысячи значений.

Объявление переменных типа Float и Long. Также можно объявить переменные с типом Dim.
Public DataSetFloat(10) As Float
Public DataSetLong(10) As Long
Dim x

Запись набора данных в память CR1000.
Data 1.1,2.2,3.3,4.4,5.5
Data -1.1,-2.2,-3.3,-4.4,-5.5
DataLong 1,2,3,4,5
DataLong -1,-2,-3,-4,-5

Объявление таблицы данных
DataTable (DataSet_,True,-1)
  Sample (10,DataSetFloat(),Float)
  Sample (10,DataSetLong(),Long)
EndTable

BeginProg

  Присвоение типа данных Float массиву переменных, объявленному с параметром As Float.
  For x = 1 To 10
     Read DataSetFloat(x)
  Next x

  Присвоение типа данных Long массиву переменных, объявленному с параметром As Long.
  For x = 1 To 10
Read DataSetLong(x)
  Next x

  Scan(1,sec,0,1)

     Запись всех данных в память окончательных данных.
     CallTable DataSet_

  NextScan

EndProg
```

7.9.5. Входные данные: выражение для назначения массива

CRBasic предусматривает следующие операции для одно аспекта многомерного массива:

- инициализация;
- преобразование;
- копирование;
- математические операции;
- логические операции.

Например:

- Обработка массива переменных без использования команд **For/Next (Для/далее).**
- Создание логических массивов на основе сравнения с другим массивом или скалярной переменной.
- Копирование данных измерений в новое место.
- Выполнение логических операций для каждого элемента в измерении с помощью скаляра или аналогично расположенных элементов из разных массивов и размерностей.

Примечание. Выражение для назначения массива является альтернативой команде **For/Next**, и обычно используется опытными программистами. Вероятно, это не сократит время обработки по сравнению с использованием команды **For/Next** (Для/далее). Чтобы сократить время обработки попробуйте использовать команду **Move()**, которая требует более насыщенного кода.

Правила синтаксиса:

- Определения:
  - Наименее значимый аспект — последняя или самая правая цифра в индексе массива. Например, в массиве ***array(a,b), b*** является наименее значимым индексом. В массиве ***array(a,b,c),*** c является наименее значимым индексом.
  - Отрицание — размещение отрицания или знака минус (–) перед индексом массива. Например, применение отрицания к наименее значимому размеру массива ***array(a,b,c)***, он примет вид ***array(a,b,–c).***
- Пустой набор скобок обозначает выражение для назначения массива. Например, ссылка ***array()*** или ***array(a,b,c)().***
- Только одно измерение массива может работать одновременно.
- Чтобы выбрать размер для выполнения, необходимо присвоить отрицательное значение размеру данного индекса.
- Операции не пересекают размеры. Операция начинается с указанной начальной точки и продолжается до одного из следующих значений:
  - Конец размера.
  - В случае, если размер задается отрицательным.
  - В случае, если размер является наименее значимым (по умолчанию).
- Если индексы не указаны или ни один из них не обрабатывался со знаком минус, предполагается использование наименее значимого размера массива.
- Доступ к смещению в размере предоставляется ***(a,b,c).***
- Если на массив ссылаются как на ***array(),*** становится доступна начальная точка массива ***array(1,1,1)*** и его наименее значимый размер. Например, если массив объявлен, как ***test(a,b,c)***, а затем на него дается ссылка ***test()***, то отправной точкой является ***test(1,1,1)*** и доступ к размеру **c.**

Таблица 24. Пример CRBasic. Выражение для назначения массива: сумма столбцов и строк

```
В этом примере проводится суммирование трех строк и двух столбцов массива 3x2.

Изображение исходного массива:
'1.23,2.34
'3.45,4.56
'5.67,6.78

Public Array(3,2) = {1.23,2.34,3.45,4.56,5.67,6.78} 'load values into source array
Public RowSum(3)
Public ColumnSum(2)

BeginProg
   Scan(1,Sec,0,0)
      Для каждой строки сложить две колонки.
      RowSum() = Array(-1,1)() + Array(-1,2)()
      Для каждого столбца сложить три строки.
      ColumnSum() = Array(1,-1)() + Array(2,-1)() + Array(3,-1)()
   NextScan
EndProg
```

Таблица 25. Пример CRBasic. Выражение для назначения массива: преобразование массива

```
В данном примере массив 3x2 преобразовывается в массив 2x3.
Изображение исходного массива:
'1,2
'3,4
'5,6

Изображение итогового массива (преобразование источника):
'1,3,5
'2,4,6

Размер и инициализация исходного массива
Public A(3,2) = {1,2,3,4,5,6}

Размер итогового массива
Public At(2, 3)

Объявление для / следующий счетчик
Dim i

BeginProg
   Scan (1,Sec,0,0)
    For i = 1 To 2
       Для каждого столбца исходного массива A(), скопировать столбец в строку
       итогового массива At().
       At(i, -1)() = A(-1,i)()
    Next i
   NextScan
EndProg
```

Таблица 26. Пример CRBasic. Выражение для назначения массива: сравнение / логическая оценка

```
'Пример: сравнение / логическая оценка

'Поэлементно сравнение выполняется путем увеличения скаляра или сравнения каждого
'элемента в одном массиве с аналогично расположенным элементом в другом массиве для создания
'логического массив, который будет использоваться для принятия решений и контроля, например,
'входного массива в SDM-CD16AC.

Public TempC(3) = {15.1234,20.5678,25.9876}
Public TempC_Rounded(3)
Public TempDiff(3)
Public TempC_Alarm(3) As Boolean
Public TempF_Thresh(3) = {55,60,80}
Public TempF_Alarm(3) As Boolean

BeginProg
  Scan(1,Sec,0,0)

    'Поэлементное сравнение каждой температуры в массиве со скалярной величиной
    'присваивает соответствующему логическому аварийному сигналу значение истина, если
    'температура превышает 20 градусов Цельсия.
    TempC_Alarm() = TempC() > 20

    'Однако только некоторые, а не все команды будут принимать это обозначение массива создания
    'автоматического индекса
    'массива.

    'Округление каждого значения температуры с точностью до десятых долей градуса.
    TempC_Rounded() = Round(TempC(),1)

    'Поэлементное вычитание
    'Каждый элемент в TempC_Rounded вычитается из аналогично расположенного элемента inTempC
    'вычисление разности между каждым значением TempC и округленной копией
    TempDiff() = TempC() - TempC_Rounded()

    'Операции поэлементного сравнения можно сочетать со скалярными операциями расширения
    'Выбор соответствующего сигнала тревоги, если логическое значение истинно, если
    'температура, после
    'преобразования в градусы по Фаренгейту превышает соответствующее пороговое значение для
    'градусов по Фаренгейту.
    TempF_Alarm() = (TempC() * 1.8 + 32) > TempF_Thresh()

  NextScan
EndProg
```

Таблица 27. Пример CRBasic. Выражение для назначения массива: размер заполнения массива

```
Пример. Размер заполнения массива

Public A(3)
Public B(3,2)
Public C(4,3,2)
Public Da(3,2) = {1,1,1,1,1,1}
Public Db(3,2)
Public DMultip1ier(3) = {10,100,1000}
Public DOffset(3) = {1,2,3}

BeginProg
  Scan(1,Sec,0,0)

     A() = 1 установить все элементы 1D массива или первой размерности равными 1

     B(1,1)() = 100      Установить B(1,1) и B(1,2) равными 100.
     B(-2, 1) () = 200   Установить B(2,1) и B(3,1) равными 200.
     B(-2, 2) () = 300   Установить B(2,2) и B(3,2) равными 300.

     C(1,-1,1)() = A() Копировать A(1), A(2) и A(3) в C(1,1,1), C(1,2,1) и C(1,3,1)
                       соответственно.
     C(2,-1,1)() = A() * 1.8 + 32 масштабирование, а затем копирование A(1), A(2), м A(3) в
                                  C(2,1,1), C (2,2,1) и C (2,3,1), соответственно,


     Масштабирование первого столбца Da с помощью соответствующего множителя и копирование
     смещения в результат для первого столбца Db,
     затем установка второго столбца Db равным NAN.
     Db(-1,1)() = Da(-1,1)() * DMu1tip1ier() + DOffset()
     Db(-1,2) () = NAN

  NextScan
EndProg
```

7.9.6. Выходные данные: вычисление скользящего среднего

Команда **AvgRun()** вычисляет скользящее среднее значение измерения или расчетной величины. Скользящее среднее(***Dest***) представляет собой среднее значение последних N значений, где N равно числу значений, указанному в уравнении скользящего среднего:

$$\mathbf{Dest} = \frac{\sum_{i-1}^{i-N} X_i}{N}$$

где X_N — последнее значение исходной переменной, а X_{N-1} — предыдущее значение (X_1 это самое старое значение, используемое для вычисления среднего значения, то есть значения N–1 возвращают назад от самого последнего). NAN игнорируются при обработке **AvgRun()**, если все значения являются NAN.

AvgRun() использует расчеты с высокой точностью, поэтому сохранение и использование внутри системы 32-битное расширения дробной части приводит к точности 56 бит.

Примечание. Эту команду не следует вводить в логическую структуру **For/Next** с индексированными параметрами ***Source*** и ***Dest*** и параметром ***Reps***, который задан равным ***1***. Если это выполнить, то скользящее среднее будет определяться, используя значения различных элементов массива, вместо расчета независимого скользящего среднего значения по каждому элементу массива. Результаты будут представлять собой скользящее среднее пространственного среднего значения по различным элементам исходного массива.

Скользящее среднее является цифровым фильтром нижних частот; его выход ослабляется в зависимости от частоты, а также выдается с задержкой по времени. Степень затухания и фазовый сдвиг (время задержки) зависит от частоты входного сигнала и продолжительности времени (которое связано с числом точек) для вычисления скользящего среднего.

На рисунке *Скользящее среднее частотной характеристики* *(стр. 194)* показан график затухания сигнала в зависимости от частоты сигнала, нормированной до 1/(интервал для определения скользящего среднего). Сигнал ослабляется синхронизирующим фильтра с коэффициентом 1 (простое усреднение): Sin(πX) / (πX), где X — отношение частоты входного сигнала у скользящей средней частоты (скользящая средняя частоты = 1/промежуток усреднения).

Пример:

Период сканирования = 1 мс,

Значение N = 4 (среднее количество точек),

Средняя длительность выполнения = 4 мс.

Скользящее среднее значение частоты = 1 / (длительность скользящего среднего = 250 Гц).

Частота входного сигнала = 100 Гц.

Частота входного сигнала для скользящего среднего (нормированная частота) = 100/250 = 0,4.

Sin(0,4π) / (0,4π) = 0,757 (или см. рисунок *Скользящее среднее частотной характеристики* *(стр. 194)*, , где ось X составляет 0,4).

Для входного сигнала 100 Гц с полной амплитудой 10 В, скользящее среднее выдает сигнал 100 Гц с полной амплитудой 7,57 В.

Существует также фазовый сдвиг или задержка на выходе команды **AvgRun()**. Формула для расчета задержки для некоторых измерений:

Задержка в измерениях = (N–1)/2

Примечание. N = количество точек при вычислении скользящее среднего.

Для того, чтобы вычислить задержку во времени, умножьте результат из приведенного выше уравнения на период расчета скользящее среднее значения (как правило, период сканирования):

Задержка во времени = (период сканирования) • (N–1) / 2

Для приведенного выше примера, задержка:

Задержка по времени = (1 мс)• (4 – 1) / 2 = 1,5 мс.

Пример:

Тензодатчик прошел испытание при установке на балке. Испытание показало следующие результаты:

- резонансная частота тензодатчика ≈ 36 Гц;
- период измерения = 2 мс;
- средняя длительность выполнения = 20 мс (частота 50 Гц).

Нормированная резонансная частота была рассчитана следующим образом:

```
36 Hz / 50 Hz = 0.72
SIN(0.72π) / (0.72π) = 0.34.
```

Таким образом, записанная амплитуда составляла около 1/3 от амплитуды входного сигнала. Была написана программа CRBasic с переменными **Accel2** и **Accel2RA**. Необработанные измерения хранились в **Accel2. В Accel2RA** сохранялся результат расчета скользящего среднего значения для **Accel2**. Оба значения сохранялись с частотой 500 Гц. На рисунке *Затухание скользящего среднего значения сигнала* *(стр. 195)* для иллюстрации затухания показаны графики двух переменных. Скользящее среднее значение имеет более низкую амплитуду.

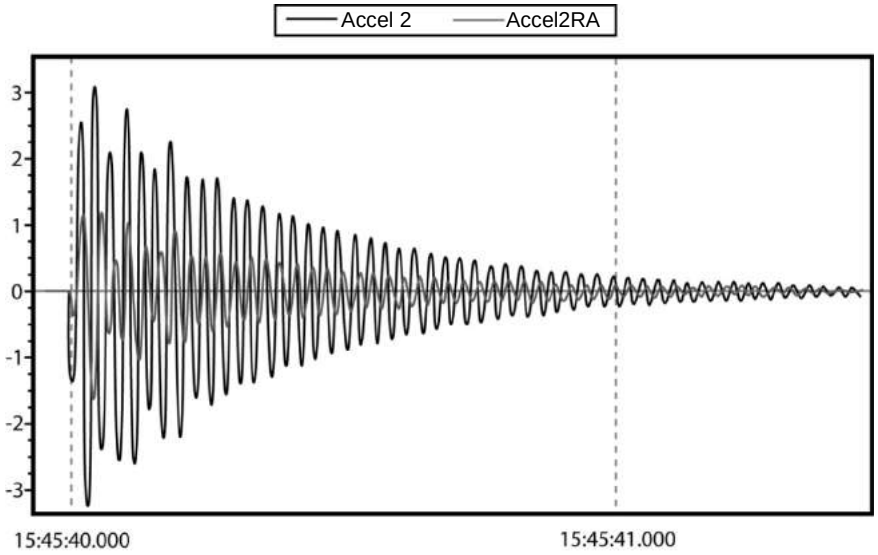
Полученное в результате задержки значение D_r, рассчитывается следующим образом:

D_r = (скорость сканирования) • (N-1)/2 = 2 мс (10-1)/2 = 9 мс

D_r составляет около 1/3 периода входного сигнала.

Рисунок 52. Скользящее среднее частотной характеристики

Рисунок 53. Затухание скользящего среднего значения сигнала



7.9.7. Выходные данные: триггеры и исключительная выборка

TrigVar — третий параметр команды **DataTable()**. Она контролирует, внесена ли запись в память окончательных данных. ***TrigVar*** выполняется с учетом других условных команд, таких как **DataInterval** и **DataEvent()**.

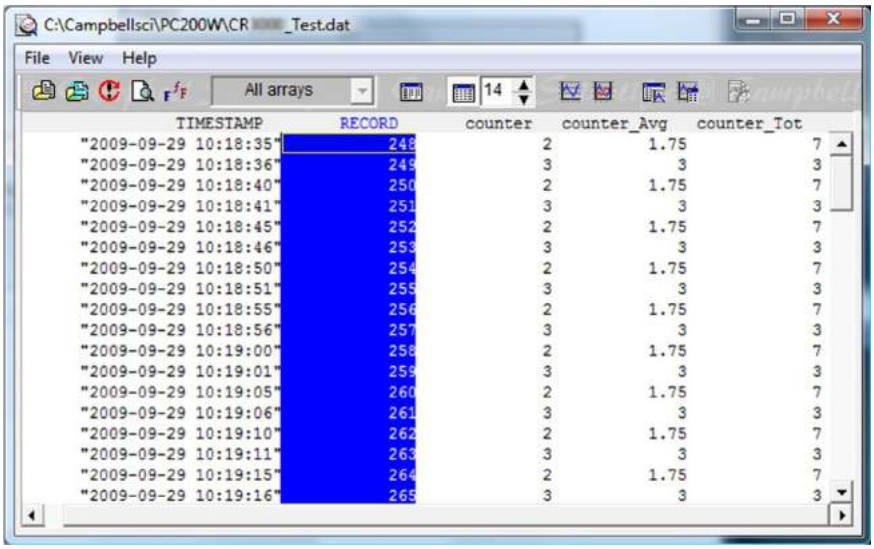
DisableVar — последний параметр в большинстве команд для обработки выходных данных, например, **Average(), Maximum(), Minimum()** и т. д. Он контролирует, используется ли конкретное измерение или значение функцией обработки выходных данных.

Для отдельных измерений, которые влияют на сводные данные, команды для обработки выходных данных, такие как **Average(),** должны выполняться при каждом вызове таблицы данных из программы — обычно один раз за сканирование. Например, для среднего значения, рассчитанного за час каждое измерение необходимо прибавлять к общей протяжности более часа. Такое накопление данных не зависит от ***TrigVar. TrigVar*** контролирует только тот момент, когда выполняется окончательный расчет, а обработанные данные (усредненные) записываются в таблицу данных. В этот момент ***TrigVar*** и все прочие условия (например, **DataInterval**() и **DataEvent()**) должны иметь значение истина. Говоря по-другому, если ***TrigVar*** имеет значение ложь, команды по обработке выходных данных (например, **Average()**) выполняют промежуточную обработку, но не окончательный процесс, поэтому новая запись не создается.

Примечание. Во многих приложениях выходные записи основаны исключительно на интервале, и ***TrigVar*** всегда имеет значение **TRUE (–1) (Истина).** В таких приложениях команда **DataInterval** ( ) является единственным указателем состояния выходного триггера.

На рисунке *Данные программы TrigVar* *(стр. 196)* показаны данные, полученные в примере CRBasic *Использование TrigVar для триггера сохранения данных* *(стр. 196)*, где для триггера сохранения данных используется ***TrigVar*** вместо **DataInterval**.

Рисунок 54. Данные программы TrigVar



TIMESTAMP	RECORD	counter	counter_Avg	counter_Tot
"2009-09-29 10:18:35"	248	2	1.75	7
"2009-09-29 10:18:36"	249	3	3	3
"2009-09-29 10:18:40"	250	2	1.75	7
"2009-09-29 10:18:41"	251	3	3	3
"2009-09-29 10:18:45"	252	2	1.75	7
"2009-09-29 10:18:46"	253	3	3	3
"2009-09-29 10:18:50"	254	2	1.75	7
"2009-09-29 10:18:51"	255	3	3	3
"2009-09-29 10:18:55"	256	2	1.75	7
"2009-09-29 10:18:56"	257	3	3	3
"2009-09-29 10:19:00"	258	2	1.75	7
"2009-09-29 10:19:01"	259	3	3	3
"2009-09-29 10:19:05"	260	2	1.75	7
"2009-09-29 10:19:06"	261	3	3	3
"2009-09-29 10:19:10"	262	2	1.75	7
"2009-09-29 10:19:11"	263	3	3	3
"2009-09-29 10:19:15"	264	2	1.75	7
"2009-09-29 10:19:16"	265	3	3	3

CRBasic: пример 32. Использование TrigVar для триггера сохранения данных

```
'В этом примере программы показано использование параметра TrigVar в DataTable().
'Команда для триггера сохранения данных. В этом примере значение переменной Counter (Счетчик)
'увеличивается на 1 при каждом сканировании. Таблица данных, которая включает команды Sample(),
'Average() и Totalize(), вызывается при каждом сканировании. Данные сохраняются, когда TrigVar
'имеет значение Истина, а TrigVar это происходит тогда, когда Счетчик = 2 или Счетчик = 3.
'Сохраняемые данные являются измерением, средним значением и суммой переменной счетчика, которая
'равна 0, 1, 2, 3 или 4 при вызове таблицы данных.


Public Counter

DataTable(Test,Counter=2 or Counter=3,100)
  Sample(1,Counter,FP2)
  Average(1,Counter,FP2,False)
  Totalize(1,Counter,FP2,False)
EndTable

BeginProg
  Scan(1,Sec,0,0)
    Counter = Counter + 1
    If Counter = 5 Then
     Counter = 0
    EndIf
    CallTable Test
  NextScan
EndProg
```

7.9.8. Выходные данные: два интервала в одной таблице данных

CRBasic: пример 33. Два интервала выходных данных в одной таблице данных.

```
'В этом примере программы показано использование двух временных интервалов в таблице данных.
'Наличие одного интервала в таблице данных является нормой, однако некоторым приложениям
'требуется два интервала.

'Таблице с двумя временными интервалами следует выделять память так же, как это делается для
'условной таблицы:
'отключить автоматическое выделение памяти и установить фиксированное количество записей.

'Определение общих переменных
Public PTemp, batt_volt, airtempC, deltaT
Public int_fast As Boolean
Public int_slow As Boolean
Public counter(4) As Long

'Объявление таблицы данных

'Таблица выдает значения в одном из двух интервалов, в зависимости от указанного условия.
'Обратите внимание на круглые скобки вокруг операторов TriggerVariable и AND (И).

DataTable(TwoInt,(int_fast AND TimeIntoInterval(0,5,Sec)) OR (int_slow AND _
          TimeIntoInterval(0,15,sec)),-1)
   Minimum(1,batt_volt,FP2,0,False)
   Sample(1,PTemp,FP2)
   Maximum(1,counter(1),Long,False,False)
   Minimum(1,counter(1),Long,False,False)
   Maximum(1,deltaT, FP2 ,False,False)
   Minimum(1,deltaT, FP2 ,False,False)
   Average(1,deltaT,IEEE4,false)
EndTable

'Основная программа
BeginProg
   Scan(1,Sec,0,0)

    PanelTemp(PTemp,250)
    Battery(Batt_volt)
    counter(1) = counter(1) + 1

    'Измерение с помощью термопары
    TCDiff(AirTempC,1,mV2_5C,1,TypeT,PTemp,True,0,250,1.0,0)
    'вычисляется разницу между температурой воздуха и температурой коммутационной панели.
    deltaT = airtempC - PTemp

    'Когда разность температуры воздуха > = 3, загорается светодиод и вызывает
    'сокращение двух интервалов для данных таблицы.
    If deltaT >= 3 Then
      PortSet(4,true)
      int_fast = true
      int_slow = false
    Else
      PortSet(4,false)
      int_fast = false
      int_slow = true
    EndIf
```

```
    Вызов таблицы выходных данных.
    CallTable TwoInt

  NextScan
EndProg
```

7.9.9. Выходные данные: использование типа данных Bool8

Переменные, которые используются исключительно для хранения значений типа **True** (Истина) или **False** (Ложь), обычно объявляются логическими переменными или **As BOOLEAN**. При записи в память окончательных данных статус логических переменных, как правило, сохраняется в виде данных типа ***BOOLEAN***. Тип данных ***BOOLEAN*** использует четырехбайтный формат целого. Для экономии памяти окончательного данных или затрат на коммуникации можно использовать тип данных ***BOOL8***. BOOL8 представляет собой значение одного байта, который содержит восемь бит информации (восемь состояний, один бит на состояние). Для того, чтобы сохранить ту же информацию с помощью 32 битной типа данных ***BOOLEAN,*** потребуется 256 бит (8 состояний * 32 бита на одно состояние).

При программировании с использованием типа данных BOOL8 количество повторов в команде обработки выходных данных **DataTable()** должно быть четным, так как не удастся сохранить нечетное число байт. Также отметим, что, когда CR1000 преобразует тип данных LONG или FLOAT в BOOL8, используются только менее значимые восемь бит двоичного эквивалента, то есть используется только бинарное представление десятичного целого числа, разделенное *по модулю* *(стр. 520)* на 256.

Пример:

```
Given: LONG integer 5435
Find: BOOL8 representation of 5435
Solution:
     5435 / 256 = 21.2304687
     0.2304687 * 256 = 59
     Binary representation of 59 = 00111011 (CR1000 stores these
     bits in reverse order)
```

Когда *ПО поддержки регистратора данных* *(стр. 95)* возвращает значение BOOL8, оно разделяется на восемь полей **–1** или **0**, которые сохраняются в ASCII-файл. Следовательно, для файла ASCII требуется больший объем памяти, но память CR1000 сохраняется. Компактный тип данных ***BOOL8*** также использует меньшую полосу пропускания при передаче по коммуникационным каналам.

В примере CRBasic *Программирование с помощью Bool8 и операторов битового сдвига* *(стр. 200)* в качестве учебника упражнения показана программа CR1000 для отслеживания состояния 32 «аварийных сигналов». Аварийные сигналы переключаются вручную с помощью ввода нулевого или не нулевого значения (например, 0 или 1) в каждую общую переменную, представляющую аварийный сигнал, как показано на рисунке *Пример переключение аварийных сигналов при битовом сдвиге* *(стр. 199)*. Измерения четырех общих переменных ***FlagsBool8(1), FlagsBool8(2), FlagsBool8(3)*** и ***FlagsBool8(4)*** хранятся в таблице данных **Bool8Data** в виде четырех однобайтных значений. Однако, как показано на рисунке *Данные типа Bool8 из примера битового сдвига (цифровой монитор)* *(стр. 199)* при просмотре таблицы данных в *цифровом мониторе* *(стр. 521)*, данные удобно трансформированы в 32 значения **True** (Истина) или **False** (Ложь). Кроме того, как показано на рисунке *Данные типа Bool8 из примера битового сдвига (файл данных ПК)* *(стр. 200)*, когда *ПО поддержки регистратора данных* *(стр. 95)* хранит данные в файле ASCII, он хранится в виде 32 столбцов значений **–1** или **0**, где каждый столбец представляет аварийный сигнал. В программе CRBasic можно использовать *псевдонимы переменных* *(стр. 138)*, чтобы упростить структуру данных.

Рисунок 55. Пример переключения аварийных сигналов с битовым сдвигом

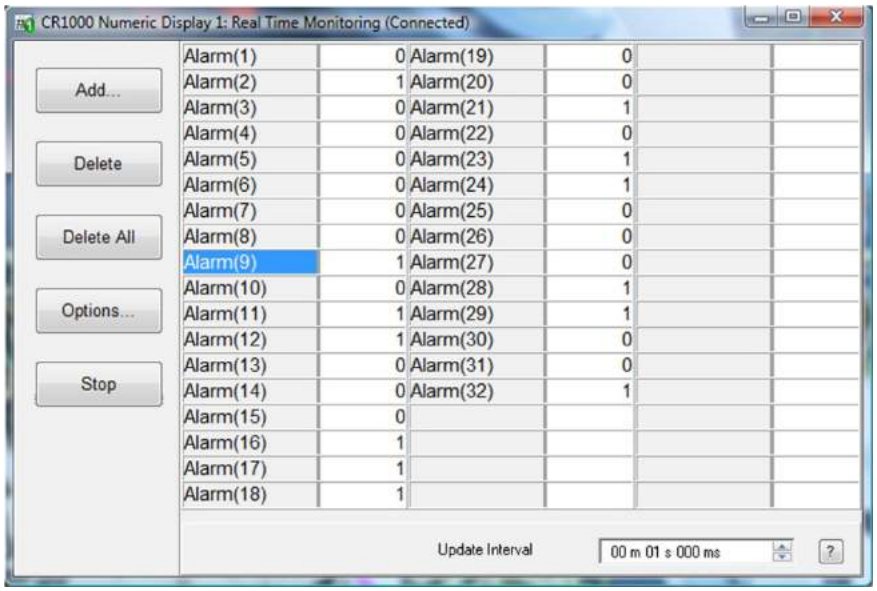


Рисунок 56. Данные типа Bool8 из примера битового сдвига (цифровой монитор)

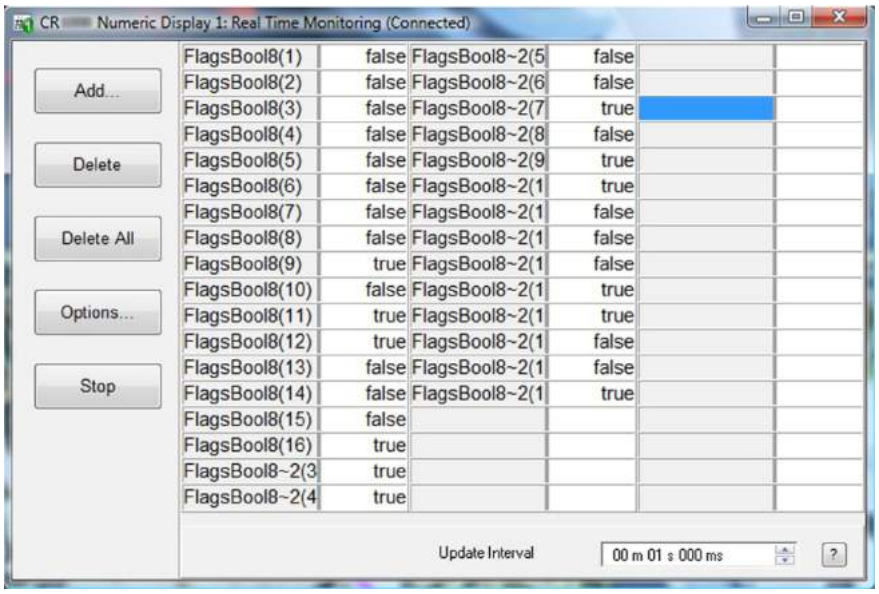
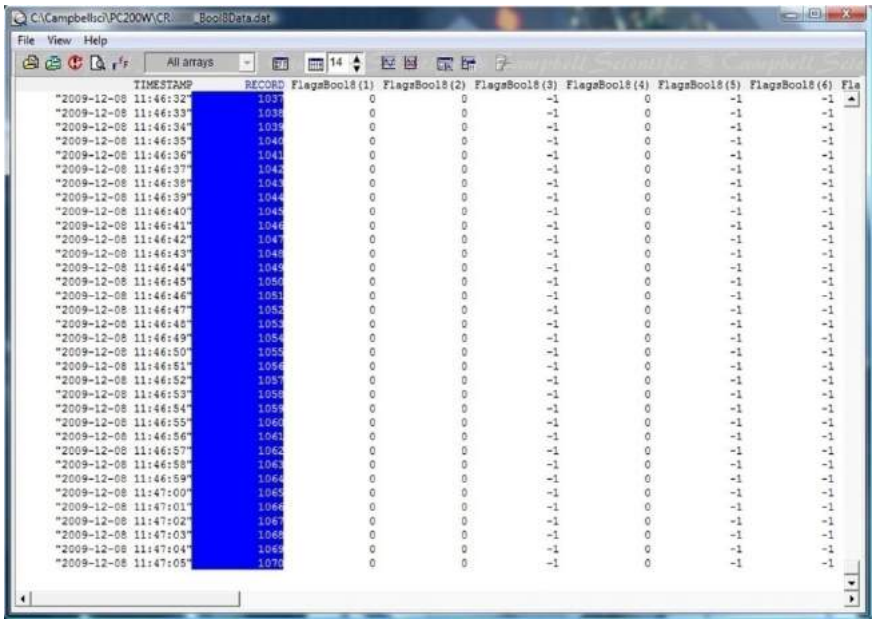


Рисунок 57. Данные типа Bool8 из примера битового сдвига (файл данных ПК)



CRBasic: пример 34. Программирование с помощью Bool8 и оператора битового сдвига

```
В этом примере программы показано использование типа данных Bool8 и оператора битового сдвига «>>».

Public Alarm(32)
Public Flags As Long
Public FlagsBoo18(4) As Long

DataTable(Bool8Data,True,-1)
   DataInterval(0,1,Sec,10)
   сохранение бит с 1 по 16 в колонках файла данных с 1 по 16
   Sample(2 , FlagsBool8(1) , Bool8)
   сохранение бит с 17 по 32 в колонках файла данных с 17 по 32
   Sample(2 , FlagsBool8(3) , Bool8)
EndTable

BeginProg
   Scan(1,Sec,3,0)

      Сброс всех битов на каждый проход до выборочного определения бит.
      Flags = &h0

      Выборочное определение бит Для экономии места используется шестнадцатеричный формат.

      Логическое ИЛИ побитовое сравнение
```

```
' Если бит в   ИЛИ бит в     Результат
' флажках =    двоичю/       =
               шестнадц. =
' ----------   ----------    ----------
'     0            0             0
'     0            1             1
'     1            0             1
'     1            1             1

Двоичный эквивалент шестн.:
If Alarm(1) Then Flags = Flags OR &h1          '                                  &b1
If Alarm(2) Then Flags = Flags OR &h2          '                                 &b10
If Alarm(3) Then Flags = Flags OR &h4          '                                &b100
If Alarm(4) Then Flags = Flags OR &h8          '                               &b1000
If Alarm(5) Then Flags = Flags OR &h10         '                              &b10000
If Alarm(6) Then Flags = Flags OR &h20         '                             &b100000
If Alarm(7) Then Flags = Flags OR &h40         '                            &b1000000
If Alarm(8) Then Flags = Flags OR &h80         '                           &b10000000
If Alarm(9) Then Flags = Flags OR &h100        '                          &b100000000
If Alarm(10) Then Flags = Flags OR &h200       '                         &b1000000000
If Alarm(11) Then Flags = Flags OR &h400       '                        &b10000000000
If Alarm(12) Then Flags = Flags OR &h800       '                       &b100000000000
If Alarm(13) Then Flags = Flags OR &h1000      '                      &b1000000000000
If Alarm(14) Then Flags = Flags OR &h2000      '                     &b10000000000000
If Alarm(15) Then Flags = Flags OR &h4000      '                    &b100000000000000
If Alarm(16) Then Flags = Flags OR &h8000      '                   &b1000000000000000
If Alarm(17) Then Flags = Flags OR &h10000     '                  &b10000000000000000
If Alarm(18) Then Flags = Flags OR &h20000     '                 &b100000000000000000
If Alarm(19) Then Flags = Flags OR &h40000     '                &b1000000000000000000
If Alarm(20) Then Flags = Flags OR &h80000     '               &b10000000000000000000
If Alarm(21) Then Flags = Flags OR &h100000    '              &b100000000000000000000
If Alarm(22) Then Flags = Flags OR &h200000    '             &b1000000000000000000000
If Alarm(23) Then Flags = Flags OR &h400000    '            &b10000000000000000000000
If Alarm(24) Then Flags = Flags OR &h800000    '           &b100000000000000000000000
If Alarm(25) Then Flags = Flags OR &h1000000   '          &b1000000000000000000000000
If Alarm(26) Then Flags = Flags OR &h2000000   '         &b10000000000000000000000000
If Alarm(27) Then Flags = Flags OR &h4000000   '        &b100000000000000000000000000
If Alarm(28) Then Flags = Flags OR &h8000000   '       &b1000000000000000000000000000
If Alarm(29) Then Flags = Flags OR &h10000000  '      &b10000000000000000000000000000
If Alarm(30) Then Flags = Flags OR &h20000000  '     &b100000000000000000000000000000
If Alarm(31) Then Flags = Flags OR &h40000000  '    &b1000000000000000000000000000000
If Alarm(32) Then Flags = Flags OR &h80000000  '   &b10000000000000000000000000000000
```

Примечание. &HFF = &B11111111. При переносе 8 бит с шагом по 32 битам Flags (тип данных Long), первые 8 битов в четырех переменных Long FlagsBool8(4) (загружаются с предупредительным сигналом. Только первые 8 бит каждой переменной Long FlagsBool8' сохраняются при преобразовании в Bool8.

Логическое И побитовое сравнение

```
' Если бит в   ИЛИ бит в     Результат
' флажках =    двоич./шест-  =
               надц. =
' ----------   ----------    ----------
' 0            0             0
' 0            1             0
' 1            0             0
' 1            1             1
```

```
    FlagsBool8(1) = Flags AND &HFF            И 1-е 8 бит флагов & 11111111
    FlagsBool8(2) = (Flags >> 8) AND &HFF     И 2-е 8 бит флагов & 11111111
    F1agsBool8(3) = (Flags >> 16) AND &HFF    И 3-е 8 бит флагов & 11111111
    FlagsBool8(4) = (Flags >> 24) AND &HFF    И 4-е 8 бит флагов & 11111111

    CallTable(Bool 8Data)
  NextScan
EndProg
```

7.9.10. Выходные данные: использование типа данных NSEC

Данные типа NSEC хранятся только в памяти окончательном данных. Тип данных NSEC из восьми байт — четыре байта для секунд, прошедших с 1990 года, и четыре байта для обозначения наносекунды в текущей секунде. ***Nsec*** объявляется в параметре ***Data Type*** (Тип данных) в *памяти окончательных данных для команд по обработке выходных данных* *(стр. 542)*. Он используется в следующих случаях:

- Размещение метки времени на второй позиции в записи.
- Доступ к временной метке из таблицы данных и сохранение ее, как части большей таблицы данных. Команды **Maximum(), Minimum()** и **FileTime()** создают временную метку, к которой можно получить доступ из программы после ее записи в таблицу данных. Время других событий, например сигналов тревоги, можно сохранить с помощью команды **RealTime()**.
- Доступ к временной метке и возможность ее сохранения с другого регистратора данных в сети PakBus.

7.9.10.1. Варианты использования NSEC

NSEC используется в программе CRBasic одним из следующих способов. Во всех случаях переменная времени получается только с помощью команды **Sample()** с параметром ***Reps = 1.***

1. Переменная времени объявляется с помощью типа **As Long.** Команда **Sample()** предполагает, что переменная времени содержит количество секунды с 1990 года, при этом количество микросекунд в секунде равно 0. Значение, сохраненное в память окончательных данных, является стандартной временной меткой. См. пример CRBasic *NSEC — одноэлементный массив времени* *(стр 202.)*.
2. Для массива временных переменных с размерностью (2) и типом **As Long** в команде **Sample()** предполагается, что первый элемент массива временных переменных содержит количество секунд с 1990 года, а второй — количество микросекунд в секунде. См. пример CRBasic *NSEC — двухэлементный массив времени* *(стр. 203)*.
3. Для массива временных переменных с размерностью (7) или (9) и типом **As Long или As Float** в команде **Sample()** предполагается, что данные в массиве переменных хранятся в последовательности: год, месяц, день года, час, минуты, секунды и миллисекунды. См. пример CRBasic *SEC— массивы времени с семью и девятью элементами* *(стр 204)*.

В примере CRBasic *NSEC — преобразование метки времени в универсальное время* *(стр. 202)* показан один из нескольких вариантов практического использования типа данных NSEC.

CRBasic: пример 35. NSEC — одноэлементный массив времени

```
В этом примере программы показано использование типа данных NSEC для определения количества
секунд с 00:00:00 1 января 1990 года. Метка времени извлекается в переменную TimeVar(1), как
количество секунд начиная с 00:00:00 1 января 1990 года. Поскольку размерность переменной равна
1, NSEC предполагает, что значение = количество секунды с 00:00:00 1 января 1990 года.

Объявления
Public PTemp
Public TimeVar(1) As Long

DataTable(FirstTable,True, -1)
  DataInterval(0,1,Sec,10)
  Sample(1,PTemp,FP2)
EndTable

DataTable(SecondTable,True,-1)
  DataInterval(0,5,Sec,10)
  Sample(1,TimeVar, Nsec)
EndTable

Программа
BeginProg
  Scan(1,Sec,0,0)
    TimeVar = FirstTable.TimeStamp
    CallTable FirstTable
    CallTable SecondTable
  NextScan
EndProg
```

CRBasic: пример 36. NSEC — двухэлементный массив времени

```
В этом примере программы показано, как определить количество секунд с 00:00:00 1 января 1990
года
и количество микросекунд за последнюю секунду. Это делается путем извлечения переменной
TimeStamp в
переменные TimeOfMaxVar(1) и TimeOfMaxVar(2). Так как размерность переменной TimeOfMaxVar()
равна 2, NSEC предполагает следующее:
1) TimeOfMaxVar(1) = кол-во секунд с 00:00:00 1 января 1990 года и,
2) TimeOfMaxVar(2) = кол-во микросекунд в секунде.

Объявления
Public PTempC
Public MaxVar
Public TimeOfMaxVar(2) As Long

DataTable(FirstTable,True, -1)
  DataInterval(0,1,Min,10)
  Maximum(1,PTempC,FP2,Fa1se,True)
EndTable

DataTable(SecondTable,True,-1)
  DataInterval(0,5,Min,10)
  Sample(1,MaxVar,FP2)
  Sample(1,TimeOfMaxVar,Nsec)
EndTable
```

```
Программа
BeginProg
  Scan(1,Sec,0,0)

    PanelTemp(PTempC,250)
    MaxVar = FirstTable.PTempC_Max
    TimeOfMaxVar = FirstTable.PTempC_TMx
    CallTable FirstTable
    CallTable SecondTable

  NextScan
EndProg
```

CRBasic: пример 37. NSEC — массивы времени с семью и девятью элементами

```
В этом примере программы показано использование типа данных NSEC для определения временной метки
в памяти окончательных данных с использованием массива размерностью до 7 или 9.

Метка времени извлекается в переменную rTime(1) с помощь rTime(9) – год, месяц, день,
час, минута, секунда и микросекунда с помощью команды RealTime(). Первые
семь значений копируются в переменную rTime2(1) из rTime2(7). Поскольку
переменные имеют размерность 7 или выше, NSEC принимает первые семь параметров времени
в массиве за год, месяц, день, час, минуту, секунду и микросекунду.

Объявления
Public rTime(9) As Long                    (или Float)
Public rTime2(7) As Long                   (или Float)
Dim x

DataTable(SecondTable,True,-1)
  DataInterval(0,5,Sec,10)
  Sample(1,rTime,NSEC)
  Sample(1,rTime2,NSEC)
EndTable

Программа
BeginProg
  Scan(1,Sec,0,0)

    RealTime(rTime)
    For x = 1 To 7
      rTime2(x) = rTime(x)
    Next

    CallTable SecondTable

  NextScan
EndProg
```

CRBasic: пример 38. NSEC — преобразование метки времени в универсальное время

```
В этом примере программы показано использование типа данных NSEC для преобразования временной
метки данных во всемирное время.

Применение. CR1000 необходимо отобразить всемирное время (UT) в строчном формате,
удобном для чтения человеком. CR1000 можно вычислить UT с помощью добавления соответствующего
смещения к стандартной временной сетке. Для добавления смещения необходимо преобразовать
временную метку в числовую форму, применить смещение, а полученный результат преобразовать
обратно строчную форму.

Данная операция выполнятся следующим образом:
1) Считывание Public.TimeStamp в длинную числовую переменную.
2) Сохранение переменой в виде даты типа NSEC в памяти окончательных данных.
3) Возврат в строчную форму с обозначением TableName.FieldName.

Объявления
Public UTTime(3) As String * 30
Dim TimeLong As Long
Const UTC_Offset = -7 * 3600                    Смещение -7 часов (в секундах).

DataTable(TimeTable,true, 1)
  Sample(1,TimeLong,Nsec)
EndTable

Программа
BeginProg
  Scan(1,Sec,0,0)

      1) Считывание Public.TimeStamp в числовую переменную типа LONG. Обратите внимание, что
      TimeStamp является
      системная переменная, поэтому она не объявляется.
      TimeLong = Public.TimeStamp(1,1) + UTC_Offset

      2) Сохранение переменой в виде даты типа NSEC в памяти окончательных данных.
      CallTable(TimeTable)

      3) Возврат времени в трехстрочную форму с обозначением TableName.FieldName.
      Форма 1: «мм/дд/гггг чч:мм:сс»
      UTTime(1) = TimeTable.TimeLong(1,1)
      Форма 2: «дд/мм/гггг чч:мм:сс»
      UTTime(2) = TimeTable.TimeLong(3,1)
      Форма 3: «гггг-мм-дд чч:мм:сс (дата согласно ISO 8601)»
      UTTime(3) = TimeTable.TimeLong(4,1)

  NextScan
EndProg
```

7.9.11. Выходные данные: Запись высокочастотных данных на карты памяти

Связанные темы:

- *Карта памяти (CRD: диск) — общие сведения* *(стр. 89)*
- *Карта памяти (CRD: диск) — подробная информация* *(стр. 376)*
- *Карты памяти и номера записей* *(стр. 466)*
- *Выходные данные: запись высокочастотные данные на карты памяти* *(стр. 205)*
- *Ошибки файловой системы* *(стр. 389)*
- *Устройства хранения данных — список* *(стр. 653)*
- *Примеры форматов файлов данных* *(стр. 379)*
- *Таблица устройств для хранения данных* *(стр. 373)*

Для записи высокочастотных данных временных рядов на карты памяти, особенно в приложениях для высокоскоростных измерений, лучше всего использовать команду **TableFile()** с параметром ***Option 64***. Она поддерживает карты памяти 16 ГБ или меньше и позволяет использовать небольшие файлы и файлы с изменяемым размером.

7.9.11.1. Команда TableFile() с параметром Option 64

Параметр Option 64 добавляется в качестве параметра команды CRBasic **TableFile().** Она сочетает в себе скорость и эффективность команды **CardOut()** с гибкостью команды **TableFile()**. Поддерживаются карты памяти[1] до 16 ГБ. **TableFile()** — команда CRBasic, которая создает файл из таблицы данных в памяти ЦП регистратора данных. ***Option 64*** указывает, что файл записывается в формате TOB3 исключительно на диск ЦП[2].

Синтаксис команды **TableFile()** выглядит следующим образом:

```
TableFile(FileName, Option, MaxFiles, NumRecs/
TimeIntoInterval, Interval, Units, OutStat, LastFileName)
```

где параметр ***Option*** задается аргументом ***64***. Подробное описание каждого параметра приведено в *Справка к редактору CRBasic*[3]**.**

Примечание. Диск CRD (обозначение для дополнительной карты памяти) является единственным диском, для которого разрешено использовать параметр ***Option 64***.

Примечание. Использование карт памяти повышает сохранность добавляемых данных. Однако во всех системах существуют ошибки. Считать карту памяти после того, как лось потоптался внутри метеостанции или трактор раскопал станцию контроля влажности почвы, может быть затруднительно. Поэтому лучше всего считывать данные CR1000 до того, как накопится такой объем данных, который вы не можете себе позволить потерять. Другими словами, если вы можете позволить себе потерять данные за весь месяц, то считывать данные лучше чаще чем один раз в месяц.

[1] Карты памяти, используемые в регистраторе CR1000, представляют собой карты типа (CF).

[2] Диск CRD: является диском памяти, который создается при подсоединении карты к CR1000 при посредстве соответствующего периферийного устройства. Регистратор CR1000 адаптирован под применение карт CF путем добавления модулей NL115 или CFM100. Модули NL115 и CFM100 приобретаются в компании Campbell Scientific за дополнительную плату.

[3] Редактор *CRBasic Editor* входит в пакеты *ПО поддержки регистратора данных Campbell Scientific* *(стр. 95)* *LoggerNet*, *PC400* и *RTDAQ*.

7.9.11.2. Команда TableFile() с параметром Option 64 как замена команды CardOut()

Для большинства случаев применения команда TableFile() с параметром ***Option 64*** имеет ряд преимуществ перед командой **CardOut()**. Сюда входит следующее:

- Возможность написания нескольких небольших файлов из одной таблицы данных, чтобы хранилище для одной таблицы могло превышать 2 ГБ. Команда **TableFile()** задает размер этих выходных файлов посредством параметров ***NumRecs, TimeIntoInterval*** и ***Interval***.
- Сокращенная длительность компиляции, если задаются небольшие размеры файлов.

- Простая выгрузка закрытых файлов при помощи сервисной программы **File Control** *(стр. 515)*, FTP или электронной почты.

7.9.11.3. Программирование команды TableFile() с параметром Option 64

Как показано в следующем фрагменте кода CRBasic, команда **TableFile()** должна размещаться в объявлении **DataTable() / EndTable**. Команда **TableFile()** записывает данные на карту памяти исходя из задаваемых пользователем параметров, определяющих размер файла на основе количества записей для хранения или интервала, в течение которого данные должны храниться. Результирующий файл сохраняется с суффиксом X.dat, где X — номер, увеличивающийся на единицу при каждом записывании нового файла.

```
DataTable(TableName,TriggerVariable, Size)
   TableFile(FileName...LastFileName)
   Здесь указываются команды по обработке выходных данных
EndTable
```

Например, в случае микрометеорологического применения команда **TableFile()** с параметром ***Option 64*** используется для ежедневного создания нового файла по высокочастотным данным. Размер создаваемого файла зависит от частоты сканирования регистратора данных и количества переменных, сохраненных в таблице данных. Для типовой микрометеорологической станции размер такого ежедневного файла составляет около 50 МБ (частота сканирования 10 Гц и 15 измерительных точек формата IEEE4). Пример CRBasic *Примером микрометеорологического примера является использование команды TableFile() с параметром Option 64 с применением карт CF* *(стр. 207)*.

CRBasic: пример 39. Использование команды TableFile() с параметром Option 64 с картой CF

```
В данном примере программы продемонстрировано использование команды TableFile() с параметром
Option 64 в программах для микрометеорологических станций, использующих метод турбулентной
ковариации. Схема именования файлов, используемая для команды TableFile(), формируется с
применением переменных, констант и текстовой информации.

Public Sensor(10)

DataTable(Ts_data,TRUE,-1)
   TableFile("filename",Option,MaxFiles,NumRec/TimeIntoInterval,Interval, Units,
   OutStat,LastFileName)
   TableFile("CRD:"&Status.SerialNumber(1,1)&".ts_data_",64,-1,0,1,Day,0,0)
   Sample(10,sensor(1),IEEE4)
EndTable

BeginProg
   Scan(100,mSec,100,0)
     Здесь указываются команды для измерения.
     Здесь указываются команды по обработке.
     CallTable ts_data
   NextScan
EndProg
```

7.9.11.4. Преобразование файлов TOB3 с помощью CardConvert

Формат TOB3 используется для записи данных на карты памяти, за счет чего обеспечивается экономия дискового пространства. Однако результирующие двоичные файлы должны быть конвертированы в другой формат, чтобы их могли считывать или использовать другие программы. Программное обеспечение CardConvert, входящее в пакет *ПО поддержки регистратора данных Campbell Scientific* *(стр. 95)*, обеспечивает такое преобразование между форматами. В справке по программе *CardConvert* содержится более подробная информация.

7.9.11.5. Команда TableFile() с параметром Option 64 — вопросы и ответы

В: Чем параметр ***Option 64*** отличается от других параметров **TableFile()**?

А: Предварительное распределение памяти включает в себя формат данных TOB3, что дает параметру ***Option 64*** два важных преимущества перед другими параметрами **TableFile()**. А именно:

- повышенная производительность записи в среде выполнения;
- короткий промежуток времени извлечения карты.

Option 64 является уникальным параметром среди других параметров файла таблицы в том, что выделяет достаточное пространство на карте памяти для хранения интервала данных[1]. Предварительное распределение памяти обеспечивает непрерывную запись данных с повышенной скоростью на карте блоками размером ≈1 кБ. Двоичный формат TOB3 копирует данные непосредственно из памяти CPU на карту памяти без конвертации формата, дополнительно повышая скорость и производительность процесса сохранения данных.

Примечание. Предварительное распределение файлов карты памяти существенно повышает производительность записи в среде выполнения. Оно также снижает риск повреждения файла, которое может иметь место в результате пропадания питания или неправильного извлечения карты.

Примечание. Во избежание повреждения и потери данных извлечение карты памяти должно осуществляться путем нажатия кнопки **Initiate Removal** (Начать извлечение) на лицевой панели модулей NL115 или CFM100. Карту необходимо извлекать только после того, как индикатор **Status** (Состояние) непрерывно горит зеленым цветом.

В: Почему размер отдельных файлов ограничивается 2 ГБ?

А: Так же, как и во множестве других систем, регистратор данных обладает встроенной поддержкой целые четырехбайтные числа со знаком. Этот тип данных может представлять число размером до 231 или, применительно к байтам, приблизительно 2 ГБ. Это максимальная длина файла, которая может быть представлена в таблице каталогов регистратора данных.

В: Почему использование карты большого объема является причиной длительных периодов компиляции программы?

А: Длительность компиляции увеличивается в зависимости от размера карты и файла. При загрузке регистратора данных производится поиск на карте для определения доступного места для хранения данных. Кроме того, для таблиц, созданных командой **TableFile()** с параметром ***Option 64***, должен создаваться пустой файл размером, достаточным для хранения всех заданных записей (т. е. память должна быть предварительно распределена). Если используется команда **TableFile()** с параметром ***Option 64***, длительность компиляции программы может быть сокращена путем уменьшения количества записей или интервала вывода данных, который будет включен в каждый файл. Например, если для файла установлен максимальный размер 2 ГБ, регистратор данных должен выполнить сканирование и предварительно выделить 2 ГБ на карте памяти CF. С другой стороны, если задается меньший размер файлов, то длительность компиляции сокращается ввиду того, что регистратору данных необходимо только выполнить сканирование памяти для предварительного выделения достаточного пространства в памяти под файлы меньшего размера.

В: Почему использование только что отформатированной карты приводит к продолжительным периодам компиляции?

А: Компиляция программы при использовании только что отформатированной карты выполняется в течение более длительного промежутка времени, потому что на картах используется файловая система FAT32 (таблица распределения файлов с 32-разрядными элементами таблицы), чтобы обеспечить

совместимость с ПК. Во избежание длительной компиляции на вновь отформатированной карте отформатируйте карту на ПК, затем скопируйте небольшой файл на карту, а затем удалите его (не извлекая карту из ПК). При копировании файла на вновь отформатированную карту ПК обновляет информационный сектор. ПК обновляет информационный сектор гораздо быстрее, чем регистратор данных.

В системе FAT32 используется «информационный сектор» для хранения информации о свободных кластерах. Данный информационный сектор исключает необходимость циклического обхода системы FAT для получения информации по свободным байтам. После форматирования карты компьютером информационный сектор не обновляется автоматически. По этой причине регистратор данных при загрузке должен обнаружить доступные байты на карте перед загрузкой таблицы **Status** (Состояние). Для обхода всей файловой системы FAT карты объемом 16 ГБ может занять до 30 и более минут. Однако при последующей компиляции длительность будет гораздо короче, поскольку информационный сектор обновил информацию о свободных байтах.

B: Какую карту памяти необходимо использовать?

A: Campbell Scientific рекомендует и поддерживает только карты CF производства компании FMJ. Данные карты представляют собой карты промышленного класса и прошли аппаратное тестирование со стороны Campbell Scientific. Вышеназванные карты обладают следующими преимуществами перед менее дорогими картами коммерческого класса:

- пониженная подверженность отказам и потере данных;
- соответствие диапазону рабочих температур регистратора данных;
- повышенная скорость чтения/записи;
- повышенная устойчивость к воздействию вибрации и ударным нагрузкам;
- увеличенный срок службы (увеличенное число циклов чтения/записи).

Примечание. Расширенные рекомендации по применению карт CF представлены в указаниях по применению, *CF Card Information (Информация по картам CF)* на сайте *www.campbellsci.com.*

B: Можно ли дистанционно выгрузить закрытые файлы?

A: Да. Закрытые файлы можно получить с помощью функции **Retrieve** (Получить) в ПО поддержки регистратора данных — сервисная программа *File Control (Управление файлами)* *(стр. 515)*, FTP, HTTP или электронная почта. Несмотря на то, что открытые файлы появятся в каталоге диска CRD:, не пытайтесь получить открытые файлы. Это может привести к повреждению файла и, как следствие, потере данных. Файлы меньшего размера обычно передаются с большей скоростью и надежностью в сравнении с файлами большого размера.

B: Можно ли получить доступ к данным?

A: Да. Данные в открытом или в наиболее новом файле можно получить при помощи сервисных программ **Collect** (Сбор) или **Custom Collect** (Выборочный сбор) в программном обеспечении *LoggerNet, PC400* или *RTDAQ*. Данные также можно просмотреть при помощи ПО поддержки регистратора данных или получить к ним доступ посредством регистратора данных при помощи синтаксиса доступа к таблице данных, например **TableName.FieldName** (см. справку по редактору *CRBasic*). Если файл был закрыт, доступ к данным можно получить, только получив файл, как рассматривалось выше, и выполнив обработку файла с помощью программного обеспечения *CardConvert.*

B: Что происходит, когда вставляется карта?

A: Как только вставляется карта, независимо от того, новая она или бывшая в употреблении, всегда создается новый файл.

B: Какие функции выполняют команды выключения-включения питания или перезапуска программы?

A: При каждом запуске программы, независимо от того, запускается она пользователем, при включении-выключении питания или сторожевой схемой, команда **TableFile()** с параметром ***Option 64*** создает новый файл.

B: Что происходит при заполнении карты?

A: После заполнения карты памяти новые данные записываются поверх старых данных. Карту необходимо заменить до того, как она заполнится. В противном случае самые старые данные будут перезаписаны входящими новыми записями и, следовательно, потеряны. При замене карты, как только старая карта будет извлечена, необходимо вставить новую карту до того, как таблица данных в памяти ЦП регистратора данных не будет «закольцована»[2]. В противном случае данные будут перезаписаны и, следовательно, потеряны. Например, представьте случай применения, при котором таблица данных в памяти ЦП регистратора данных имеет объем, достаточный для сохранения данных в течение приблизительно 45 минут[3]. Замена карты должна быть произведена в любой момент времени до истечения этих 45 минут. Если замена была произведена с задержкой в 5 минут, 5 минут данных в начале последнего 45-минутного интервала (поскольку это самые старые данные) будут записаны с заменой в памяти ЦП до того, как они будут переданы на новую карту и потеряны.

[1] Другие параметры команды **TableFile()** не осуществляют предварительное распределение памяти, поэтому их не следует использовать при сборе данных временного ряда с высокой частотой обновления. Более подробная информация приводится в справке по редактору *CRBasic*.

[2] «Закольцована»: регистратор данных имеет кольцевую память. Другими словами, после заполнения новые данные записываются с заменой старых, т. е. запись данных не прекращается при заполнении. В этом контексте под «кольцом» подразумевается запись новых данных с заменой старых.

[3] Время заполнения таблицы данных ЦП можно проверить в таблице **Status** (Состояние) регистратора данных.

7.9.12. Калибровка на месте — описание

Связанные темы:

- *Калибровка на месте — общие сведения* *(стр. 73)*
- *Калибровка на месте — подробная информация* *(стр. 210)*

Калибровка увеличивает точность датчика, регулируя или корректируя его выходные параметры таким образом, чтобы они соответствовали величинам, измеренным независимым прибором. Непосредственная регулировка выходного сигнала датчика является предпочтительным вариантом, но он не всегда доступен. Если используется команда **FieldCal()** или **FieldCalStrain()**, линейный выход датчика можно скорректировать в CR1000 после измерения, отрегулировав множитель и сдвиг.

Если команды **FieldCal()** и **FieldCalStrain()** в программу CRBasic, их можно использовать с помощью *мастера калибровки ПО поддержки регистратора данных* *(стр. 509)*. В данном программном обеспечении имеется справка по использованию мастера.

Еще более малоизвестную процедуру, для которой не требуется ПК, можно выполнить с помощью экранной клавиатуры/дисплея CR1000KD. Если клавиатура отсутствует, эту же процедуру можно выполнить при помощи *цифрового монитора* *(стр. 521)*. Захваты изображения экрана используются в следующих процедурах. Выполнение данных процедур дает понимание того, как выполнять калибровку по месту, однако в повседневной работе рекомендуется использование мастера калибровки.

Сводная информация по синтаксису команд **FieldCal()** и **FieldCalStrain()** представлена в разделе *Функции клабировки* *(стр. 598)*. Более подробная информация приводится в справке по редактору *CRBasic*.

7.9.12.1. Калибровка на месте — файлы CAL

Данные калибровки сохраняются автоматически, обычно на диске CPU:

регистратора CR1000 в файлах CAL (.cal). Эти данные используются в качестве основы для коэффициентов калибровки при запросе от команды **LoadFieldCal()**. Файл создается автоматически на том же накопителе регистратора CR1000, и ему присваивается то же имя, что и программа, которая создает и использует его. Например, программный файл CPU:MyProg.cr1 в программе CRBasic генерирует файл CAL CPU:MyProg.cal.

Файлы CAL создаются, если программа, которая использует команды **FieldCal()** или **FieldCalStrain()**, не находит существующего совместимого файла CAL. Файлы обновляются при каждой успешной калибровке путем создания новых коэффициентов калибровки. История калибровки регистрируется, только если программа CRBasic создает *таблицу данных* *(стр. 512)* посредством команды **SampleFieldCal()**.

Примечание. Файлы CAL, создаваемые командами **FieldCal()** и **FieldCalStrain()**, отличаются от файлов, созданных командой **CalFile()** (*Управление файлами* *(стр. 382)*).

7.9.12.2. Калибровка на месте — программирование

Функции калибровки по месту представлены в программе CRBasic следующими командами:

- **FieldCal()** — основная инструкция, используемая для датчиков, отличных от тензодатчиков. В ознакомительных целях используйте одну команду **FieldCal()** и уникальный набор переменных **FieldCal()** для каждого датчика. Для более сложных случаев применения используйте массивы переменных.
- **FieldCalStrain()** — основная команда, используемая для тензодатчиков, измеряющих микродеформацию. Используйте одну команду **FieldCalStrain()** и уникальный набор переменных **FieldCalStrain()** для каждого датчика. Для более сложных случаев применения используйте массивы переменных.

FieldCal() и **FieldCalStrain()** используют следующие команды:

- **LoadFieldCal()** — дополнительная команда, выполняющая оценку действительности и сами значения нагрузки из файла CAL.
- **SampleFieldCal** — дополнительная команда для хранения выходных данных, которая записывает последние данные калибровки в таблицу данных (а не в файл CAL).

FieldCal() и **FieldCalStrain()** используют следующие зарезервированные логические переменные:

- **NewFieldCal** — зарезервированная логическая под управлением CR1000, используемая для дополнительного однократного запуска выходной таблицы хранения данных после успешной калибровки.

См. справку по редактору *CRBasic* для получения оперативных данных по командам CRBasic.

7.9.12.3. Калибровка на месте — общее описание мастера калибровки

Мастера калибровки по месту *LoggerNet* и *RTDAQ* помогают в выполнении процедуры, автоматически внося изменения в переменные режима и осуществляя измерения. Вам необходимо установить датчик на известные значения и ввести данные значения в мастер.

Если выполняется программа с командами **FieldCal()** или **FieldCalStrain()**, выберите *LoggerNet* или *RTDAQ* | **Datalogger | Calibration Wizard**, чтобы запустить мастер. Отображается список использованных измерений.

Для получения дополнительной информации по использованию мастера ознакомьтесь со справочной информацией по *LoggerNet* или *RTDAQ*.

7.9.12.4. Калибровка на месте — процедуры мониторинга данных

В данном разделе рассматривается ручная калибровка на месте при помощи

цифрового монитора (вместо клавиатуры/дисплея CR1000KD), чтобы познакомить пользователя с использованием и принципом работы команд **FieldCal()** и **FieldCalStrain()**. Данный раздел не предоставляет обширного обзора по всем темам, связанным с калибровкой на месте. На текущий момент самым детальным ресурсом, в котором рассматривается использование команд **FieldCal()** и **FieldCalStrain()**, является документация по программному обеспечению *RTDAQ*, которую можно скачать на сайте *www.campbellsci.com http://www.campbellsci.com.* Обратите внимание на следующие меры предосторожности:

- Регистратор CR1000 не проверяет наличие выходящих за пределы допустимого диапазона значений в переменных режима.
- Действительными записями переменных режима являются **1** или **4**.

Перед, во время и после калибровки один из следующих кодов сохраняется в переменной **CalMode**:

Таблица 28. Коды FieldCal()

Обратное значение	*Состояние*
–1	Ошибка в настройках калибровки
–2	Множитель установлен на ***0*** или ***NAN***; measurement = **NAN**
–3	***Reps*** установлен на значение, отличное от ***1*** или от размера ***MeasureVar***
0	Нет калибровки
1	Готовность к вычислению (***KnownVar*** содержит первую из двух точек калибровки)
2	Выполняется
3	Первая точка завершена (применяется только к двухточечной калибровке)
4	Готовность к вычислению (***KnownVar*** содержит вторую из двух точек калибровки)
5	Выполняется (применяется только к двухточечной калибровке)
6	Калибровка завершена

7.9.12.4.1. Одноточечная калибровка (с нулем или со смещением)

Калибровка нуля применяется к смещению равной величины, но противоположной по знаку. Например, если выполняется калибровка нуля по измерению 15,3, значение «–15,3» будет добавляться к последующим измерениям.

Калибровка смещения применяется к смещению равной величины и того же знака. Например, если выполняется калибровка смещения по измерению 15,3, значение «15,3» будет добавляться к последующим измерениям.

См. *Пример калибровки FieldCal() с нулем или Tare (Opt 0)* *(стр. 214)* и *Пример калибровки FieldCal() смещения (Opt 1)* *(стр. 216)* для демонстрационных программ:

1. Используйте отдельную команду и переменные **FieldCal()** для каждого датчика, который необходимо откалибровать. В программе CRBasic поместите команду **FieldCal()** непосредственно под связанной командой измерения.
2. Установите переменную режима = 0 или 6 перед запуском.
3. Установите датчик в нулевое или смещенное состояние.
4. Задайте для переменной ***KnownVar*** смещенное или нулевое значение.
5. Установите для переменной режима значение = **1** для запуска калибровки.

7.9.12.4.2. Двухточечная калибровка (усиление и смещение)

Используйте данную процедуру двухточечной калибровки для регулировки множителей (уклоны) и смещений (пересечения с осью y). См. *Пример калибровки FieldCal() наклона и смещения (Opt 2)* *(стр. 218)* и *Пример калибровки наклона FieldCal() (Opt 3)* *(стр. 220)* для демонстрационных программ:

1. Используйте отдельную команду и отдельные переменные **FieldCal()** для каждого датчика, который необходимо откалибровать.
2. Установите переменную режима = **0** или **6** перед запуском.
    a. Если **Mode (Режим) > 0** и **≠ 6**, калибровка находится в процессе выполнения.
    b. Если **Mode (Режим) < 0**, в ходе калибровки была обнаружена ошибка.
3. Переведите датчик в состояние первой известной точки.
4. Установите для переменной KnownVar значение первой известной точки.
5. Установите для переменной ***Mode*** (Режим) = **1** для запуска первой части калибровки.
    a. ***Mode* = 2** (автоматический) во время калибровки первой точки.
    b. ***Mode* = 3** (автоматический), когда калибровка первой точки завершена.
6. Переведите датчик в состояние второй известной точки.
7. Установите для переменной KnownVar значение второй известной точки.
8. Задайте для переменной ***Mode* = 4**, чтобы запустить вторую часть калибровки.
    a. ***Mode* = 5** (автоматический) во время калибровки второй точки.
    b. ***Mode* = 6** (автоматический), когда калибровка второй точки завершена.

7.9.12.4.3. Калибровка нулевой базовой точки

Калибровка нулевой точки (команда **FieldCal()** ***(Option 4***)) предназначена для использования при измерениях статичных струнных датчиков. Она загружает значения в переменные нулевой точки для отслеживания состояния на момент калибровки нуля. См. *Пример команды FieldCal() Zero Basis (Opt 4)* *(стр. 223)* для демонстрации программы.

7.9.12.5. Примеры калибровки

Команда FieldCal() предусматривает следующие варианты калибровки:

- Нуль
- Смещение
- Двухточечный: наклон и смещение
- Двухточечный: только наклон
- Базовый нуль (предназначена для использования со статичными струнными датчиками)

Данные демонстрационные программы представлены для ознакомления с функциями команды **FieldCal()** при тестировании без использования действующих датчиков. В демонстрационных целях сигналы датчиков моделируются клеммами регистратора CR1000, настроенными на возбуждение. Чтобы сбросить тесты, используйте команды меню ПО поддержки *File Control (Управление файлами)* *(стр. 515)* для удаления файлов .cal, а затем снова отправьте демонстрационную программу в CR1000. Эквивалентами термина являются следующие:

"offset" = "y- intercept" = "zero" (смещение = пересечение с осью y = нуль)
"multiplier" = "slope" = "gain" (множитель = наклон = усиление)

7.9.12.5.1. Пример калибровки FieldCal() с нулем или Tare (Opt 0)

Большинство команд измерения в CRBasic имеют параметры ***multiplier*** (множитель) и ***offset*** (смещение). Команда **FieldCal() с параметром *Option 0*** настраивает аргумент ***offset*** (смещение) так, что выход калибруемого датчика устанавливается на значение параметра **FieldCal()** ***KnownVar***, который имеет значение **0**. Из последующих измерений вычитается аналогичное смещение. Параметр ***Option 0*** не влияет на аргумент ***multiplier*** (множитель).

Пример. Датчик измеряет относительную влажность (ОВ) воздуха. Известно, что множитель стабилен, но смещение датчика сдвигается и требует регулярной калибровки нуля в камере осушения. В результате следующей процедуры датчик ОВ обнуляется, чтобы получить представленный отчет о калибровке. Чтобы продолжить выполнение данного примера, измените значения переменных, как указано, с помощью экранной клавиатуры CR1000KD или программного *цифрового монитора* *(стр. 521)***.**

Таблица 29. Отчет о калибровке датчика относительной влажности

Переменная CRBasic	***При развертывании***	***При 30-дневной эксплуатации***
SimulatedRHSignal – выход	***100*** мВ	***105*** мВ
KnownRH (камера осушения)	***0*** %	***0*** %
RHMultiplier	***0,05*** %/мВ	***0,05*** %/мВ
RHOffset	**–5** %	**–5,25** %
RH	**0** %	**0** %

1. Отправьте пример CRBasic *FieldCal() Zero* *(стр. 214)* в CR1000. Клемма, настроенная на возбуждения, была запрограммирована на моделирование выхода датчика.
2. Чтобы создать для смоделированного датчика ОВ смоделированные условия калибровки (в реальных условиях датчик помещается в камеру осушения), установите проволочную перемычку между клеммами **VX1** и **SE1**. Следующие параметры имеют значения, предварительно заданные программой: ***SimulatedRHSignal = 100, KnownRH = 0.***
3. Чтобы запустить калибровку, установите для переменной ***CalMode = 1***. Когда переменная ***CalMode*** увеличится до **6**, калибровка нуля будет завершена. Откалиброванное значение ***RHOffset*** будет равняться **–5** % на этом этапе примера.
4. Чтобы продолжить данный пример и смоделировать состояние сдвига нуля, задайте для переменной ***SimulatedRHSignal = 105.***
5. Чтобы смоделировать условия для калибровки через 30 дней эксплуатации, опять же исходя из условий камеры осушения, установите для переменной ***KnownRH = 0.0.*** Чтобы запустить калибровку, установите для переменной ***CalMode = 1***. Когда переменная ***CalMode*** увеличится до **6**, калибровка нуля через 30 дней эксплуатации будет завершена. Откалиброванное значение ***RHOffset*** будет равняться **–5,2** %.

CRBasic: пример 40. FieldCal() — нулевое значение

```
'Этот пример программы демонстрирует использование команды FieldCal() при расчете и применении
'калибровки нуля. При калибровке нуля измеряется амплитуда сигнала датчика, нулевое состояние
'которого известно, и вычисляется отрицательная амплитуда для использования в качестве значения
'смещения при последующих измерениях.
'@@@condition and calculates the negative magnitude to use as an offset in subsequent
'@@@measurements. Это не влияет на множитель.

'В этой программе показана калибровка нуля в соответствии со следующей процедурой:
'-- Моделирование сигнала от датчика относительной влажности.
'-- Измерение сигнала «датчика».
'-- Вычисление и применение калибровки нуля.

'Можно использовать настройку, при которой моделирование будет осуществляться при загрузке данной
'программы в CR1000 и при соединении следующих клемм при помощи перемычки, чтобы смоделировать
'сигнал датчика относительной влажности следующим образом:
'@@@the following terminals with a jumper wire to simulate the relative-humidity sensor signal
'@@@as follows:
' Vx1 --- SE1

'В целях моделирования исходный сигнал «датчика» устанавливается автоматически. Запустите
'подпрограмму для калибровки нуля, установив переменную CalMode = 1.
'@@@by setting variable CalMode = 1. Если CalMode = 6 (достигается автоматически через 10
'измерений), подпрограмма завершается. Запишите новое значение в переменной RHOffset. Теперь
'введите значение напряжения в милливольтах, аналогичное сигналу смоделированного датчика, и
'запишите, каким образом новое значение смещения добавляется к измерению:
'@@@enter the following millivolt value as the simulated sensor signal and note how the new
'@@@offset is added to the measurement:
'SimulatedRHSignal = 1000

'ПРИМЕЧАНИЕ. Данная программа помещает файл .cal на диск CPU: регистратора CR1000. Файл .cal
'должен быть удален для сброса демонстрации.
'@@@be erased to reset the demonstration.

'ОБЪЯВИТЕ ПЕРЕМЕННУЮ СМОДЕЛИРОВАННОГО СИГНАЛА И УСТАНОВИТЕ ИСХОДНУЮ АМПЛИТУДУ МИЛЛИВОЛЬТНОГО
'СИГНАЛА
Public SimulatedRHSignal = 100

'ОБЪЯВИТЕ СТАНДАРТНУЮ ПЕРЕМЕННУЮ КАЛИБРОВКИ И УСТАНОВИТЕ АМПЛИТУДУ ОТНОСИТЕЛЬНОЙ ВЛАЖНОСТИ В
'ПРОЦЕНТАХ
Public KnownRH = 0

'ОБЪЯВИТЕ ПЕРЕМЕННУЮ РЕЗУЛЬТАТА ИЗМЕРЕНИЙ.
Public RH

'ОБЪЯВИТЕ ПЕРЕМЕННУЮ РЕЗУЛЬТАТА СМЕЩЕНИЯ.
Public RHOffset

'ОБЪЯВИТЕ ПЕРЕМЕННУЮ ДЛЯ КОНТРОЛЯ FieldCal()
Public CalMode

'ОБЪЯВИТЕ ТАБЛИЦУ ДАННЫХ ДЛЯ ВЫГРУЖАЕМЫХ РЕЗУЛЬТАТОВ КАЛИБРОВКИ
DataTable(Cal Hist,NewFieldCal,200)
   SampleFieldCal
EndTable

BeginProg
   'ЗАГРУЗИТЕ КОНСТАНТЫ КАЛИБРОВКИ ИЗ ФАЙЛА CPU:CALHIST. CAL
   'Выполняется после процедуры калибровки нуля (если переменная CalMode = 6)
   LoadFieldCal (true)
```

```
  Scan(100,mSec,0,0)

    СМОДЕЛИРУЙТЕ СИГНАЛ, А ЗАТЕМ ВЫПОЛНИТЕ ИЗМЕРЕНИЕ
    Калибровка нуля применяется, когда значение переменной CalMode = 6
    ExciteV(Vx1,Si mulatedRHSignal,0)
    VoltSE(RH,1,mV2500,1,1,0,250,0.05,RHOffset)

    ВЫПОЛНИТЕ КАЛИБРОВКУ НУЛЯ.
    Начните с присвоения переменной значения CalMode = 1. Процедура завершена, когда значение
    переменной CalMode = 6.
    FieldCal(Function, MeasureVar, Reps, MultVar, OffsetVar, Mode, KnownVar, Index, Avg)
    FieldCal(0,RH,1,0,RHOffset,Cal Mode,KnownRH,1,30)

    Если была произведена калибровка, сохраните значения калибровки в таблицу данных CalHist
    CallTable(CalHist)

  NextScan
EndProg
```

7.9.12.5.2. Пример калибровки FieldCal() смещения (Opt 1)

Большинство команд измерения в CRBasic имеют параметры ***multiplier*** (множитель) и ***offset*** (смещение). Команда **FieldCal() с параметром *Option 1*** настраивает аргумент ***offset*** (смещение) так, что выход калибруемого датчика устанавливается на величину параметра **FieldCal()** ***KnownVar***. К последующим измерениям добавляется аналогичное смещение. Параметр ***Option 0*** не влияет на аргумент ***multiplier*** (множитель). Параметр ***Option 0*** не влияет на аргумент ***multiplier*** (множитель).

Пример. Датчик измеряет минерализацию воды. Известно, что множитель стабилен, но смещение датчика сдвигается и требует регулярной корректировки на основе стандартного раствора. В результате следующей процедуры осуществляется смещение измерений, чтобы получить представленный отчет о калибровке.

Таблица 30. Отчет о калибровке датчика содержания соли

Переменная CRBasic	*При развертывании*	*Через семь дней эксплуатации*
SimulatedSalinitySignal – выход	**1350** мВ	**1345** мВ
KnownSalinity (стандартный раствор)	**30** мг/л	**30** мг/л
SalinityMultiplier	**0,05** мг/л/мВ	**0,05** мг/л/мВ
SalinityOffset	**–37,50** мг/л	**–37,23** мг/л
Salinity – показание	**30** мг/л	**30** мг/л

1. Отправьте пример CRBasic *FieldCal() Offset* *(стр. 217)* в CR1000. Клемма, настроенная на возбуждения, была запрограммирована на моделирование выхода датчика.
2. Чтобы создать для смоделированного датчика содержания солей смоделированные условия калибровки (в реальных условиях датчик помещается в стандартный раствор 30 мг/л), установите проволочную перемычку между клеммами **VX1** и **SE1**. Следующие параметры имеют значения, предварительно заданные программой: SimulatedSalinitySignal = ***1350, KnownSalinity = 30.***
3. Чтобы запустить смоделированную калибровку, установите для переменной ***CalMode = 1***. Когда переменная ***CalMode*** увеличится до ***6***, калибровка смещения будет завершена. Откалиброванное смещение будет равняться ***–37,48*** мг/л.
4. Чтобы продолжить данный пример и смоделировать состояние сдвига смещения, задайте для переменной ***SimulatedSalinitySignal = 1345.***

5. Чтобы смоделировать условия калибровки через семь дней эксплуатации (стандартный раствор 30 мг/л), для значения переменной ***KnownSalinity*** должно оставаться неизменно ***30,0***. Измените значение переменной ***CalMode*** на ***1***, чтобы запустить калибровку. Когда переменная ***CalMode*** увеличится до ***6***, калибровка нуля через семь дней эксплуатации будет завершена. Откалиброванное смещение будет равняться ***–37,23*** мг/л.

CRBasic: пример 41. FieldCal() — **смещение**

```
Этот пример программы демонстрирует использование команды FieldCal() при расчете и применении
калибровки смещения. При калибровке смещения амплитуда сигнала датчика сопоставляется с
известной стандартной амплитудой и производится расчет смещения для регулировки выхода датчика
на известное значение.
@@known standard and calculates an offset to adjust the sensor output to the known value.
Затем смещение используется для регулировки последующих значений.

В этой программе показана калибровка смещения в соответствии со следующей процедурой:
-- Моделирование сигнала от датчика содержания соли.
-- Измерение сигнала «датчика».
-- Вычисление и применение сдвига.

Можно использовать настройку, при которой моделирование будет осуществляться при загрузке данной
программы в CR1000 и при соединении следующих клемм при помощи перемычки, чтобы смоделировать
сигнал датчика содержания соли следующим образом:
@@following terminals with a jumper wire to simulate the salinity sensor signal as follows:
' Vx1 --- SE1

В целях моделирования значение калибровочного стандарта и исходный сигнал «датчика»
устанавливаются автоматически. Запустите калибровку, установив переменную CalMode = 1. Если
CalMode = 6 (достигается автоматически через 10 измерений), подпрограмма завершается.
Запишите новое значение в переменной SalinityOffset. Теперь введите значение напряжения в
милливольтах, аналогичное сигналу смоделированного датчика, и запишите, каким образом новое
значение смещения добавляется к измерению:
@@the simulated sensor signal and note how the new offset is added to the measurement:
SimulatedSalinitySignal = 1345

ПРИМЕЧАНИЕ. Данная программа помещает файл .cal на диск CPU: регистратора CR1000. Файл .cal
должен быть удален для сброса демонстрации.
@@be erased to reset the demonstration.

ОБЪЯВИТЕ ПЕРЕМЕННУЮ СМОДЕЛИРОВАННОГО СИГНАЛА И УСТАНОВИТЕ ИСХОДНУЮ АМПЛИТУДУ
Public SimulatedSalinitySignal = 1350          мг/л

ОБЪЯВИТЕ СТАНДАРТНУЮ ПЕРЕМЕННУЮ КАЛИБРОВКИ И УСТАНОВИТЕ АМПЛИТУДУ
Public KnownSalinity = 30                      мг/л

ОБЪЯВИТЕ ПЕРЕМЕННУЮ РЕЗУЛЬТАТА ИЗМЕРЕНИЙ.
Public Salinity

ОБЪЯВИТЕ ПЕРЕМЕННУЮ РЕЗУЛЬТАТА СМЕЩЕНИЯ.
Public SalinityOffset

ОБЪЯВИТЕ ПЕРЕМЕННУЮ ДЛЯ КОНТРОЛЯ FieldCal()
Public CalMode

ОБЪЯВИТЕ ТАБЛИЦУ ДАННЫХ ДЛЯ ВЫГРУЖАЕМЫХ РЕЗУЛЬТАТОВ КАЛИБРОВКИ
DataTable(CalHist,NewFieldCal,200)
   SampleFieldCal
EndTable
```

```
BeginProg
   ЗАГРУЗИТЕ КОНСТАНТЫ КАЛИБРОВКИ ИЗ ФАЙЛА CPU:CALHIST.CAL
   Выполняется после процедуры калибровки нуля (если переменная CalMode = 6)
   LoadFieldCal (true)

   Scan(100,mSec,0,0)

      СМОДЕЛИРУЙТЕ СИГНАЛ, А ЗАТЕМ ВЫПОЛНИТЕ ИЗМЕРЕНИЕ
      Калибровка нуля применяется, когда значение переменной CalMode = 6
      ExciteV(Vx1,SimulatedSalinitySignal,0)
      VoltSE(Sa1inity,1,mV2500,1,1,0,250,0.05,SalinityOffset)

      ВЫПОЛНИТЕ КАЛИБРОВКУ СМЕЩЕНИЯ.
      Начните с присвоения переменной значения CalMode = 1. Процедура завершена, когда значение
      переменной CalMode = 6.
      FieldCal(Function, MeasureVar, Reps, MultVar, OffsetVar, Mode, KnownVar, Index, Avg)
      FieldCal (1,Salinity,1,0,SalinityOffset,CalMode,KnownSalinity,1, 30)

      Если была произведена калибровка, сохраните значения калибровки в таблицу данных CalHist
      CallTable(CalHist)

   NextScan
EndProg
```

7.9.12.5.3. Пример калибровки FieldCal() наклона и смещения (Opt 2)

Большинство команд измерения в CRBasic имеют параметры ***multiplier*** (множитель) и ***offset*** (смещение). Команда **FieldCal()** ***Вариант 2*** корректирует аргументы ***множителя*** и ***смещения*** таким образом, чтобы на выходе калибруемого датчика устанавливалось значение, соответствующее линейному соотношению двух известных пересекающихся точек, последовательно указанных в параметре **FieldCal()** ***KnownVar***. Последующие измерения масштабируются с учетом тех же значений множителя и смещения.

Пример. Датчик измеряет объем воды, проходящей через трубу. Известно, что для множителя и смещения характерен дрейф, поэтому требуется периодически выполнять двухточечную калибровку по известным значениям расхода. В ходе следующей процедуры значения множителя и смещения корректируются с учетом дрейфа датчика, как показано ниже в отчете о калибровке. Обратите внимание, что выраженный в мВ результат на выходе расходомера обратно пропорционален расходу.

Таблица 31. Отчет о калибровке расходомера

Переменная CRBasic	*При развертывании*	*Через семь дней эксплуатации*
SimulatedFlowSignal	***300*** мВ	***285*** мВ
KnownFlow	***30*** л/с	***30*** л/с
SimulatedFlowSignal	***550*** мВ	***522*** мВ
KnownFlow	***10*** л/с	***10*** л/с
FlowMultiplier	**–0,0799** л/с/мВ	**–0,0841** л/с/мВ
FlowOffset	**53,90** л	**53,92** л

1. Пример CRBasic *FieldCal() — двухточечная калибровка наклона и смещения* *(стр. 219)* передается в CR1000.
2. Чтобы создать для смоделированного датчика расхода смоделированные условия калибровки (в реальных условиях датчик размещается в условиях известного расхода), установите проволочную перемычку между клеммами **VX1** и **SE1**.

3. Смоделированная калибровка развертывания осуществляется следующим образом:
   a. На первом этапе установите значение переменной ***SimulatedFlowSignal = 300.*** Затем задайте значение переменной ***KnownFlow = 30.0.***
   b. Запустите калибровку, установив переменную ***CalMode = 1.***
   c. Когда ***CalMode*** увеличится до **3**, на втором этапе установите переменную ***SimulatedFlowSignal = 550***. Задайте значение переменной ***KnownFlow = 10.***
   d. Продолжайте выполнение калибровки развертывания посредством определения переменной ***CalMode = 4.***
4. Когда переменная ***CalMode*** увеличится до **6**, калибровка развертывания завершится. Откалиброванный множитель равен **–0,08**; откалиброванное смещение — **53,9.**
5. Для продолжения работы с этим примером допускается дрейф откалиброванного множителя и смещения датчика. Для корректировки дрейфа необходимо смоделировать семидневную сервисную калибровку следующим образом:
   a. На первом этапе установите значение переменной ***SimulatedFlowSignal = 285.*** Задайте значение переменной ***KnownFlow = 30.0.***
   b. Запустите семидневную сервисную калибровку, установив переменную ***CalMode = 1***.
   c. Когда ***CalMode*** увеличится до **3**, на втором этапе установите переменную ***SimulatedFlowSignal*** = 522. Задайте значение переменной ***KnownFlow = 10***.
   d. Продолжайте выполнение калибровки посредством определения переменной **CalMode = 4**.
6. Когда переменная **CalMode** увеличится до 6, калибровка завершится. Скорректированный множитель равен **–0,08**; смещение составляет **53,9**.

CRBasic: пример 42. FieldCal() — двухточечная калибровка наклона и смещения

```
Этот пример программы демонстрирует использование команды FieldCal() при расчете и применении
калибровки множителя и смещения. В процессе калибровки множителя и смещения происходит
сравнение амплитуды сигналов датчика с известными стандартами. Расчетный множитель и смещение
приводят отчетную амплитуду датчика к значению, соответствующему линейному соотношению,
при котором пересекаются известные точки, последовательно указанные в параметре FieldCal()
KnownVar.
Последующие измерения масштабируются с учетом новых значений множителя и смещения.

В этой программе показана калибровка множителя и смещения в соответствии со следующей
процедурой:
-- Моделирование сигнала от датчика расхода.
-- Измерение сигнала «датчика».
-- Вычисление и применение множителя и смещения.

Можно использовать настройку, при которой моделирование будет осуществляться при загрузке
данной программы в CR1000 и при соединении следующих клемм при помощи перемычки, чтобы
смоделировать сигнал датчика относительной влажности следующим образом:
следующих клемм при помощи проволочной перемычки, чтобы смоделировать сигнал датчика расхода
следующим образом:
' Vx1 – SE1

В целях моделирования значение калибровочного стандарта и исходный сигнал «датчика»
устанавливаются автоматически. Запустите подпрограмму для множителя и смещения, установив
переменную CalMode = 1. Значение CalMode будет увеличиваться автоматически. Если CalMode = 3,
установите значения переменных SimulatedFlowSignal = 550 и KnownFlow = 10, а также CalMode = 4.
CalMode снова будет прирастать автоматически. Если CalMode = 6 (достигается автоматически через
```

```
10 измерений), подпрограмма завершается. Обратите внимание на новые значения в переменных
FlowMultiplier и FlowOffset. Теперь введите новое значение смоделированного сигнала датчика,
как показано ниже, и обратите внимание на то, как новые множитель и смещение корректируют
измерение:
SimulatedFlowSignal = 1000

ПРИМЕЧАНИЕ. Данная программа помещает файл .cal на диск CPU: регистратора CR1000. Файл .cal
должен быть удален для сброса демонстрации.
@@@be erased to reset the demonstration.

ОБЪЯВИТЕ ПЕРЕМЕННУЮ СМОДЕЛИРОВАННОГО СИГНАЛА И УСТАНОВИТЕ ИСХОДНУЮ АМПЛИТУДУ
Public SimulatedFlowSignal = 300              Ток возбуждения (мВ), второй параметр равен 550

ОБЪЯВИТЕ СТАНДАРТНУЮ ПЕРЕМЕННУЮ КАЛИБРОВКИ И УСТАНОВИТЕ АМПЛИТУДУ
Public KnownFlow = 30                         Известный расход, второй параметр равен 10

ОБЪЯВИТЕ ПЕРЕМЕННУЮ РЕЗУЛЬТАТА ИЗМЕРЕНИЙ.
Public Flow

ОБЪЯВЛЕНИЕ ИТОГОВЫХ ПЕРЕМЕННЫХ МНОЖИТЕЛЯ И СМЕЩЕНИЯ, УСТАНОВКА НАЧАЛЬНЫХ АМПЛИТУД
Public FlowMultiplier = 1
Public FlowOffset = 0

ОБЪЯВИТЕ ПЕРЕМЕННУЮ ДЛЯ КОНТРОЛЯ FieldCal()
Public CalMode


ОБЪЯВИТЕ ТАБЛИЦУ ДАННЫХ ДЛЯ ВЫГРУЖАЕМЫХ РЕЗУЛЬТАТОВ КАЛИБРОВКИ
DataTable(CalHist,NewFieldCal,200)
  SampleFieldCal
EndTable

BeginProg
  ЗАГРУЗИТЕ КОНСТАНТЫ КАЛИБРОВКИ ИЗ ФАЙЛА CPU:CALHIST. CAL
  Выполняется после процедуры калибровки нуля (если переменная CalMode = 6)
  LoadFieldCal (true)

  Scan(100,mSec,0,0)
     СМОДЕЛИРУЙТЕ СИГНАЛ, А ЗАТЕМ ВЫПОЛНИТЕ ИЗМЕРЕНИЕ
     Калибровка множителя применяется, когда значение переменной CalMode = 6
     ExciteV(Vx1,SimulatedFlowSignal,0)
     VoltSE(Flow,1,mV2500,1,1,0,250,FlowMultiplier,FlowOffset)


     ВЫПОЛНЕНИЕ КАЛИБРОВКИ МНОЖИТЕЛЯ.
     Начните с присвоения переменной значения CalMode = 1. Процедура завершена, когда значение
     переменной CalMode = 6.
     FieldCal(Function, MeasureVar, Reps, MultVar, OffsetVar, Mode, KnownVar, Index, Avg)
     FieldCal(2,Flow,1,FlowMultiplier,FlowOffset,CalMode,KnownFlow,1,30)

     Если была произведена калибровка, сохраните результат в таблицу данных
     CallTable(CalHist)

  NextScan
EndProg
```

7.9.12.5.4. Пример калибровки наклона FieldCal() (вариант 3)

В большинстве команд измерения CRBasic присутствуют параметры ***множитель*** и ***смещение***. Команда **FieldCal()** ***Вариант 3*** корректирует аргумент ***множителя*** таким образом, чтобы на выходе калибруемого датчика устанавливалось значение, соответствующее линейному соотношению двух известных пересекающихся точек, последовательно указанных в параметре **FieldCal()** ***KnownVar***. Последующие измерения масштабируются с учетом того же значения множителя. Команда **FieldCal()** ***Вариант 3*** не затрагивает ***смещение***.

Для некоторых практических случаев измерения не требуется определение смещения. Для частотного анализа, например, может быть необходимо только наличие относительных данных, чтобы охарактеризовать изменения.

Пример. Планируется использование датчика содержания влаги в почве для обнаружения импульсов, создаваемых водой при движении сквозь почву. Пульсация воды в почве может быть обнаружена со смещением, однако важное значение имеет непосредственно чувствительность к пульсации, поэтому необходим точный множитель. В целях корректировки чувствительности датчика анализируются два образца почвы с объемным водосодержанием 10 % и 35 %, что дает в результате две известные точки.

Таблица 32. Отчет о калибровке датчика влагосодержания

Переменная CRBasic	***При развертывании***
SimulatedWaterContentSignal	***175*** мВ
KnownWC	***10*** %
SimulatedWaterContentSignal	***700*** мВ
KnownWC	***35*** %
WCMultiplier	**0,0476** %/мВ

С помощью следующей процедуры настраивается чувствительность смоделированного датчика содержания влаги в почве.

1. Пример CRBasic *FieldCal() — множитель* *(стр. 221)* передается в CR1000.
2. Чтобы смоделировать сигнал датчика содержания влаги в почве, установите проволочную перемычку между клеммами **VX1** и **SE1**.
3. Условия калибровки развертывания моделируются в два этапа, как показано ниже:
   a. Установите для переменной ***SimulatedWaterContentSignal*** значение ***175***. Для переменной ***KnownWC*** задайте значение ***10,0***.
   b. Запустите калибровку, установив переменную ***CalMode = 1.***
   c. Когда ***CalMode*** увеличится до **3**, установите для переменной ***SimulatedWaterContentSignal*** значение ***700***. Для переменной ***KnownWC*** задайте значение ***35***.
   d. Продолжайте выполнение калибровки посредством определения переменной ***CalMode = 4.***
4. Когда переменная ***CalMode*** увеличится до 6, калибровка завершится. Откалиброванный множитель равен ***0,0476.***

CRBasic: пример 43. FieldCal() **— множитель**

```
'Этот пример программы демонстрирует использование команды FieldCal() при расчете и применении
'калибровки только по множителю. В процессе калибровки множителя амплитуда сигнала датчика
'сравнивается с известными стандартами. Расчетный множитель приводит отчетную амплитуду
'датчика к значению, соответствующему линейному соотношению, при котором пересекаются известные
'точки, последовательно указанные в параметре FieldCal() KnownVar. Последующие измерения
'корректируются с учетом множителя.
```

```
В этой программе показана калибровка множителя в соответствии со следующей процедурой:
-- Моделирование сигнала от датчика влагосодержания.
-- Измерение сигнала «датчика».
-- Вычисление и применение сдвига.

Можно использовать настройку, при которой моделирование будет осуществляться при загрузке данной
программы в CR1000 и при соединении следующих клемм при помощи перемычки, чтобы смоделировать
сигнал датчика относительной влажности следующим образом:
следующих клемм при помощи проволочной перемычки, чтобы смоделировать сигнал датчика
влагосодержания следующим образом:
' Vx1 --- SE1

В целях моделирования значение калибровочного стандарта и исходный сигнал «датчика»
устанавливаются автоматически. Запустите подпрограмму для множителя, установив переменную
CalMode = 1. Когда CalMode = 6 (достигается автоматически через 10 измерений), подпрограмма
завершается. Обратите внимание на новое значение переменной WCMultiplier. Теперь введите новое
значение смоделированного сигнала датчика, как показано ниже, и обратите внимание на то, как
новый множитель корректирует измерение:
SimulatedWaterContentSignal = 350

ПРИМЕЧАНИЕ. Данная программа помещает файл .cal на диск CPU: регистратора CR1000. Файл .cal
должен быть удален для сброса демонстрации.
@@@be erased to reset the demonstration.

ОБЪЯВИТЕ ПЕРЕМЕННУЮ СМОДЕЛИРОВАННОГО СИГНАЛА И УСТАНОВИТЕ ИСХОДНУЮ АМПЛИТУДУ
Public SimulatedWaterContentSignal = 175      (мВ), второй параметр равен 700 мВ

ОБЪЯВИТЕ СТАНДАРТНУЮ ПЕРЕМЕННУЮ КАЛИБРОВКИ И УСТАНОВИТЕ АМПЛИТУДУ
Public KnownWC = 10                           % от объема, второй параметр равен 35%

ОБЪЯВИТЕ ПЕРЕМЕННУЮ РЕЗУЛЬТАТА ИЗМЕРЕНИЙ.
Public WC

ОБЪЯВЛЕНИЕ ИТОГОВОЙ ПЕРЕМЕННОЙ МНОЖИТЕЛЯ, УСТАНОВКА НАЧАЛЬНОЙ АМПЛИТУДЫ
Public WCMultiplier = 1

ОБЪЯВИТЕ ПЕРЕМЕННУЮ ДЛЯ КОНТРОЛЯ FieldCal()
Public CalMode

ОБЪЯВИТЕ ТАБЛИЦУ ДАННЫХ ДЛЯ ВЫГРУЖАЕМЫХ РЕЗУЛЬТАТОВ КАЛИБРОВКИ
DataTable(CalHist,NewFieldCal,200)
  SampleFieldCal
EndTable

BeginProg
  ЗАГРУЗИТЕ КОНСТАНТЫ КАЛИБРОВКИ ИЗ ФАЙЛА CPU:CALHIST. CAL
  Выполняется после процедуры калибровки нуля (если переменная CalMode = 6)
  LoadFieldCal (true)

  Scan(100,mSec,0,0)
    СМОДЕЛИРУЙТЕ СИГНАЛ, А ЗАТЕМ ВЫПОЛНИТЕ ИЗМЕРЕНИЕ
    Калибровка множителя применяется, когда значение переменной CalMode = 6
    ExciteV(Vx1,SimulatedWaterContentSignal,0)
    VoltSE(WC,1,mV2500,1,1,0,250,WCMultiplier,0)
```

```
  ВЫПОЛНЕНИЕ КАЛИБРОВКИ МНОЖИТЕЛЯ.
  Начните с присвоения переменной значения CalMode = 1. Процедура завершена, когда значение
  переменной CalMode = 6.
  FieldCal(Function, MeasureVar, Reps, MultVar, OffsetVar, Mode, KnownVar, Index, Avg)
  FieldCal(3,WC,1,WCMu1ti p1ier,0,CalMode,KnownWC,1,30)

  Если была произведена калибровка, сохраните результат в таблицу данных CalHist
  CallTable(CalHist)

 NextScan
EndProg
```

7.9.12.5.5. Пример калибровки нулевой точки FieldCal() (вариант 4) — 8 10 30

Калибровка нулевой точки (команда **FieldCal()** ***Вариант 4***) предназначена для использования при измерениях статичных струнных датчиков. Более подробную информацию можно найти в руководствах, доступных на сайте www.campbellsci.com:

Двухканальное устройство измерения струнных датчиков VSPECT серии AVW200 Руководство оператора регистратора данных CR6 по измерению и управлению

7.9.12.6. Примеры калибровки деформации

Связанные темы:
- *Тензометрические измерения — общие сведения (стр. 68)*
- *Тензометрические измерения — подробная информация (стр. 342)*
- *Примеры FieldCalStrain ()* (стр. 223)

Системы датчиков деформации состоят из одного или нескольких тензометрических датчиков, моста сопротивления, на котором расположен датчик, и измерительного устройства, такого как регистратор данных CR1000. Команда **FieldCalStrain()** упрощает параллельную калибровку систем датчиков деформации и предназначена исключительно для случаев механического напряжения, когда единицей измерения является микродеформация. Команда **FieldCal()** (Примеры *FieldCal()* (стр. 213)), как правило, используется в случаях, не относящихся к микронапряжению.

Параллельная калибровка систем датчиков деформации является стандартной практикой. Однако этот метод предоставляет множество возможностей для некорректного применения и неправильного толкования. Этот раздел не призван служить учебником по теории параллельной калибровки, а только знакомит читателя с методом, применяемым в регистраторе данных CR1000. Campbell Scientific настоятельно рекомендует изучить теорию параллельной калибровки, опираясь на другие источники. Подробное описание тензометрических датчиков и теорию параллельной калибровки, предоставленные компанией Vishay, можно найти по запросам «микроизмерения», «анализ напряжений», «датчики деформации», «список счетчиков» на сайте

http://www.vishaypg.com

Технические специалисты Campbell Scientific также располагают ресурсами, которые могут помочь при работе с датчиками деформации.

7.9.12.6.1. Примеры калибровки деформации

1. При параллельной калибровке не калибруется сам тензометрический датчик.
2. Параллельная калибровка компенсирует длину проводов и нелинейность моста сопротивления. Длинные провода снижают чувствительность из-за перепада напряжения. Команда **FieldCalStrain()** использует известное значение шунтирующего резистора для корректировки усиления (множитель/интервал) в целях компенсации. Корректировка усиления (S) объединяется в рамках команды **FieldCalStrain()** с предоставленным

производителем коэффициентом чувствительности тензодатчика (GF), в результате чего получается скорректированный коэффициент тензочувствительности (GF_{adj}), который затем используется в качестве коэффициента чувствительности тензодатчика в **StrainCalc().** GF сохраняется в файле CAL и продолжает использоваться в последующих калибровках. Нелинейность моста компенсируется посредством выбора шунтирующего резистора со значением, которое наилучшим образом моделирует измерение возле диапазона выполняемых измерений. Производители тензометрических датчиков обычно определяют и поставляют ряд резисторов, поддерживающих параллельную калибровку.

3. При параллельной калибровке проверяются функции CR1000.
4. Нулевая функция **FieldCalStrain()** позволяет устанавливать конкретное значение напряжения в качестве нулевой валентности, если необходимо. Установка на нуль, как правило, выполняется после параллельной калибровки.

Опции нуля и шунта могут комбинироваться в рамках одной программы CRBasic.

Пример CRBasic *FieldCalStrain() — калибровка* *(стр. 225)* предназначен для демонстрации использования функций **FieldCalStrain()**. Если тензометрический датчик, настроенный, как показано на рисунке *Четвертьмостовой тензодатчик с шунтом на базе RC-резистора* *(стр. 225)*, недоступен, сигналы деформации могут быть смоделированы посредством создания простого контура с заменой тензометрического датчика на потенциометр на 1000 Ом. Чтобы сбросить проверки калибровки, воспользуйтесь меню ПО поддержки *File Control (Управление файлами)* *(стр. 515)* для удаления файлов .cal, а затем снова отправьте демонстрационную программу в CR1000.

Пример. Тензометрический датчик на 1000 Ом расположен на мосту сопротивления в позиции R1. Созданный контур представляет собой четвертьмостовой тензометрический датчик с отмеченными альтернативными позициями для шунтирующего резистора (Rc). Согласно техническим характеристикам датчика, коэффициент чувствительности равен 2,0, а с учетом шунта на 249 кОм результат измерения составит около 2000 ед. микродеформации.

Отправьте пример CRBasic *FieldCalStrain() — калибровка* *(стр. 225)* в виде программы в регистратор данных CR1000.

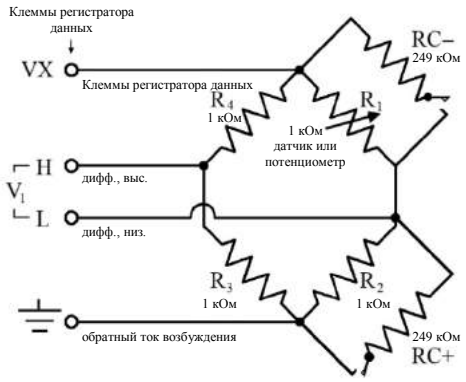
7.9.12.6.2. Примеры калибровки деформации

Пример CRBasic *FieldCalStrain() — калибровка* *(стр. 225)* предназначен для демонстрации использования функций **FieldCalStrain()**. Если тензометрический датчик, настроенный, как показано на рисунке *Четвертьмостовой тензодатчик с шунтом на базе RC-резистора* *(стр. 225)*, недоступен, сигналы деформации могут быть смоделированы посредством создания простого контура с заменой тензометрического датчика на потенциометр на 1000 Ом. Чтобы сбросить проверки калибровки, воспользуйтесь меню ПО поддержки *File Control (Управление файлами)* *(стр. 515)* для удаления файлов .cal, а затем снова отправьте демонстрационную программу в CR1000.

Пример. Тензометрический датчик на 1000 Ом расположен на мосту сопротивления в позиции R1. Созданный контур представляет собой четвертьмостовой тензометрический датчик с отмеченными альтернативными позициями для шунтирующего резистора (Rc). Согласно техническим характеристикам датчика, коэффициент чувствительности равен 2,0, а с учетом шунта на 249 кОм результат измерения составит около 2000 ед. микродеформации.

Отправьте пример CRBasic *FieldCalStrain() — калибровка* *(стр. 225)* в виде программы в регистратор данных CR1000.

Рисунок 58. Четвертьмостовой тензодатчик с шунтом на базе RC-резистора



CRBasic: пример 44. FieldCalStrain() — калибровка

```
В этом примере программы показано использование команды FieldCalStrain() для измерения
показаний четвертьмостовых тензометрических датчиков.

Public Raw_mVperV
Public MicroStrain

Переменные, являющиеся аргументами для функции нуля
Public Zero_Mode
Public Zero_mVperV

Переменные, являющиеся аргументами для функции шунта
Public Shunt_Mode
Public KnownRes
Public GF_Adj
Public GF_Raw

'---------------------------------------- Таблицы ---------------------------------------------
DataTable(CalHist,NewFieldCal,50)
SampleFieldCal
EndTable

'////////////////////////// ПРОГРАММА ////////////////////////////
BeginProg

   Установка коэффициентов чувствительности тензодатчиков
   GF_Raw = 2.1
   GF_Adj = GF_Raw Скорректированные коэффициенты тензочувствительности используются при расчете
   микродеформации

   Если была проведена калибровка, следующая команда загрузит нулевое значение или
   скорректированный коэффициент тензочувствительности из файла калибровки
   LoadFieldCal (True)
```

```
 Scan(100,mSec,100,0)
   Измерение сопротивления моста
   BrFull(Raw_mVperV,1,mV25,1,Vx1,1,2500,True ,True ,0,250,1.0,0)

   Расчет напряжения для 1/4 моста (1 активный элемент)
   StrainCalc(microStrain,1,Raw_mVperV,Zero_mVperV,1,GF_Adj,0)

   Шаги (1) и (3): калибровка нуля
   Выполните балансировку моста и установите параметр Zero_Mode = 1 на цифровом мониторе.
   Повторите операцию после
   калибровки шунта.
   FieldCalStrain(10,Raw_mVperV,1,0,Zero_mVperV,Zero_Mode,0,1,10,0 ,microStrain)

   Шаг (2): калибровка шунта
   После калибровки нуля и балансировки моста (установки на нуль) установите для параметра
   KnownRes значение сопротивления датчика (в состоянии покоя), а затем задайте параметр
   Shunt_Mode = 1. Когда Shunt_Mode увеличится до 3, разместите шунтирующий резистор
   и приравняйте значение параметра KnownRes к сопротивлению шунта, а затем установите параметр
   Shunt_Mode = 4.
   FieldCaIStrain(13,MicroStrain,1,GF_Adj ,0,Shunt_Mode,KnownRes,1,10,GF_Raw,0)

   CallTable CalHist
 NextScan
EndProg
```

7.9.12.6.3. Пример калибровки четвертьмостового шунта FieldCalStrain()

После отправки примера CRBasic *FieldCalStrain() — калибровка* *(стр. 225)* в CR1000 и стабилизации тензометрического датчика воспользуйтесь экранной клавиатурой CR1000KD или программным цифровым монитором для изменения значения переменной **KnownRes** на номинальное сопротивление тензодатчика (**1000 Ом**), как показано на рисунке *Начало калибровки шунта тензодатчика* *(стр. 226)*. Установите для параметра **Shunt_Mode** значение **1**, чтобы запустить двухточечную калибровку шунта. Когда **Shunt_Mode** увеличится до **3**, первый шаг завершится.

Чтобы завершить калибровку, включите резистор на 249 кОм параллельно позиции R1. Установите для переменной **KnownRes** значение **249000**. Как показано на рисунке *Окончание калибровки шунта тензодатчика* *(стр. 227)*, установите для параметра **Shunt_Mode** значение **4**. Если **Shunt_Mode = 6**, параллельная калибровка завершена.

Рисунок 59. Начало калибровки шунта тензодатчика

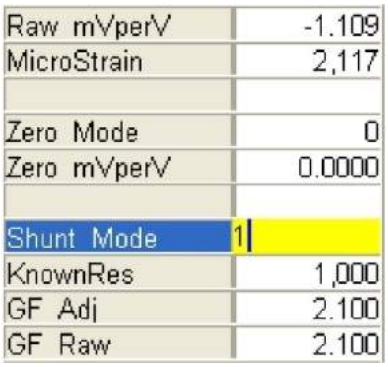


Рисунок 60. Окончание калибровки шунта тензодатчика

Raw mVperV	-1.109
MicroStrain	-2,215
Zero Mode	0
Zero mVperV	0.0000
Shunt Mode	6
KnownRes	249,000
GF Adj	-2.008
GF Raw	2.000

7.9.12.6.4. FieldCalStrain() — калибровка четвертьмостового нуля

Опираясь на *Пример калибровки четвертьмостового шунта FieldCalStrain()* *(стр. 226)*, не меняйте положение резистора на 249 кОм, чтобы смоделировать напряжение. С помощью экранной клавиатуры CR1000KD или программного цифрового монитора измените значение переменной **Zero_Mode** на **1**, чтобы запустить калибровку нуля, как показано на рисунке *Начало нулевой процедуры* *(стр. 227)***.** Когда **Zero_Mode** увеличится до **6**, калибровка нуля завершается, как показано на рисунке *Окончание нулевой процедуры* *(стр. 227)***.**

Рисунок 61. Начало нулевой процедуры

Raw mVperV	-1.110
MicroStrain	-2,214
Zero Mode	1
Zero mVperV	0.0000
Shunt Mode	6
KnownRes	249,000
GF Adj	-2.010
GF Raw	2.000

Рисунок 62. Окончание нулевой процедуры

Raw mVperV	-1.110
MicroStrain	0
Zero Mode	6
Zero mVperV	-1.1096
Shunt Mode	6
KnownRes	249,000
GF Adj	-2.010
GF Raw	2.000

7.9.13. Измерения: возбуждение, задержка, считывание показаний

В этом примере демонстрируется способ выполнения измерений напряжения, для которых требуется подача тока возбуждения с регулируемой длиной пути, прежде чем будет произведено измерение. Типичной проблемой в рамках данного метода является компенсация задержки, вызванной чрезмерной длиной кабеля, подключенного к датчику.

CRBasic: пример 45. Измерение с учетом возбуждения и задержки

```
'Этот пример программы иллюстрирует метод выполнения операции возбуждения/задержки/измерения.
'В данном примере для стабилизации системы требуется возбуждение в течение 1 секунды, прежде чем
'можно будет считать показания датчиков. По отдельности выполняются несимметричное и
'дифференциальное измерения. Чтобы увидеть программу в действии, подключите следующие контактные
'пары, смоделировав таким образом подключение датчиков:

'   Vx1 -------- SE1
'   Vx2 -------- DIFF 2 H
'   DIFF 2 L -------- Символ заземления
'
'После подключения значения переменных VoltageSE и VoltageDiff будут равны 2500 мВ.

'Объявление переменных.
Public VoltageSE As Float
Public VoltageDIFF As Float

'Объявление таблицы данных
DataTable (Voltage,True,-1)
  Sample (1,VoltageSE,Float)
  Sample (1,VoltageDIFF,Float)
EndTable

BeginProg

  Scan(5,sec,0,0)

    'Возбуждение – задержка на 1 с – несимметричное измерение:
    ExciteV (Vx1,2500,0)'<<<<Примечание. задержка = 0
    Delay (0,1000,mSec)
    VoltSe (VoltageSE,1,mV5000,1,1,0,250,1.0,0)

    'Возбуждение – задержка на 1 с – дифференциальное измерение:
    ExciteV (Vx2,2500,0) '<<<<Note: Delay = 0
    Delay (0,1000,mSec)
    VoltDiff (VoltageDIFF,1,mV5000,2,True,0,250,1.0,0)

    'Запись данных в память окончательных данных
    CallTable Voltage

  NextScan

EndProg
```

7.9.14. Измерения: повышенные аналоговые частоты

Для некоторых приложений сбора данных требуется, чтобы аналоговые измерения в CR1000 выполнялись чаще, чем раз в секунду (частота > 1 Гц *(стр. 517)*). CR1000 может выполнять непрерывные измерения с частотой до 100 Гц, при необходимости временно увеличивая частоту до 2000 Гц (так называемые *«вспышки»* *(стр. 509)*. Ниже рассматриваются методы программирования быстрых измерений с использованием команды несимметричного измерения аналогового напряжения **VoltSE()**. Приведенные методы могут также использоваться в сочетании со следующими командами:

```
VoltSE()
VoltDiff()
TCDiff()
TCSE()
BrFull()
BrFull6W()
BrHalf()
BrHalf3W()
BrHalf4W()
```

В таблице *Частота аналоговых измерений напряжения — краткое описание* *(стр. 230)* сгруппированы методы программирования, применяемые для трех разных классов быстрых измерений: с максимальной частотой 100 Гц, 600 Гц и 2000 Гц. Для измерений с частотой 100 Гц возможен 100-процентный *коэффициент использования* *(стр. 514)*. Это означает, что измерения, как правило, не приостанавливаются, чтобы выровняться с процессом обработки. Приостановленные измерения приравниваются к упущенным возможностям измерения и могут быть нежелательными. Для измерений с частотой 600 Гц и 2000 Гц (превышающих 100 Гц) характерен коэффициент использования ниже 100 %.

Таблица 33. Частота аналоговых измерений напряжения — краткое описание			
Максимальная частота	100 Гц	600 Гц	2000 Гц
Число параллельных вводов	Множество вводов	Несколько вводов	Один ввод
Максимальный коэффициент использования	100 %	< 100 %	< 100 %
Макс. число измерений на «вспышку»	Отсутствует	Переменная	65535
Описание	Практически одновременные измерения во множестве каналов До 8 последовательных дифференциальных или 16 несимметричных каналов. Буферы постоянно используются повторно, поэтому пропусков сканирования нет.	Практически одновременные измерения в нескольких каналах Буферы могут быть задействованы и освобождены только после пропуска сканирования. Выделение большего числа буферов обычно означает, что между пропусками сканирования пройдет больше времени.	Единственная «вспышка» команды измерения CRBasic в одном канале. Измерения выполняются во множестве каналов с помощью множества команд, однако «вспышка» в одном канале завершается до возникновения «вспышки» в другом канале.
Последовательность аналоговых клемм	Дифференциальные: 1, 2, 3, 4, 5, 6, 7, 8, затем повтор. Несимметричные: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, затем повтор.	Дифференциальные и несимметричные: 1, 2, 1, 2 и т. д.	1, 1, 1... до завершения, затем 2, 2, 2... до завершения, затем 3, 3, 3 и т. д.
Ток возбуждения для измерений моста	Указано в инструкции.	Указано в инструкции.	Указано в инструкции. ***Количество измерений на интервал возбуждения*** должно равняться значению параметра ***Повторы***
Программирование CRBasic — основные особенности	Рекомендуется использовать команды **Scan() / NextScan** с интервалом сканирования 10 мс. На программном уровне применяется до 10 буферов. См. пример CRBasic *Измерение VoltSE() при 100 Гц*	Используйте команды **Scan() / NextScan** с интервалом сканирования 20 мс или больше. На программном уровне применяется до 100 буферов. Также можно использовать команды **SubScan() / NextSubScan** с дополнительным сканированием на 1600 мкс и 12 считываниями счетчика. См. пример CRBasic *Измерение VoltSE() при 200 Гц*	Используйте команды **Scan() / NextScan** с интервалом сканирования 1 с. Перед аргументом аналогового ввода ***Канал*** ставится прочерк (-). См. пример CRBasic *Измерение VoltSE() при 2000 Гц*

7.9.14.1. Измерения от 1 до 100 Гц

Допустим, используется минимальная программа CRBasic, в рамках которой частота измерения от 1 до 100 Гц определяется параметрами ***Интервал*** и ***Единицы измерения*** в паре команд **Scan() / NextScan**. С помощью следующей программы команда **VoltSE()** выполняется с частотой 1 Гц и 100-процентным коэффициентом использования.

CRBasic: пример 46. Измерение VoltSE() при 1 Гц

```
PipelineMode <<<<Поточный режим обеспечивает точную координацию измерений по времени.

Public FastSE

DataTable(FastSETable,1,-1)
   Sample(1,FastSE(),FP2)
EndTable
```

```
BeginProg
   Scan(1,Sec,0,0) <<<<Частота измерений определяется параметрами «Интервал» и «Единицы
   измерения»
      VoltSe(FastSE(),1,mV2_5,1,Fa1se,100, 250,1.0,0)
      CallTable FastSETable
   NextScan
EndProg
```

При изменении аргументов ***«Интервал», «Единицы измерения» и «Буферы»*** команда **VoltSE()** может выполняться с частотой 100 Гц и 100-процентным коэффициентом использования. С помощью следующей программы считываются показания 16 аналоговых входных клемм при частоте 100 Гц.

CRBasic: пример 47. Измерение VoltSE() при 100 Гц

```
PipelineMode<<<<Поточный режим обеспечивает точную координацию измерений по времени.

Public FastSE(16)

DataTable(FastSETable,1,-1)
Sample(16,FastSE(),FP2)
EndTable

BeginProg
   Scan(10,mSec,10,0) <<<<Частота измерений определяется параметрами «Интервал», «Единицы
   измерения» и «Буферы»
      VoltSe(FastSE(),1,mV2_5,1,False,100, 250,1.0,0)
      CallTable FastSETable
   NextScan
EndProg
```

7.9.14.2. Частота измерений: от 101 до 600 Гц

Для измерения с частотой от 100 до 600 Гц добавляется пара команд **SubScan() / NextSubScan**. Измерения с частотой выше 100 Гц не обладают 100-процентным коэффициентом использования и выполняются с помощью «вспышек» измерения. Длительность каждой «вспышки» составляет определенную часть интервала сканирования. В течение оставшегося интервала сканирования процессор CR1000 выполняет незавершенные задачи и обрабатывает данные, сохраненные в буферах. Например, можно запрограммировать CR1000 на применение команды **VoltSE()** для измерения восьми последовательных входов с частотой 200 Гц и 95-процентным коэффициентом использования, как показано в следующем примере:

CRBasic: пример 48. Измерение VoltSE() при 200 Гц

```
PipeLineMode<<<<Поточный режим обеспечивает точную координацию измерений по времени.

Public BurstSE(8)

DataTable(BurstSETable,1,-1)
   Sample(8, BurstSE() , FP2)
EndTable

BeginProg
   Scan(1,Sec,10,0) <<<<Добавление буферов
    SubScan(5,mSec,190) <<<<Параметры «Интервал», «Единицы измерения» и «Отсчеты» определяют
    скорость и число измерений
      VoltSe(BurstSE(),8,mV2_5,1,False,100,250,1.0,0)
      CallTable BurstSETable
    NextSubScan
   NextScan
EndProg
```

Этот базовый код в различных вариациях может применяться для других значений частоты «вспышек» и коэффициентов использования.

При использовании пары команд **SubScan() / NextSubScan** возникают дополнительные сложности. В разделе *SubScan() /NextSubScan — подробная информация* *(стр. 231)* описываются некоторые из них. Из соображений предосторожности рекомендуется тщательно проверять конкретную

конфигурацию перед развертыванием. В общих чертах, для более высоких значений частоты требуется измерение по меньшему числу вводов. При тестировании программы мониторинг реестров ***SkippedScan*** *(стр. 628)*, ***BuffDepth*** *(стр. 612)* и ***MaxBuffDepth*** *(стр. 620)* в таблице CR1000 **Status** (Состояние) может предоставить оперативную информацию по использованию ресурсов буфера. Следует учитывать, что при превышении числа буферов **Scan() / NextScan** произойдет пропуск сканирования и, следовательно, событие потери данных.

7.9.14.2.1. Измерения от 101 до 600 Гц 2

- Количество ***Отсчетов*** (циклов) при дополнительном сканировании ограничено до 65535
- Дополнительные сканирования выполняются только в рамках структуры **Scan() / NextScan** с достаточно большим заданным интервалом **Scan()**, чтобы дополнительное сканирование могло выполняться до достижения предельного числа отсчетов.
- Если умножить интервал дополнительного сканирования (i) на количество прогонов дополнительного сканирования (n), получится некая доля времени измерения (MT_1), которое приведено в реестре ***MeasureTime*** таблицы *Описание полей таблицы «Status»* *(стр. 603)*. На команду **EndScan** приходится дополнительно 100 мкс от времени измерения, поэтому интервал основного сканирования должен составлять ≥ 100 мкс, плюс время измерения вне конструкции **SubScan() / EndSubScan**, плюс время, затрачиваемое на дополнительное сканирование.
- Поскольку устройство определения последовательности задач контролирует дополнительное сканирование, выполнение операции не завершается, пока не будут завершены все дополнительные прогоны сканирования и последующие задачи. Как следствие, обработка не запускается, пока не будет завершено дополнительное сканирование, а устройство определения последовательности задач не определит задержку начала следующего прогона основного сканирования. Таким образом, в одном буфере **Scan() / NextScan** хранятся все необработанные измерения, как относящиеся, так и не относящиеся к дополнительному сканированию; т.е. все измерения, выполненные в процессе единого основного сканирования. Так, например, при одном выполнении следующей последовательности кода в одном буфере сохраняется 30 000 измерений:

```
Scan(40,Sec,3,0) Сканирование (интервал, единицы измерения,
буфер, счетчик)
   SubScan(2,mSec,10000)
      VoltSe(Measurement(),3,mV5000,1,False,150,250,1.0,0)
      CallTable All4
   NextSubScan
NextScan
```

Примечание. Время измерения в предыдущем коде составляет 300 мкс +19 мс, так что при интервале **Scan()** менее 20 мс возникнет ошибка компиляции.

- Преимуществом дополнительного сканирования является возможность выполнения с частотой выше 100 Гц. Однако измерения, которые могут выполняться с частотой ровно 100 Гц, также имеют преимущество: поскольку все прогоны дополнительного сканирования должны быть выполнены, прежде чем устройство определения последовательности задач сможет установить задержку основного сканирования, обработка откладывается на этот срок (20 мс в приведенном выше примере). Таким образом, для приблизительного пространства буфера в режиме дополнительного сканирования необходим больший объем памяти, чтобы операции выполнялись с той же скоростью, что и в режиме без дополнительного сканирования; кроме того, увеличивается задержка перед «вспышкой» ввиду необходимости завершения обработки. Поток (буфер необработанных данных) должен заполниться, прежде чем можно будет начать обработку.

- Еще один аспект дополнительного сканирования заключается в том, что это удобный (и единственный) способ зацикливания набора измерений. В рамках команд **SubScan() / NextSubScan** задается синхронизируемый цикл с таким количеством повторов набора измерений, который поддерживается устройством определения последовательности задач.

7.9.14.3. Частота измерений: от 601 до 2000 Гц

Для выполнения измерений с частотой выше 600 Гц команда **VoltSE()** переводится в пакетный режим путем размещения прочерка (-) перед аргументом параметра **SEChan**, а также альтернативных аргументов в других параметрах. Альтернативные аргументы описаны в таблице *Параметры для аналогового пакетного режима* *(стр. 234)*. В пакетном режиме с помощью команды **VoltSE()** задействуется отдельный канал, для которого выполняются измерения с частотой до 2000 Гц, как показано в примере CRBasic «Измерение VoltSE() при 2000 Гц». Коэффициент использования в данном примере программы составляет 86 %. Это означает, что измерения выполняются только по первым 86 % результатов сканирования. Обратите внимание, что в пакетном режиме все измерения для заданной «вспышки» попадают в отдельный массив переменных, который сохраняется в виде единственной (и очень длинной!) записи в таблице данных. Точный интервал выборки рассчитывается как

```
Tsample = 1.085069 * INT((SettleUSEC / 1.085069) + 0.5
```

где ***SettleUSEC*** — это интервал выборки (мкс), указанный в параметре ***SettlingTime*** команды аналогового ввода.

CRBasic: пример 49. Измерение VoltSE() при 2000 Гц

```
PipelineMode<<<<Поточный режим обеспечивает точную координацию измерений по времени.

Public BurstSE(1735)

DataTable(BurstSETable,1,-1)
   Sample(1735,BurstSE(),FP2)
EndTable

BeginProg
   Scan(1,Sec,10,0)
      'Скорость измерения и счетчик устанавливаются в рамках VoltSE()
      VoltSe(BurstSE(),1735,mV2_5,-1,False,500,0,1.0,0)
      CallTable BurstSETable
   NextScan
EndProg
```

Возможны различные вариации пакетной программы. Измерения могут затрагивать множество входов, однако до начала новой «вспышки» должна быть завершена обработка предыдущей. Из соображений предосторожности рекомендуется тщательно проверять конкретную конфигурацию перед развертыванием.

Таблица 34. Параметры для аналогового пакетного режима (от 601 до 2000 Гц)

Входные параметры аналогового напряжения CRBasic	***Описание при активном пакетном режиме***
Назначение	Массив переменных, измерения которого позволяют хранить все показания с одного входа. Например, с помощью команды `Dim FastTemp(500)` устанавливается размер массива ***FastTemp()***, при котором допускается хранение 500 показаний (одна секунда данных при 500 Гц, полсекунды данных при 1000 Гц и т. д.) Размер может выражаться любым целым числом от ***1*** до ***65535***.
Повторы	Количество измерений, выполняемых для одного входа. Это число обычно равняется числу элементов согласно размеру массива ***Назначение.*** Допустимые аргументы находятся в диапазоне от ***1*** до ***65535***.
Диапазон напряжения	Диапазон аналогового напряжения на входе, используемый при измерениях. Без изменений по сравнению со стандартным режимом измерения. Может использоваться любой допустимый диапазон напряжения. Тем не менее измерения в диапазонах с добавлением символа «C» в конце выполняются медленнее остальных.
Несимметрич-ный канал	Номер клеммы несимметричного аналогового ввода с прочерком впереди (-). Допустимые аргументы находятся в диапазоне от ***–1*** до –16.
Дифференци-альный канал	Номер клеммы дифференциального аналогового ввода с прочерком впереди (-). Допустимые аргументы находятся в диапазоне от ***–1*** до -8.
Смещение измерения	Без изменений по сравнению со стандартным режимом измерения. Значение ***False*** («ложь») ускоряет измерения.
Количество измерений на интервал возбуждения	Значение должно равняться параметру ***Повторы***
Обратный ток возбуждения	Без изменений по сравнению со стандартным режимом измерения. Для увеличения частоты установите значение ***False*** («ложь»).
Обратный дифференциал (RevDiff)	Без изменений по сравнению со стандартным режимом измерения. Для увеличения частоты установите значение ***False*** («ложь»).
Время регулирования	Интервал выборки в мкс. Этот аргумент определяет частоту измерения. 500 мкс (интервал) = 2000 Гц (частота) 750 мкс (интервал) = 1333,33 Гц (частота)
Целое (Integ)	Игнорируется, если установлено целое число. Значения ***_50Hz*** и ***_60Hz*** допустимы для подавления переменного тока, однако очень редко используются в пакетном режиме.
Multiplier	Без изменений по сравнению со стандартным режимом измерения. Введите соответствующий множитель. Это отклонение линейного уравнения, приравнивающего выходное напряжение к измеренному событию. Допустимо любое число больше или меньше ***0.***
Смещение	Без изменений по сравнению со стандартным режимом измерения. Введите соответствующее смещение. Это точка пересечения с осью Y для линейного уравнения, приравнивающего выходное напряжение к измеренному событию.

7.9.15. Измерения: ПТС

ПТС (платиновые термометры сопротивления) — это высокоточные устройства сопротивления, используемые при измерении температуры.

7.9.15.1. Измерение PT100 (ПТС на 100 Ом)

PT100 (ПТС на 100 Ом) общедоступны. CR1000 может считывать показания PT100 в нескольких конфигурациях, каждая из которых имеет свои преимущества.

7.9.15.1.1. Самонагрев и разрешающая способность

Измерения платиновых термометров сопротивления (ПТС) производятся по принципу дихотомии. Для максимизации разрешающей способности измерительной системы напряжение возбуждения также должно быть максимизировано. Наоборот, при минимизации напряжения возбуждения минимизируется степень самонагрева ПТС.

Если перепад напряжения на ПТС ≤ 25 мВ, самонагрев не должен превышать 0,001 °C в спокойном воздухе. Для максимизации разрешающей способности измерительной системы необходимо оптимизировать напряжение возбуждения (V_X) так, чтобы перепад напряжения охватывал, но не превосходил входной диапазон напряжения.

7.9.15.1.2. Стандарты вычислений для ПТС

Для поддержки измерений с помощью ПТС доступны две команды CRBasic.

PRT() — устаревшая команда. Расчет температуры производится на основе сопротивления резистивных датчиков температуры (РДТ) в соответствии со стандартом DIN 43760. Практически во всех случаях этот подход заменяется на **PRTCalc().**

PRTCalc() — расчет температуры производится на основе сопротивления РДТ в соответствии с одним из нескольких поддерживаемых стандартов. **PRTCalc()** превалирует над **PRT()** практически во всех случаях.

Для ПТС промышленного типа отношение между температурой и сопротивлением описывается уравнением Каллендера – Ван Дьюзена (CVD). Коэффициенты для различных типов датчиков приводятся в публикуемых стандартах или задаются производителем в случае нестандартных типов. Измеряемые температуры сравниваются со шкалой ITS-90, стандартом калибровки измерительных приборов для температуры.

PRTCalc() соответствует принципам и уравнениям, приведенным в стандарте US ASTM E1137-04 по преобразованию сопротивления в температуру. Для температурного диапазона от 0 до 650 °C прямое решение уравнения CVD имеет погрешность < ±0,0005 °C (вследствие погрешности округления в системе CR1000). Для диапазона от –200 до 0 °C используется полином четвертого порядка, позволяющий преобразовать сопротивление в температуру с погрешностью < ±0,003 °C.

Эта погрешность появляется только в результате приближенного вычисления соотношения между температурой и сопротивлением согласно применимым стандартам. Уравнения CVD и таблицы на их основе дают лишь приблизительную оценку истинной линейной зависимости РДТ, но считаются подходящими для промышленного использования. Погрешность приближения может составлять несколько сотых градуса Цельсия в различных точках температурного диапазона и различаться в зависимости от конкретного датчика. Кроме того, отдельные датчики могут иметь погрешность относительно стандартного значения, составляющую до ±0,3 °C при 0 °C, причем по мере удаления от 0 °C погрешность увеличивается (в зависимости от шкалы датчика). Максимальная точность обычно достигается за счет калибровки отдельных датчиков в диапазоне использования и применения корректировок значений R_S/R_O при вводе в команду **PRTCalc()** (путем использования калиброванного значения R_O), а также параметров смещения и множителя.

Специфичные вводные значения параметров *PRTCalc()* см. в **Справке по редактору CRBasic**. Следующая информация представлена в качестве дополнительных сведений к доступным в *Справке по редактору CRBasic.*

Общий вид уравнения Каллендера – Ван Дьюзена (CVD) представлен следующими уравнениями.

Если $R/R_0 < 1$ ($K = R/R_0 – 1$):

```
T = g * K^4 + h * K^3 + I * K^2 + j * K
```

Если $R/R_0 >= 1$:

```
T = (SQRT(d * (R/R0) + e) -a) / f
```

В зависимости от кода, введенного для параметра ***Тип***, определяющего тип платинового термометра сопротивления, коэффициентам присваиваются значения в соответствии со следующими таблицами.

Примечание. Коэффициенты округляются до седьмого значащего разряда в соответствии с заданной точностью математической функции CR1000.

Альфа определяется следующим образом:

$\alpha = (R_{100} – R_0) / (100 \bullet R_0)$

$\alpha = (R_{100} / R_0 – 1) / 100$

где R_{100} и R_0 — это сопротивление ПТС при 100 °C и 0 °C соответственно.

Таблица 35. PRTCalc(): датчик с кодом типа 1

IEC 60751:2008 (IEC 751), альфа = 0,00385. На данный момент принято на международном уровне и внесено в стандарты ASTM E1137-04, JIS 1604:1997, EN 60751 и прочие. Этот код типа также используется для измерительных элементов, соответствующих устаревшим стандартам DIN43760, BS1904 и т. д. (Ссылка: IEC 60751. ASTM E1137)

Постоянная	***Коэффициент***
a	3,9083000E-03
d	–2,3100000E-06
e	1,7584810E-05
f	–1,1550000E-06
g	1,7909000E+00
h	–2,9236300E+00
i	9,1455000E+00
j	2,5581900E+02

Таблица 36. PRTCalc(): датчик с кодом типа 2	
Промышленный стандарт США, альфа = 0,00392 (Ссылка: Logan Enterprises)	
Постоянная	***Коэффициент***
a	3,9786300E-03
d	–2,3452400E-06
e	1,8174740E-05
f	–1,1726200E-06
g	1,7043690E+00
h	–2,7795010E+00
i	8,8078440E+00
j	2,5129740E+02

Таблица 37. PRTCalc(): датчик с кодом типа 3	
Промышленный стандарт США, альфа = 0,00391 (Ссылка: OMIL R84 (2003))	
Постоянная	***Коэффициент***
a	3,9690000E-03
d	–2,3364000E-06
e	1,8089360E-05
f	–1,1682000E-06
g	1,7010560E+00
h	–2,6953500E+00
i	8,8564290E+00
j	2,5190880E+02

Таблица 38. PRTCalc(): датчик с кодом типа 4	
Старый японский стандарт, альфа = 0,003916 (Ссылка: JIS C 1604:1981, National Instruments)	
Постоянная	***Коэффициент***
a	3,9739000E-03
d	–2,3480000E-06
e	1,8139880E-05
f	–1,1740000E-06
g	1,7297410E+00
h	–2,8905090E+00
i	8,8326690E+00
j	2,5159480E+02

Таблица 39. PRTCalc(): датчик с кодом типа 5	
Промышленные датчики Honeywell, альфа = 0,00375 (Ссылка: Honeywell)	
Постоянная	***Коэффициент***
a	3,8100000E-03
d	–2,4080000E-06
e	1,6924100E-05
f	–1,2040000E-06
g	2,1790930E+00
h	–5,4315860E+00
i	9,9196550E+00
j	2,6238290E+02

Таблица 40. PRTCalc(): датчик с кодом типа 6	
Стандарт ITS-90 SPRT, альфа = 0,003926 (Ссылка: Minco / Instrunet)	
Постоянная	***Коэффициент***
a	3,9848000E-03
d	–2,3480000E-06
e	1,8226630E-05
f	–1,1740000E-06
g	1,6319630E+00
h	–2,4709290E+00
i	8,8283240E+00
j	2,5091300E+02

7.9.15.2. PT100 в четырехжильном полумостовом подключении

Пример демонстрирует:

- Принцип измерения ПТС в четырехжильной полумостовой конфигурации
- Принцип компенсации крупного шага

Преимущества:

- Высокая точность при крупном шаге

Пример технических характеристик ПТС:

- Альфа = 0,00385 (ПТС: тип 1)

Четырехжильный полумост, измеряемый с помощью **BrHalf4W(),** является наилучшим вариантом конфигурации, обеспечивающим точность в тех случаях, когда ПТС отделяется от шунтирующих резисторов длинным проводом с сопротивлением более нескольких тысячных ома. В данном примере диапазон измерений составляет от –10 до 40 °C. Длина кабеля от CR1000 и шунтирующих резисторов до ПТС равна 500 футов.

На рисунке *PT100 в четырехжильном полумостовом подключении* *(стр. 240)* представлен контур, используемый при измерении ПТС на 100 Ω. Резистор на 10 kΩ позволяет использовать высокое напряжение возбуждения и низкий входной диапазон. Это гарантирует отсутствие влияния помех при возбуждении на помехи сигнала. Поскольку постоянный резистор (R_f) и ПТС (R_S) обладают примерно одинаковыми показателями сопротивления, дифференциальное измерение перепада напряжения в ПТС может осуществляться в пределах того же диапазона, что и дифференциальное измерение перепада напряжения для R_f. Использование одинакового диапазона устраняет ошибки перехода диапазонов, которые могут быть следствием допуска 0,01 % для резисторов перехода диапазона в рамках CR1000.

7.9.15.2.1. Расчет напряжения возбуждения

Перепад напряжения для ПТС равен значению V_X, умноженному на отношение R_S к общему сопротивлению и достигает максимума при максимальном значении R_S (R_S = 115,54 Ω при 40 °C). Чтобы найти максимальное значение напряжения возбуждения, которое может быть использовано в диапазоне входных значений ±25 мВ, предположим, что V_2 равняется 25 мВ, и используем Закон Ома, получая решение для результирующего тока, I.

$I = 25 мВ/R_S = 25 мВ/115.\ 54 Ом = 0,216 мА$

Следующее решение для V_X:

$V_X = I*(R_1 + R_S + R_f) = 2,21 В$

Если фактические значения сопротивления совпадают с номинальными, CR1000 не выходит за рамки диапазона при V_X = 2,2 В. Тем не менее, учитывая фактический допуск резисторов, следует задать для V_X значение 2,1 В (например, если показатели резистора на 10 kΩ снижены на 5 %, т. е. $R_S/(R_1+R_S+R_f)$=115,54 / 9715,54, а значение V_X должно составлять 2,102 В, чтобы V_S не превышало 25 мВ).

7.9.15.2.2. Расчет значения множителя BrHalf4W()

Результат **BrHalf4W()** равен R_S/R_f.

$X = R_S/R_f$

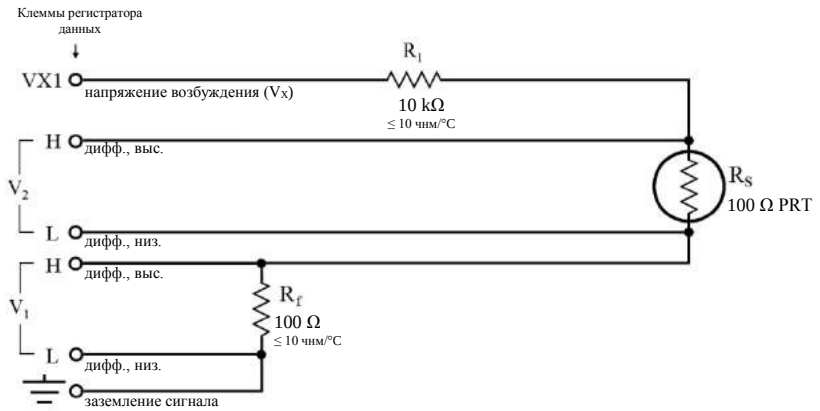
PRTCalc() рассчитывает температуру (°C) для ПТС в соответствии со стандартом DIN 43760 на основании отношения сопротивления ПТС к его же сопротивлению при 0 °C (R_S/R_0). Таким образом, множитель R_f/R_0 используется в **BrHalf4W()** для нахождения желаемого среднего значения, R_S/R_0 (=R_S/R_f • R_f/R_0). Если значения R_S и R_0 составляют ровно 100 Ω, множитель равен 1. Однако ни одно из значений сопротивления не является точным. Правильное значение множителя находится посредством подключения ПТС к CR1000 и ввода **BrHalf4W()** с множителем 1. Затем ПТС помещается в водяную баню (0 °C), и считывается результат измерения по мостовой схеме. Измеренное значение R_S/R_f равно R_0/R_f, поскольку R_S=R_0 при 0 °C. Корректное значение множителя, R_f/R_0, обратно пропорционально этому значению. Начальное показание, принятое в данном примере, составляло 0,9890. Корректный множитель: R_f/R_0 = 1/0,9890 = 1,0111.

7.9.15.2.3. Выбор Rf

Постоянный резистор на 100 Ω должен быть термически стабильным. Его погрешность не так важна, поскольку точное значение сопротивления корректируется калиброванным множителем, как и значение ПТС. Температурный коэффициент постоянного резистора (10 чнм/°C) ограничивает предел погрешности вследствие изменения сопротивления и температуры до 0,15 °C в температурном диапазоне от –10 до 40 °C. Поскольку измерение является логометрическим (R_S/R_f), свойства резистора на 10 kΩ не влияют на результат.

Терминальный модуль ввода (TIM) может использоваться для завершения контура, представленного на рисунке *PT100 в четырехжильном полумостовом подключении* *(стр. 240)*. В приложении *Преобразователи сигнала* *(стр. 647)* приведена информация о доступных модулях TIM.

Рисунок 63. PT100 в четырехжильном полумостовом подключении



CRBasic: пример 50. PT100 в четырехжильном полумостовом подключении

```
'Этот пример программы демонстрирует измерение ПТС на 100 Ом с помощью четырехжильного
'полумостового подключения. Схему электропроводки см. на РИСУНКЕ PT100 в четырехжильном
'полумостовом подключении (стр. 240)

Public Rs_Ro
Public Deg_C

BeginProg
   Scan(1,Sec,0,0)

      'BrHalf4W(Dest, Reps,Range1,Range2,Di ffChan1,ExChan,MPS,Ex_mV,RevEx,RevDiff,
      'Settling, Integration,Mult,Offset)
      BrHalf4W(Rs_Ro,1,mV25,mV25,1,Vx1,1,2200,True,True,0,250,1.0111,0)

      'PRTCalc(Destination,Reps,Source,PRTType,Mult,Offset)
      PRTCalc(Deg_C,1,Rs_Ro,1,1.0,0) 'PRTType sets alpha

   NextScan
EndProg
```

7.9.15.3. PT100 в трехжильном полумостовом подключении

Пример демонстрирует:

- Принцип измерения ПТС в трехжильной полумостовой конфигурации.

Преимущества:

- Используется в два раза меньше терминалов, настроенных для аналогового ввода, чем в четырехжильном полумостовом подключении.

Недостатки:

- Точность может быть ниже, чем при четырехжильном полумостовом подключении.

Пример технических характеристик ПТС:

- Альфа = 0,00385 (ПТС: тип 1)

Требования к измерению температуры в данном примере аналогичны случаю *PT100 в четырехжильном полумостовом подключении* *(стр. 238)***.** В данном случае для измерения сопротивления ПТС используются трехжильная полумостовая конфигурация и команда CRBasic **BRHalf3W()**. Схема контура ПТС представлена на рисунке *PT100 в трехжильном полумостовом подключении* *(стр. 242)***.**

Как и в разделе *PT100 в четырехжильном полумостовом подключении* *(стр. 238)***,** рассчитывается максимально возможное напряжение возбуждения, при этом позволяющее производить измерения в диапазон входных значений ±25 мВ. Резистор на 10 kΩ имеет допуск ±1 %; таким образом, наименьшее ожидаемое сопротивление составляет 9,9 kΩ. Необходимо найти решение для V_X (максимальное напряжение возбуждения), чтобы перепад напряжения для ПТС не превышал 25 мВ:

```
0.025 V > (Vx * 115.54)/(9900+115.54)
Vx < 2.16 V
```

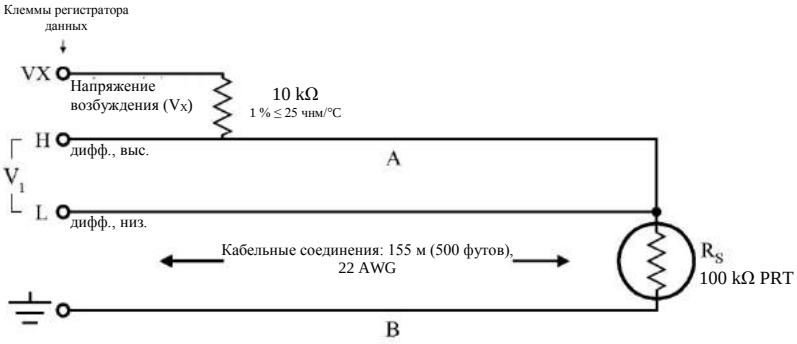
Используемое напряжение возбуждения равно 2,2 В.

Множитель, используемый в **BRHalf3W()**, определяется аналогичным образом, как и в примере *PT100 в четырехжильном полумостовом подключении* *(стр. 238)***.** В данном примере предполагается, что множитель (R_f/R_0) равен 100,93.

Трехжильная полумостовая конфигурация компенсирует сопротивление соединительного кабеля за счет предположения, что сопротивление кабеля A равно сопротивлению кабеля B. Максимально ожидаемое расхождение значений сопротивления кабелей равно 2 %, но по факту, вероятнее всего, составит около 1 %. Сопротивление Rs, рассчитанное с помощью **BRHalf3W()**, по сути составляет R_S плюс разница сопротивлений кабелей A и B. Среднее сопротивление кабеля 22 AWG равно 16,5 Ом на 1000 футов, следовательно, на каждые 500 футов соединительного кабеля приходится номинальное сопротивление 8,3 Ом. Два процента от 8,3 Ом = 0,17 Ом. Если предположить, что кабель B обладает большим сопротивлением, измеренное для ПТС значение сопротивления (R_0 = 100 Ом) в водяной бане составит 100,17 Ом, а сопротивление при 40 °C будет равно 115,71. Измеренное отношение R_S/R_0 равно 1,1551; фактическое отношение — 115,54/100 = 1,1554. Температура, рассчитанная с помощью **PRTCalc()** на основании измеренного отношения, примерно на 0,1 °C ниже фактической температуры ПТС. Этот источник погрешности отсутствует в примере *PT100 в четырехжильном полумостовом подключении* *(стр. 238)***,** так как четырехжильная полумостовая конфигурация используется для измерения сопротивления ПТС.

Терминальный модуль ввода может использоваться для завершения контура, представленного на рисунке *PT100 в трехжильном полумостовом подключении (стр. 242)*. В приложении *Преобразователи сигнала (стр. 647)* приведена информация о доступных модулях TIM.

Рисунок 64. PT100 в трехжильном полумостовом подключении



CRBasic: пример 51. PT100 в трехжильном полумостовом подключении

```
'Этот пример программы демонстрирует измерение ПТС на 100 Ом с помощью трехжильного
'полумостового подключения. Схему электропроводки см. на РИСУНКЕ PT100 в трехжильном полумостовом
'подключении (стр. 242)

Public Rs_Ro
Public Deg_C

BeginProg
   Scan(1,Sec,0,0)

      'BrHalf3W(Dest, Reps,Range1,SEChan,ExChan,MPE,Ex_mV, True,0,250,100.93,0)
      BrHalf3W(Rs_Ro,1,mV25,1,Vx1,1,2200,True,0,250,100.93,0)

      'PRTCalc(Destination,Reps,Source,PRTType,Mult,Offset)
      PRTCalc(Deg_C,1,Rs_Ro,1,1.0,0)

   NextScan
EndProg
```

7.9.15.4. PT100 в четырехжильном подключении с полным мостом

Пример демонстрирует:

- Принцип измерения ПТС в четырехжильной конфигурации с полным мостом

Преимущества:

- Используется в два раза меньше терминалов, настроенных для аналогового ввода, чем в четырехжильном полумостовом подключении.

Пример технических характеристик ПТС:

- α = 0,00392 (ПТС: тип 2)

В данном примере производится измерение ПТС на 100 Ом в четырехжильном подключении с полным мостом (рисунок *PT100 в четырехжильном подключении с полным мостом (стр. 244)*) с помощью команды CRBasic **BRFull().** В данном примере ПТС помещается в термостатическую ванну, а измерение используется в качестве входного значения для контрольного алгоритма.

Как описано в таблице *Контуры мостов сопротивления с напряжением возбуждения (стр. 338)*, результат **BRFull()** равен X,

$$X = 1000\ V_S/V_X$$

где

V_S = измеренное напряжение на выходе моста

V_X = напряжение возбуждения

или

$$X = 1000\ (R_S/(R_S+R_1) - R_3/(R_2+R_3)).$$

На основании рисунка *PT100 в четырехжильном подключении с полным мостом (стр. 244)* сопротивление ПТС (R_S) рассчитывается как:

$$R_S = R_1 \cdot X' / (1-X')$$

где

$$X' = X / 1000 + R_3/(R_2+R_3)$$

Таким образом, чтобы получить значение R_S/R_0 ($R_0 = R_S$ при 0 °C), для команды расчета температуры **PRTCalc()** используемые в **BRFull()** множитель и смещение составляют 0,001 и $R_3/(R_2+R_3)$ соответственно. Множитель (R_f), используемый в алгоритме преобразования моста ($X = R_f\ (X/(X-1))$ для получения значения R_S/R_0, составляет R_1/R_0 или (5000/100 = 50).

Для целей использования необходим контроль термостатической ванны на уровне 50 °C с минимально возможными отклонениями. Требуется высокая разрешающая способность, поэтому контрольный алгоритм будет реагировать на малейшие изменений температуры. Максимальная разрешающая способность достигается, когда на основе температурного диапазона определяется диапазон сигнала (V_S), покрывающий диапазон измерений, выбранный в **BRFull().** Конфигурация с полным мостом позволяет балансировать мост (V_S = 0 V) на уровне контрольной температуры или максимально близко к нему. Таким образом, напряжение на выходе может быть как положительным, так и отрицательным по мере изменения температуры в ванне, что позволяет применять полный диапазон измерений.

Сопротивление ПТС составляет приблизительно 119,7 Ω при 50 °C. Постоянный резистор на 120 Ω балансирует мост на уровне прибл. 51 °C. Напряжение на выходе:

$$V_S = V_S \cdot [R_S/(R_S+R_1) - R_3/R_2+R_3)]$$
$$= V_X \cdot [R_S/(R_S+5000) - 0.023438]$$

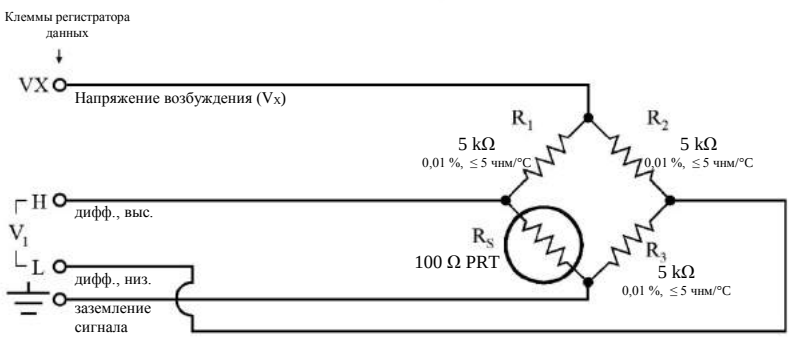
Охватываемый температурный диапазон: 50 °C ±10 °C. При 40 °C R_S составляет приблизительно 115,8 Ω или:

$$V_S = -802.24E\text{-}6\ V_X.$$

Даже при напряжении возбуждения (V_X), равном 2500 мВ, V_S может измеряться с шагом шкалы ±2_5 мВ (40 °C / 115,8 Ω / –2,006 мВ, 60 °C / 123,6 Ω / 1,714 мВ). Выходные показатели при 40 °C и при 51 °C различаются примерно на 2 мВ (181 мкВ / °C). При разрешении 0,33 мкВ в диапазоне ±2_5 мВ это означает, что разрешающая способность температуры составляет 0,0009 °C.

Температурный коэффициент постоянных резисторов (±5 чнм/°C) был выбран, потому что допуск погрешности ±0,01 % будет сохраняться во всем требуемом температурном диапазоне.

Рисунок 65. PT100 в четырехжильном подключении с полным мостом



CRBasic: пример 52. PT100 в четырехжильном подключении с полным мостом

```
'Этот пример программы демонстрирует измерение ПТС на 100 Ом с помощью четырехжильного
'подключения с полным мостом. Схема электропроводки представлена на
'РИСУНКЕ PT100 в четырехжильном подключении с полным мостом (стр. 244) .

Public BrFullOut
Public Rs_Ro
Public Deg_C

BeginProg
  Scan(1,Sec,0,0)

    'BrFull (Dst,Reps,Range,DfChan, Vx1,MPS,Ex,RevEx,RevDf, Settle,Integ,Mult,Offset)
    BrFull(BrFullOut,1,mV25,1,Vx1,1,2500,True,True,0,250,.001,.02344)

    'BrTrans = Rf*(X/(1-X) )
    Rs_Ro = 50 * (BrFullOut/(1 - BrFullOut))

    'PRTCalc(Destination,Reps,Source,PRTType,Mult,Offset)
    PRTCalc(Deg_C,1,Rs_Ro,2,1.0,0)

  NextScan
EndProg
```

7.9.16. Контроль ПЛК — подробная информация

Связанные темы:
- *Управление ПЛК — общие сведения* *(стр. 74)*
- *Управление ПЛК — подробная информация* *(стр. 244)*
- *Модули управления ПЛК — общие сведения* *(стр. 368)*
- *Модули управления ПЛК — списки* *(стр. 648)*
- *Контроль ПЛК — команды* *(стр. 562)*
- Переключаемое выходное напряжение — технические характеристики
- Переключаемое выходное напряжение — общие сведения
- *Переключаемое выходное напряжение — подробная информация* *(стр. 103)*

Этот раздел предполагаться расширить. Далее приведено несколько советов.

- Мастер программирования Short Cut имеет возможность быстрого включения и выключения.
- С помощью CR1000 можно проводить ПИД-регулирование. Для получения дополнительной информации обратитесь к специалистам Campbell Scientific.
- При управлении ПИД-алгоритмом задержка между процессором (вход алгоритма) и управлением (выход алгоритма) нежелательна. Задержка не будет возникать в любом *последовательном режиме* *(стр. 527)* или *режиме поточной обработки* *(стр. 523)*, предполагая, что запрограммирован соответствующий быстрый интервал сканирования, и программа не пропускает сканирование. В последовательном режиме при возникновении какой-либо задачи, из-за которой время обработки выходит за интервал сканирования, будут происходить пропуски сканирования, что может привести к сбою ПИД-регулятора. В режиме поточной обработки при наличии буфера сканирования соответствующего размера, пропуски сканирования не возникают. Однако в ПИД-регуляторе может произойти сбой в команде обработки из-за буфера сканирования.
- Чтобы избежать этих потенциальных проблем, в группе команды по обработке в программе CRBasic применяются **ProcHiPri** и **EndProcHiPri.** Команды по обработке этими командами получают одинаково высокий приоритет с командами измерения и не попадают в буфер сканирования при увеличении времени обработки. ProcHiPri и EndProcHiPri могут быть недоступны в *Редакторе CRBasic*. Но вы можете ввести их в любом случае, и компилятор их распознает.

7.9.17. Последовательный интерфейс ввода/вывода: считывание последовательных данных

CR1000 связывается с интеллектуальными датчиками, которые передают данные измерений с помощью протоколов последовательных данных.

Читать далее. Дополнительную информацию о последовательной связи в CR1000 см. в разделе *Телекоммуникации и вызов данных* *(стр. 391)***.**

7.9.17.1. Введение

Последовательная передача битов (единиц и нулей) является основой последовательной или «серийной» связи. Байт — это пакет последовательных битов. В стандартах RS-232 и TTL используются байты, содержащие по восемь битов. Представим измерительный прибор, передающий байт «11001010» в CR1000. Это осуществляется посредством перевода «11001010» в последовательность повышений и понижений напряжения, которая поступает в CR1000. CR1000 получает и дешифрует эти перепады напряжения как «11001010». Поскольку и прибор, и CR1000 пользуются стандартом RS-232 или TTL, байт успешно проходит маршрут между ними.

Если байт отображается на терминале в том виде, в котором был получен, он будет выведен как символ ASCII / ANSI или управляющий код. В таблице *Эквиваленты ASCII/ANSI* *(стр. 245)* представлен пример символов ASCII/ ANSI и эквивалентных им кодов.

Таблица 41. Эквиваленты ASCII/ANSI

Полученный байт	*Отображаемый символ ASCII*	*Десятичный код ASCII*	*Шестнадцатеричный код ASCII*
00110010	2	50	32
1100010	b	98	62
00101011	+	43	2b

Таблица 41. Эквиваленты ASCII / ANSI			
Полученный байт	***Отображаемый символ ASCII***	***Десятичный код ASCII***	***Шестнадцатеричный код ASCII***
00001101	cr	13	d
00000001	☺	1	1

Читать далее. Полный список кодов ASCII / ANSI и их двоичных/ шестнадцатеричных эквивалентов приведен в *Таблице ASCII /ANSI* *(стр. 637)***.**

Номинальное значение байта тем не менее не всегда является самой важной информацией. Производитель измерительного прибора должен указать, какая информация в байте является важнейшей. Например, могут быть получены два байта: один для символа «2», второй — для символа «b». Сочетание этих символов, «2b», представляет собой шестнадцатеричный код для «+», и именно «+» является важнейшей информацией. Или, допустим, ведущий бит (MSB — старший значащий бит) в каждом из двух байтов пропускается, а остальные биты соединяются, а получившийся «супер-байт» переводится в десятичное значение с учетом оставшихся битов. Разнообразие протоколов ограничивается только числом измерительных приборов на рынке. Подробный пример перевода битов в полезную информацию представлен в приложении *Формат данных FP2* *(стр. 641)***.**

Примечание. Управляющий символ ASCII / ANSI ff (прогон страницы, двоичное представление: 00001100) инициирует очистку экрана терминала. Это может быть неприятным сюрпризом для разработчика, который предпочитает видеть информацию на экране, а не пустой экран. Некоторые сторонние терминальные программы-эмуляторы, такие как *Procomm*, являются полезными инструментами для разработки интерфейсов последовательного ввода/вывода, поскольку способны справляться с этой и прочими специфическими особенностями последовательной связи.

Если в CR1000 поддерживается стандартизированный последовательный протокол, например PakBus® или Modbus, перевод байтов достаточно прост и прозрачен. Однако, если для байтов требуется специальная процедура перевода, в программе CRBasic также необходим специальный код, и время разработки может увеличиться на несколько часов или дней.

7.9.17.2. Порты ввода/вывода

В CR1000 поддерживается двусторонняя последовательная коммуникация с другими измерительными приборами через порты, перечисленные в таблице *Последовательные порты CR1000* *(стр. 247)***.** Последовательные устройства часто поставляются с девятиконтактным разъемом последовательного порта типа D. Подробная информация приведена в схеме расположения выводов производителя. В большинстве случаев используется стандартная девятиконтактная схема RS-232. В подобном случае необходимо:

Подключить датчик RX (прием, контакт 2) к клемме U или C, настроенной для **Tx** (C1, C3, C5, C7).

- Подключить датчик TX (передача, контакт 3) к клемме U или C, настроенной для **Rx** (C2, C4, C6, C8)
- Подключить землю датчика (контакт 5) к земле регистратора данных (клемма **G**)

Примечание. Линии Rx и Tx на девятиконтактных разъемах иногда меняются местами производителем.

Таблица 42. Последовательные порты CR1000		
Последовательный порт	***Уровень напряжения***	***Логика***
RS-232 (9-контактный)	RS-232	Полностью дуплексный асинхронный RS-232
CS I/O (9-контактный)	TTL	Полностью дуплексный асинхронный RS-232
COM1 (C1–C2)	TTL	Полностью дуплексный асинхронный RS-232/TTL
COM2 (C3–C4)	TTL	Полностью дуплексный асинхронный RS-232/TTL
COM3 (C5–C6)	TTL	Полностью дуплексный асинхронный RS-232/TTL
COM4 (C7–C8)	TTL	Полностью дуплексный асинхронный RS-232/TTL
C1	5 В пост. тока	SDI-12
C3	5 В пост. тока	SDI-12
C5	5 В пост. тока	SDI-12
C7	5 В пост. тока	SDI-12
C1, C2, C3	5 В пост. тока	SDM (используется только с периферийными устройствами Campbell Scientific)

7.9.17.3. Протоколы

PakBus — это родной протокол CR1000, обеспечивающий прозрачное управление стандартными двухточечными и сетевыми коммуникациями между ПК и регистраторами данных Campbell Scientific. Modbus и DNP3 являются стандартными для отрасли сетевыми протоколами SCADA, опционально способными работать в рамках CR1000 с минимально требуемыми пользовательскими настройками. PakBus®, Modbus и DNP3 работают, используя **RS-232, CS I/O** и четыре COM-порта. SDI-12 — это протокол, используемый интеллектуальными датчиками и требующий минимальной конфигурации CR1000.

Читать далее. См. разделы *SDI-12: запись (стр. 363)*, *SDI-12: поддержка датчиков (стр. 267)*, *PakBus: общие сведения (стр. 393)*, *DNP3 (стр. 408)* и *Modbus (стр. 411)*.

Для многих измерительных приборов требуются нестандартные протоколы, чтобы связываться с CR1000.

Примечание. Если измерительный прибор или датчик опционально поддерживает SDI-12, Modbus или DNP3, следует рассмотреть возможность использования этих протоколов, прежде чем программировать пользовательский. Эти протоколы высшего уровня стандартны для множества производителей и просты в использовании (в сравнении с пользовательскими протоколами). SDI-12, Modbus и DNP3 также поддерживают системы адресации, позволяющие объединять несколько датчиков в рамках одного порта связи, таким образом повышая эффективность использования ресурсов.

7.9.17.4. Глоссарий терминов для последовательных интерфейсов ввода/вывода

Термин: асинхронный

Передача данных между передающим и принимающим устройством происходит в виде последовательности нулей и единиц. Для корректного «считывания» данных принимающее устройство должно начинать считывание в правильной точке последовательности. При асинхронной связи для этого перед каждым символом и после него вставляется один или несколько стартовых и стоповых битов, которые означают начало и

конец информации (см. *синхронный (стр. 530)*).

Указывает на отсутствие синхронизации передающего и принимающего устройств посредством тактового сигнала.

Термин: скорость передачи в бодах

Скорость, с которой передаются данные.

Термин: обратный порядок байтов

«От старшего к младшему». Наиболее значимая целочисленная переменная располагается в начале цифрового слова, которое считывается слева направо. Процессор в CR1000 работает по схеме MSB или ставит наиболее значимую целочисленную переменную на первое место. См. приложение *Порядок следования байтов (стр. 643)*.

Термин: cr

Разрыв строки

Термин: биты данных

Число битов, используемое для описания данных, и промежуток между стартовым и стоповым битами. В датчиках, как правило, используется 7 или 8 битов данных.

Термин: дуплексный

Протокол последовательной передачи данных. Последовательная связь может быть симплексной, полудуплексной или полностью дуплексной.

Подробнее: *симплексная (стр. 528)*, *дуплексная (стр. 248)*, *полудуплексная (стр. 517)* и *полностью дуплексная (стр. 516)*.

Термин: lf

Перевод строки (от «line **f**eed»). Часто ассоциируется с разрывом строки (<cr>). **<cr><lf>.**

Термин: прямой порядок байтов

«От младшего к старшему». Наиболее значимая целочисленная переменная располагается в конце цифрового слова, которое считывается слева направо. Процессор в CR1000 работает по схеме MSB или ставит наиболее значимую целочисленную переменную на первое место. См. приложение *Порядок следования байтов (стр. 643)*.

Термин: LSB

Младший значащий бит (замыкающий бит). См. приложение *Порядок следования байтов (стр. 643)*.

Термин: посылки и паузы

Уровни сигнала RS-232 логически инвертируются по сравнению с TTL. Различные уровни называются посылками и паузами. Когда речь идет о заземлении сигнала, допустимый уровень напряжения RS-232 для посылки составляет от –3 до –25, а для паузы — от +3 до +25, причем промежуток от –3 до +3 определяется как диапазон перехода, не содержащий информации. Посылка — это логическая 1, отрицательное напряжение. Пауза — это логический 0, положительное напряжение.

Термин: MSB

Старший значащий бит (от «most **s**ignificant **b**it») или ведущий бит). См. приложение *Порядок следования байтов (стр. 643)*.

Термин: RS-232C

Относится к стандарту, используемому для определения аппаратных сигналов и уровней напряжения. CR1000 поддерживает несколько различных опций последовательной логики и уровней напряжения, включая логику RS-232 на уровнях TTL и логику TTL на уровнях TTL.

Термин: RX

Получение

Термин: SP

Пауза

Термин: стартовый бит

Это бит, используемый для указания начала данных.

Термин: стоповый бит

Это конец битов данных. Стоповый бит может выражаться как 1, 1,5 или 2.

Термин: TX

Передача

7.9.17.5. Программирование последовательных интерфейсов ввода/вывода CRBasic

Для передачи или приема сигналов RS-232 или TTL последовательный порт (см. таблицу *Последовательные порты CR1000 (стр. 247)*) должен быть открыт и настроен с помощью команды CRBasic **SerialOpen()**. Для закрытия последовательного порта может использоваться команда **SerialClose()**. Ниже приведен практический совет касательно использования **SerialOpen()** и **SerialClose().** В примере программы CRBasic *Получение строки RS-232 (стр. 254)* демонстрируется использование **SerialOpen().** Более подробная информация приведена в *Справке по редактору CRBasic*.

```
SerialOpen(COMPort,BaudRate,Format,TXDelay,BufferSize)
```

- **COMPort** — в *Справке по редактору CRBasic* приведен полный список COM-портов, доступных для использования в рамках **SerialOpen().**
- **BaudRate** — несоответствие скорости передачи данных в бодах часто является проблемой при разработке нового приложения. Удостоверьтесь, что значения скорости передачи в бодах совпадают. Некоторые разработчики предпочитают использовать фиксированное значение скорости передачи в бодах на начальном этапе разработки. Если устанавливается значение **-nnnn** (где nnnn — скорость передачи в бодах) или **0**, активируется автоматическое определение скорости передачи в бодах. Автобод удобен при использовании портов ввода/вывода CS и RS-232, поскольку в таком случае порты могут использоваться одновременно для телекоммуникаций ПК и датчиков.
- **Format** — определяет тип данных и возможность коммуникации по протоколу PakBus® через COM-порт. Если ожидается, что порт будет считывать показания датчика и поддерживать нормальные телеметрические операции PakBus®, следует использовать аргумент автоматического определения скорости передачи в бодах (**0** или **- nnnn**) и удостовериться, что эта опция поддерживает PakBus® в рамках конкретного приложения.
- **BufferSize** — буфер хранит полученные данные, пока они не будут удалены. Для считывания данных из буфера в виде переменных

используются команды **SerialIn(), SerialInRecord()** и **SerialInBlock()**. Как только данные переведены в переменные, для их форматирования и разбора применяются команды построковой обработки.

SerialClose() должна выполняться до того, как **SerialOpen()** будет использована для повторной конфигурации того же последовательного порта, либо до того, как порт будет использован для связи с ПК.

7.9.17.5.1. Основы программирования последовательных интерфейсов ввода/вывода

SerialOpen()[1]

- Закрывает PPP (если активно)
- Возвращает значение TRUE (истина) или FALSE (ложь) при установке равенства булевой переменной
- Не следует забывать о размере буфера (кольцевая память)

SerialClose()

- Примеры времени закрытия
    - Повторное открытие PPP
    - Завершение настройки новых параметров модема Hayes
    - Завершение вызова модема
- Возвращает значение TRUE (истина) или FALSE (ложь) при установке равенства булевой переменной

SerialFlush()

- Возвращает указатели считывания и записи в начало
- Возвращает значение TRUE (истина) или FALSE (ложь) при установке равенства булевой переменной

SerialIn()[1]

- Может ждать появления строки
- Время ожидания обновляется после получения каждого символа
- **SerialInRecord()** демонстрирует тенденцию к замене устаревшей **SerialIn()**
- Предел размера буфера (одна дополнительная запись + один байт)

SeriallnBlock()[1]

- Для двоичных данных (например, целочисленных переменных, чисел с плавающей точкой, данных с пустыми символами)
- Назначение может относиться к любому типу
- Предел размера буфера (одна дополнительная запись + один байт)

SerialOutBlock()[1,3]

- Двоичная переменная
- Может выполняться в режиме поточной обработки в рамках задачи дискретного измерения (наряду с командами SDM), если для параметра **COMPort** установлена константа, например **COM1, COM2, COM3** или **COM4**, и число байтов также вводится в виде константы.

SerialOut()

- Используется для команд ASCII и известного отклика, например команд модема Hayes.
- Если открыто, возвращает количество переданных байтов. Если не открыто, возвращает 0.

SerialInRecord()[2]

- Может выполняться в режиме поточной обработки в рамках задачи дискретного измерения (наряду с командами SDM), если для параметра **COMPort** установлен постоянный аргумент, например **COM1, COM2, COM3** или **COM4**, и число байтов также вводится в виде константы.
- Упрощает одностороннюю синхронизацию.
- Упрощает работу с протоколами, передающими «запись» данных с известными начальными и/или конечными символами, либо фиксированное число записей в ответ на команду опроса.
- Если начальное и конечное слова отсутствуют, тогда временной интервал становится единственным разделителем между записями. При использовании **COM1, COM2, COM3** или **COM4** обнаруживается временной интервал, составляющий > 100 бит, если размер записей не превышает 256 байт.
- Предел размера буфера (одна дополнительная запись + один байт).

[1] Команды по обработке

[2] Команда по измерению в режиме поточной обработки

[3] Команда по измерению, если выражение определяется как константа

7.9.17.5.2. Основы программирования последовательных интерфейсов ввода/вывода на входе

Приложения, предназначенные для получения данных от другого устройства, обычно включают следующие процедуры. Также, в зависимости от приложения, могут потребоваться другие процедуры.

1. Необходимо знать, что поддерживает датчик, и какие именно данные используются. Большинство датчиков работают с уровнями напряжения TTL и логикой RS-232. Моменты, которые следует учесть:
    - Необходимо подробно ознакомиться с данными, которые будут зарегистрированы.
    - Возможен ли опрос датчика?
    - Передает ли датчик сведения согласно собственному расписанию?
    - Есть ли метки в начале и конце данных? Метки очень удобны для идентификации записи длины переменной.
    - Есть ли в записи знак-разделитель, например точка, пробел или символ табуляции? Знаки-разделители удобны при разборе полученной последовательной строки на готовые к использованию числа.
    - Будет ли датчик передавать несколько строк данных? Перед разбором нескольких строк, как правило, требуется их фильтрация.
    - Как быстро будут переданы данные в CR1000?
    - Важны ли показатели энергопотребления?
    - Рассчитывает ли датчик контрольную сумму? Какого типа? Контрольная сумма необходима для проверки данных на предмет повреждения.
2. Последовательный порт открывается с помощью **SerialOpen().**
    - Пример:
      ```
      SerialOpen(Com1,9600,0,0,10000)
      ```
    - Укажите правильный порт в CRBasic.
    - Корректно подключите устройство к CR1000.
    - Приведите скорость передачи данных порта в соответствие со скоростью передачи данных устройства CRBasic (по возможности следует использовать фиксированную скорость передачи данных в бодах, а не автобод).

3. Последовательные данные получаются в виде строки с помощью **SerialIn()** или **SerialInRecord().**

   – Пример:
   ```
   SerialInRecord(Com2 ,SerialInString,42,0,35, "" , 01)
   ```

   - Объявите строковую переменную достаточной длины, чтобы вместить строку.

   – Пример:
   ```
   Public SerialInString As String * 25
   ```

   - Рассмотрите входящую строку в переменной входящей сроки на *цифровом мониторе* *(стр. 521)*.

Обратите внимание, что **SerialIn()** и **SerialInRecord()** принимают данные. **SerialInRecord()** является лучшим выбором для приема потоковых данных. **SerialIn()** больше подходит для приема дискретных блоков.

4. Разбор (разделение) последовательной строки выполняется с помощью **SplitStr()**

   - Разделяет строку на числовые и/или строковые переменные.
   - Пример:
     ```
     SplitStr(InStringSplit,SeriaIInString,"",2,0)
     ```

   - Объявите массив данных, который будет принимать проанализированные данные.

   – Пример:
   ```
   Public InStringSplit(2) As String
   ```

   – Пример:
   ```
   Public SplitResult(2) As Float
   ```

7.9.17.5.3. Основы программирования последовательных интерфейсов ввода/вывода на выходе

Приложения, предназначенные для передачи данных на другое устройство, обычно включают следующие процедуры. Также, в зависимости от приложения, могут потребоваться другие процедуры.

1. Последовательный порт открывается с помощью **SerialOpen()** и настраивается для целей коммуникации.

   - Параметры устанавливаются в соответствии с требованиями линии связи и последовательного устройства.
   - Пример:
     ```
     SerialOpen(Com1,9600,0,0,10000)
     ```

   - Укажите правильный порт в CRBasic.
   - Корректно подключите устройство к CR1000.
   - Приведите скорость передачи данных порта в соответствие со скоростью передачи данных устройства CRBasic.
   - По возможности следует использовать фиксированную скорость передачи данных в бодах, а не автобод.

2. Постройте выходную строку.

   - Пример:
     ```
     SerialOutString = "*" & "27.435" & "," & "56.789" & "#"
     ```

   - Совет — конкатенацию (добавление) строк следует производить с использованием символа & вместо +.
   - Совет — используйте команду **CHR()** для вставки символов ASCII/ANSI в строку.

3. Выведите строку через последовательный порт (команда **SerialOut()** или **SerialOutBlock()**).
   - Пример:
     ```
     SerialOut(Com1, SerialOutString,"",0,100)
     ```
   - Объявите переменную выходящей строки достаточной длины, чтобы вместить все объединение.
   - Пример:
     ```
     Public SerialOutString As String * 100
     ```

- **SerialOut()** и **SerialOutBlock()** выводят одинаковые данные, за одним исключением: **SerialOutBlock()** передает нулевые значения, в то время как строки **SerialOut()** завершаются нулевым значением.

7.9.17.5.4. Перевод байтов в последовательных интерфейсах ввода/вывода

В переменную ***SerialInString()*** могут попасть от одного до трех основных форматов данных (см. примеры в разделе *Основы программирования последовательного ввода* *(стр. 251)*). Данные могут представлять собой комбинацию форматов или их вариант. Производитель измерительного прибора должен предоставить правила декодирования данных

- **Буквенно-числовой формат** — каждый разряд представлен буквенно-числовым значением. Например, R = буква R, 2 = десятичное число 2. Этот протокол легче всего в плане перевода, поскольку кодировка и перевод идентичны. Как правило, CR1000 программируется таким образом, чтобы разбирать (разделять) строку и распределять значения по переменным.

Пример строки датчика влажности, температуры и давления:

```
SerialInString = "RH= 60.5 %RH T= 23.7 °C Tdf= 15.6 °C Td=
15.6 °C a= 13.0 g/m3 x= 11.1 g/kg Tw= 18.5 °C H2O= 17889
ppmV pw=17.81 hPa pws 29.43 hPa h= 52.3 kJ/kg dT= 8.1 °C"
```

- **Шестнадцатеричные пары** — байты преобразуются в шестнадцатеричные пары, состоящие из цифр от 0 до 9 и букв от a до f. Каждая пара соответствует шестнадцатеричному коду ASCII / ANSI. Некоторые коды переводятся в буквенно-числовые значения, другие — в символы или непечатаемые управляющие символы.

Пример строки датчика температуры:

```
SerialInString = "23 30 31 38 34 0D"
```

что преобразуется в

```
#01 84 cr
```

- **Двоичный формат** — байты обрабатываются бит за битом. Символ 0 (пустой символ, &b00) является значимой частью потоков двоичных данных. Однако в CR1000 используются строки с нулевым символом в конце, поэтому при получении нуля строка завершается. При считывании двоичных данных завершение обычно получается досрочным. Для устранения этой проблемы следует использовать **SerialInBlock()** или **SerialInRecord()** при считывании двоичных данных. Входящая строковая переменная должна быть массивом с заданным типом данных **As Long**, как в следующем примере:

```
Dim SerialInString As Long
```

7.9.17.5.5. Аспекты памяти для последовательных интерфейсов ввода/вывода

При получении и обработке последовательных данных следует обратить внимание на некоторые особенности, касающиеся памяти.

- **Последовательный буфер.** Буфер последовательного порта, объявленный в **SerialOpen(),** должен быть достаточно большим, чтобы вместить все данные, которые будут переданы устройством. Буфер хранит данные для дальнейшего перевода в переменные. При необходимости можно распределить дополнительную память для буфера, однако следует учесть, что при добавлении памяти к буферу сокращается объем *памяти окончательных данных* *(стр. 515)***.**

Примечание. Что касается **SerialInRecord()** в режиме поточной обработки с параметром ***NBytes*** (число байтов) = 0:

Чтобы в рамках последовательности дискретного измерения определить необходимый объем для распределения в буферы **Scan()** (по умолчанию 3), команда **SerialInRecord()** выделяет объем буфера, указанный в **SerialOpen()** (по умолчанию 10 000, с излишком), или 3 • 10 000 = 30 кБ буферного пространства. Таким образом, не только обеспечивается распределение достаточного числа байтов в **SerialOpen()** (количество байтов на запись • ((records/Scan)+1) + как минимум один дополнительный байт), но и обосновывается недопущение слишком большого объема буфера. (Обратите внимание, что при ненулевом параметре ***NumberOfBytes*** команда **SerialInRecord()** распределяет указанное именно здесь, а не в **SerialOpen()** число байтов.)

- **Объявления переменных** — переменные, используемые для получения данных из последовательного буфера, могут быть объявлены как **Public** или **Dim**. При объявлении переменной как **Dim** используется меньшая полоса частот электросвязи. Если в программе просматриваются общие переменные («public»), то вся таблица **Public** преобразуется согласно интервалу обновления. Если таблица **Public** достаточно велика, полоса частот электросвязи может быть занята настолько, что другие таблицы данных не будут задействованы.
- **Объявления строк** — строковые переменные требовательны к памяти. Определите, насколько велики строки, и объявите переменные достаточной длины, чтобы вместить строку. Если датчик передает несколько строк одновременно, рассмотрите возможность объявления отдельной строковой переменной и считывайте входящие строки по одной.

  CR1000 корректирует объявленный размер строк в сторону увеличения. Один байт всегда добавляется к объявленной длине, а затем длина увеличивается еще на 1–3 байта, чтобы ее можно было разделить на 4 без остатка.

  Объявленная длина строки (а не число символов) определяет потребление памяти при записи строк в память. В результате длинные строки, не заполненные символами, тратят значительный объем памяти.

7.9.17.5.6. Демонстрационная программа

Пример CRBasic *Получение строки RS-232* *(стр. 254)* приводится в качестве упражнения по программированию последовательного ввода/вывода.
В примере требуется только CR1000 и отдельная проводная перемычка между **COM1 Tx** и **COM2 Rx**. Программа моделирует передачу сведений с датчика температуры и относительной влажности по протоколу RS-232 (смоделированные данные поступают из **COM1** в виде буквенно-числовой строки).

CRBasic: пример 53. Получение строки RS-232

```
'В этом примере программы демонстрируются особенности последовательного ввода/вывода CR1000
посредством:
'1. Моделирования последовательного датчика
'2. Передачи последовательной строки через COM1 TX.

'Последовательная строка поступает в COM2 RX через проводную перемычку. Смоделированная
температура воздуха = 27,435 F, относительная влажность = 56,789 %.

'Электропроводка:
'COM1 TX (C1) ------ COM2 RX (C4)

'Объявления последовательных выходов
Public TempOut As Float
Public RhOut As Float

'Объявление строковой переменной достаточной длины, чтобы вместить выходную строку.
Public SerialOutString As String * 25

'Объявления последовательных входов
'Объявление строковой переменной достаточной длины, чтобы вместить входящую строку
Public SerialInString As String * 25

'Объявление строк для приема проанализированных данных. Если проанализированные данные являются
строго числовыми, этот
'массив может быть объявлен как «Float» или «Long»
Public InStringSplit(2) As String
Alias InStringSplit(1) = TempIn
Alias InStringSplit(2) = RhIn

'Основная программа
BeginProg

  'Моделирование датчика температуры и относительной влажности
  TempOut = 27.435                    'Настройка смоделированной температуры для передачи
  RhOut = 56.789                      'Настройка смоделированной относительной влажности для
                                      передачи

  Scan(5,Sec, 3, 0)

  'Код последовательного выхода
  'Передача строки «*27.435,56.789#» на выходе COM1
  Serial Open(Com1,9600,0,0,10000)    'Открытие последовательного порта

  'Построение выходной строки
  SerialOutString = "*" & TempOut & "," & RhOut & "#"

  'Выходная строка через последовательный порт
  SerialOut(Com1, SerialOutString,"",0,100)

  'Код последовательного входа
  'Получение строки «27.435,56.789» через COM2
  'Символы * и # используются в качестве фильтров
  SerialOpen(Com2,9600,0,0,10000)     'Открытие последовательного порта
```

```
  Получение последовательных данных в виде строки
  42 — это ASCII-код для "*", 35 — код для «#»
  SerialInRecord(Com2,SerialInString,42,0,35, "" ,01)

  Разбор последовательной строки
  SplitStr(InStringSplit(),SerialInString,"",2,0)

 NextScan
EndProg
```

7.9.17.6. Проверка приложения последовательного ввода/вывода

Стандартной проблемой при разработке приложения последовательного ввода/вывода является нехватка непосредственно имеющихся в распоряжении последовательных устройств, помогающих разрабатывать и тестировать программы. С помощью программы *HyperTerminal* разработчик может смоделировать выход последовательного устройства или собрать данные последовательного ввода.

Примечание. *HyperTerminal* поставляется как служебная программа вместе с *Windows XP* и более ранними версиями Windows. Программа *HyperTerminal* не поставляется с более поздними версиями Windows, но может быть приобретена отдельно на сайте http://www.hilgraeve.com. HyperTerminal автоматически преобразует двоичные данные в ASCII непосредственно на экране. Двоичные данные могут быть считаны, сохранены в файл, а затем просмотрены с помощью шестнадцатеричного редактора. Также доступны терминальные эмуляторы сторонних поставщиков, упрощающие считывание двоичных или шестнадцатеричных данных.

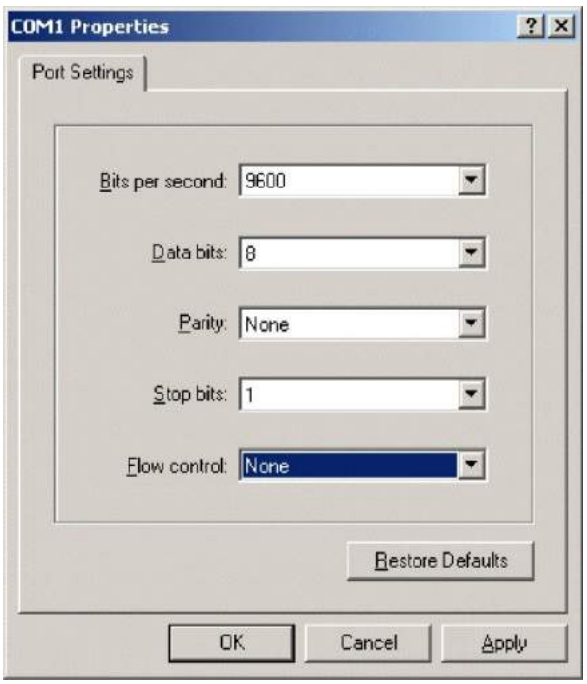
7.9.17.6.1. Конфигурация HyperTerminal

Файл экземпляра *HyperTerminal* создается в меню **Пуск | Все программы | Стандартные | Связь | HyperTerminal.** На рисунках с *HyperTerminal: описание подключения* *(стр. 256)* по *HyperTerminal: настройка ASCII* *(стр. 258)* представлены соответствующие окна. Введите имя экземпляра и нажмите **OK**.

Рисунок 66. HyperTerminal: описание нового подключения

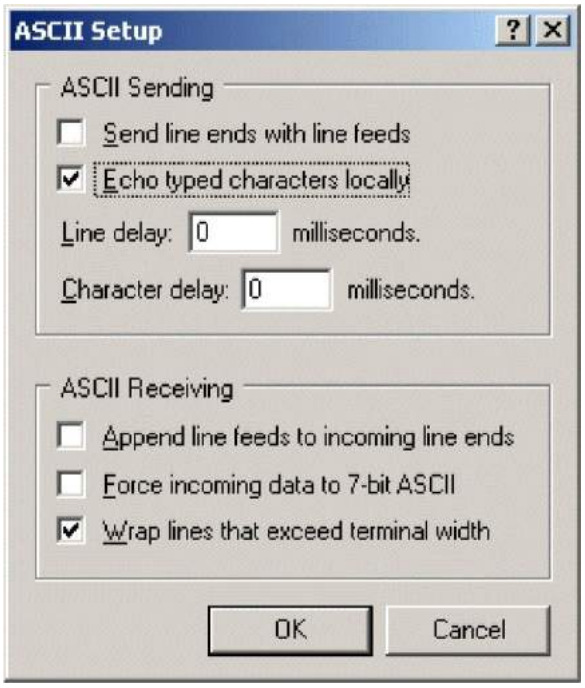
Рисунок 67. HyperTerminal: настройки подключения

Рисунок 68. HyperTerminal: вкладка настроек COM-порта



Нажмите **Файл | Свойства | Параметры настройки | Настройка ASCII**... и установите указанные значения.

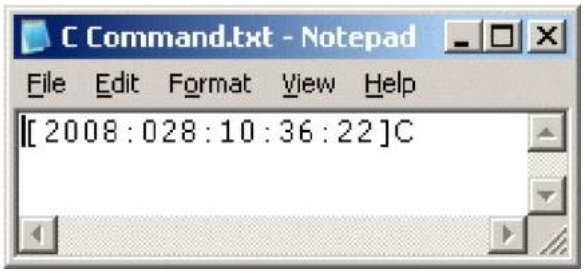
Рисунок 69. HyperTerminal: настройка ASCII



7.9.17.6.2. Создание текстового файла для отправки

Создайте файл, из которого будет передаваться последовательная строка. Файл, представленный на рисунке *HyperTerminal: пример текстового файла для отправки* *(стр. 258)*, передаст строку **[2008:028:10:36:22]C** в CR1000. Используйте *Блокнот*® (служебная программа Microsoft® *Windows*®) или другой текстовый редактор, который не будет размещать в файле скрытые символы.

Рисунок 70. HyperTerminal: пример текстового файла для отправки

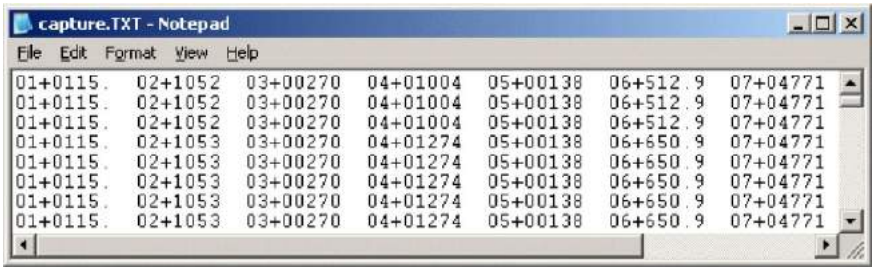


Чтобы передать файл, выберите файл в меню **Передача | Отправить текстовый файл | Поиск** и нажмите **OK**.

7.9.17.6.3. Создание файла захвата текста

На рисунке *HyperTerminal: пример файла захвата текста* *(стр. 259)* представлен файл захвата *HyperTerminal*, содержащий некоторое количество информации. До начала использования файл пуст.

Рисунок 71. HyperTerminal: пример файла захвата текста



Активируйте захват текста в меню **Передача | Захват текста | Поиск**, выберите файл и нажмите **OK**.

7.9.17.6.4. Последовательный ввод/вывод: пример II

В примере CRBasic *Измерительные датчики / Отправка данных RS-232* *(стр. 259)* продемонстрировано использование особенностей последовательного ввода/вывода CR1000.

Пример. Энергетическая компания имеет обширную сеть старых регистраторов данных модели CR510, в которую должны быть внедрены новые регистраторы данных CR1000. Регистраторы CR510 запрограммированы на вывод данных в устаревшем формате Campbell Scientific «Printable ASCII» (печатаемые ASCII-символы), удовлетворяющем требованиям пользовательской сети сбора информации. Администратор сети предпочитает синхронизировать часы CR510 с центральным компьютером, используя устаревшую C-команду Campbell Scientific. Регистратор данных CR510 жестко запрограммирован на вывод печатаемых ASCII-символов и распознавание **C**-команды. Тем не менее, чтобы регистраторы данных CR1000 могли выводить и принимать те же строки ASCII, необходимо программирование со стороны пользователя. Аналогичная программа может использоваться для моделирования регистраторов данных CR10X и CR23X.

Решение. В примере CRBasic *Измерительные датчики / Отправка данных RS-232* *(стр. 259)* производится импорт и экспорт последовательных данных через порт CR1000 RS-232. Ожидается, что импортированные данные будут представлены в виде устаревшей **C**-команды Campbell Scientific для установки времени. Экспортированные данные представлены в устаревшем формате печатаемых ASCII-символов Campbell Scientific.

Примечание. Девятиконтактный порт RS-232 может использоваться для загрузки программы CR1000, если скорость передачи данных **SerialOpen()** совпадает со скоростью передачи данных *вспомогательного ПО регистратора данных* *(стр. 654)*. Тем не менее система двусторонней связи PakBus® время от времени будет провоцировать непредусмотренную отправку пакетов PakBus® из порта RS-232 устройства CR1000 как минимум в течение 40 секунд по завершении последнего сеанса связи с PakBus®. В результате этого появятся «помехи» в предполагаемом выходящем сигнале данных.

Мониторинг порта RS-232 устройства CR1000 осуществляется с помощью программы *HyperTerminal*, как описано в разделе *Конфигурация HyperTerminal* *(стр. 256)*. Отправьте файл с **C**-командой, чтобы установить время в соответствии с текстом в файле.

Примечание. Фактическое время в файле *HyperTerminal* не будет обновляться автоматически. Файл является всего лишь смоделированным источником времени в отдельно взятом примере.

CRBasic: пример 54. Измерительные датчики / Отправка данных RS-232

```
В этом примере программы демонстрируются импорт и экспорт последовательных данных через
порт RS-232устройства CR1000 Ожидается, что импортированные данные будут представлены в виде
устаревшей C-команды Campbell Scientific для установки времени.

[YR:DAY:HR:MM:SS]C

Экспортированные данные представлены в устаревшем формате печатаемых ASCII-символов Campbell
Scientific:

' 01+0115. 02+135 03+00270 04+7999 05+00138 06+07999 07+04771

Объявления
Видимые переменные
Public StationID
Public KWH_In
Public KVarH_I
Public KWHHold
Public KVarHold
Public KWHH
Public KvarH
Public InString As String * 25
Public OutString As String * 100

Скрытые переменные
Dim i, rTime(9), OneMinData(6), OutFrag(6) As String
Dim InStringSize, InStringSplit(5) As String
Dim Date, Month, Year, DOY, Hour, Minute, Second, uSecond
Dim LeapMOD4, LeapMOD100, LeapMOD400
Dim Leap4 As Boolean, Leap100 As Boolean, Leap400 As Boolean
Dim LeapYear As Boolean
Dim ClkSet(7) As Float

Поминутная таблица данных
DataTable(OneMinTable,true,-1)
   OpenInterval                  устанавливает интервал, идентичный CR510
   DataInterval(0,1,Min,10)
   Totalize(1, KWHH,FP2,0)
   Sample(1, KWHHold,FP2)
   Totalize(1, KvarH,FP2,0)
   Sample(1, KVarHold,FP2)
   Sample(1, StationID,FP2)
EndTable

Таблица данных записей установки времени
DataTable(ClockSetRecord,True,-1)
  Sample(7,ClkSet(), FP2)
EndTable
```

```
Подпрограмма для конвертации форматов данных (из дня года в месяц и день).
Sub DOY2MODAY

   Сохранение значений года, дня года, часа, минуты и секунды в области входов.
   Year = InStringSplit(1)
   DOY = InStringSplit(2)
   Hour = InStringSplit(3)
   Minute = InStringSplit(4)
   Second = InStringSplit(5)
   uSecond = 0

   Проверка на високосный год:
   Если Year Mod 4 = 0, и Year Mod 100 <> 0, то это високосный год ИЛИ
   Если Year Mod 4 = 0, Year Mod 100 = 0, и Year Mod 400 = 0, то это
   високосный год

   LeapYear = 0                          Сброс области состояния года

   LeapMOD4 = Year MOD 4
   LeapMODlOO = Year MOD 100
   LeapMOD400 = Year MOD 400
   If LeapMOD4 = 0 Then Leap4 = True Else Leap4 = False
   If LeapMOD100 = 0 Then Leap100 = True Else Leap100 = False
   If LeapMOD400 = 0 Then Leap400 = True Else Leap400 = False

   If Leap4 = True Then
     LeapYear = True
     If Leap100 = True Then
       If Leap400 = True Then
         LeapYear = True
       Else
         LeapYear = False
       EndIf
     EndIf
   Else
      LeapYear = False
   EndIf

   Если год високосный, использовать этот раздел.
   If (LeapYear = True) Then
      Select Case DOY
      Case Is < 32
        Month = 1
        Date = DOY
      Case Is < 61
        Month = 2
        Date = DOY + -31
      Case Is < 92
        Month = 3
        Date = DOY + -60
      Case Is < 122
        Month = 4
        Date = DOY + -91
      Case Is < 153
        Month = 5
        Date = DOY + -121
      Case Is < 183
        Month = 6
        Date = DOY + -152
```

```
      Case Is < 214
        Month = 7
        Date = DOY + -182
      Case Is < 245
        Month = 8
        Date = DOY + -213
      Case Is < 275
        Month = 9
        Date = DOY + -244
      Case Is < 306
        Month = 10
        Date = DOY + -274
      Case Is < 336
        Month = 11
        Date = DOY + -305
      Case Is < 367
        Month = 12
        Date = DOY + -335
   EndSelect

Если год не високосный, использовать этот раздел.
 Else
   Select Case DOY
      Case Is < 32
        Month = 1
        Date = DOY
      Case Is < 60
        Month = 2
        Date = DOY + -31
      Case Is < 91
        Month = 3
        Date = DOY + -59
      Case Is < 121
        Month = 4
        Date = DOY + -90
      Case Is < 152
        Month = 5
        Date = DOY + -120
      Case Is < 182
        Month = 6
        Date = DOY + -151
      Case Is < 213
        Month = 7
        Date = DOY + -181
      Case Is < 244
        Month = 8
        Date = DOY + -212
      Case Is < 274
        Month = 9
        Date = DOY + -243
```

```
      Case Is < 305
        Month = 10
        Date = DOY + -273
      Case Is < 336
        Month = 11
        Date = DOY + -304
      Case Is < 366
        Month = 12
        Date = DOY + -334
    EndSelect
  EndIf
EndSub

//////////////////////////// ПРОГРАММА ////////////////////////////
BeginProg
  StationID = 4771
  Scan(1,Sec, 3, 0)

    //////////////////Раздел измерений//////////////////////
    PulseCount(KWH_In, 1, 1, 2, 0, 1, 0) 'Активируйте эту строку в рабочей программе
    KWH_In = 4.5                  Моделирование — удалите эту строку из рабочей программы

    PulseCount(KVarH_I, 1, 2, 2, 0, 1, 0) 'Активируйте эту строку в рабочей программе
    KVarH_I = 2.3                 Моделирование — удалите эту строку из рабочей программы
    KWHH = KWH_In
    KvarH = KVarH_I
    KWHHold = KWHH + KWHHold
    KVarHold = KvarH + KVarHold

    CallTable OneMinTable

    ////////////////////Раздел последовательного ввода/вывода////////////////////
    SerialOpen(ComRS232,9600,0,0,10000)

    ///////////////Раздел установки времени для последовательного входа///////////////
    Принимается старая C-команда — [2008:028:10:36:22]C — разбор, обработка, установка
    времени (Примечание. Chr(91) = «[», Chr(67) = «C»)
    Serial InRecord(ComRS232 ,InString,91,0,67, InStringSize, 01)

    If InStringSize <> 0 Then
      SplitStr(InStri ngSplit,InString,"",5,0)
      Call DOY2MODAY                              Вызов подпрограммы для преобразования дня года
                                                  в месяц и день
      ClkSet(1) = Year
      ClkSet(2) = Month
      ClkSet(3) = Date
      ClkSet(4) = Hour
      ClkSet(5) = Minute
      ClkSet(6) = Second
      ClkSet(7) = uSecond
      Примечание: Для массива ClkSet требуются значения года, месяца, дня, часа, минуты, секунды,
      миллисекунды
      ClockSet(ClkSet())
      CallTable(ClockSetRecord)
   EndIf
```

```
    ////////////////Раздел последовательного вывода////////////////////
    Построение старого формата «печатаемых ASCII-символов» Campbell Scientific и вывод на COM1

    Считывание времени регистратора данных
    RealTime(rTime)
    If TimeIntoInterval(0,5,Sec) Then
      Загрузка данных таблицы OneMinData для преобразования в печатаемые ASCII-символы
      GetRecord(OneMinData(),OneMinTable,1)

      Присвоение знака +/-
      For i=1 To 6
        If OneMinData(i) < 0 Then
          Примечание. chr45 — это знак «-»
          OutFrag(i)=CHR(45) & FormatFloat(ABS(OneMinData(i)),"%05g")
        Else
          Примечание. chr43 — это знак «+»
          OutFrag(i)=CHR(43) & FormatFloat(ABS(OneMinData(i)),"%05g")
        EndIf
      Next i

      Конкатенация строки печатаемых ASCII-символов, затем вывод строки на RS-232
      (первые 2 поля — ID, ччмм):
      OutString = "01+0115." & " 02+" & FormatFloat(rTime(4),"%02.0f") & _
        FormatFloat(rTime(5) , "%02 .0f")
      OutString = OutString & " 03" & OutFrag(1) & " 04" & OutFrag(2) & _
        " 05" & OutFrag(3)
      OutString = OutString & " 06" & OutFrag(4) & " 07" & OutFrag(5) & _
        CHR(13) & CHR(10) & "" вставка пустого значения CR LF

      Передача строки печатаемых ASCII-символов через порт RS-232
      SerialOut(ComRS232 ,OutString,"",0,220)
    EndIf

  NextScan
EndProg
```

7.9.17.7. Последовательный ввод/вывод: вопросы и ответы

B: Я пишу программу CR1000 для передачи последовательной команды, содержащей нулевой символ. Передаваемая строка выглядит следующим образом:

```
CHR(02)+CHR(01)+"CWGT0"+CHR(03)+CHR(00)+CHR(13)+CHR(10)
```

Как регистратор обрабатывает *нулевой* символ?
Есть ли способ заставить *регистратор* передать эту строку?

A: Создаваемые с помощью CRBasic строки завершаются нулевым символом. Сложение строк означает, что вторая строка начнется с первого нулевого символа, найденного в первой строке.

Используйте команду **SerialOutBlock()**, позволяющую передавать нулевые символы, как показано ниже.

```
SerialOutBlock(COMRS232, CHR(02) + CHR(01) + "CWGT0" +
CHR(03), 8)
SerialOutBlock(COMRS232, CHR(0),1)
SerialOutBlock(COMRS232, CHR(13) + CHR(10), 2)
```

B: Поясните вкратце, когда CR1000 задействует порт RS-232. Как я понимаю, установлен параметр «всегда включено». Что делать при наличии маркеров? Активирует ли команда **SerialOpen()** другие циклы включения/выключения питания?

A: Порт RS-232 остается включенным при следующих условиях:

- Если задан параметр **Питание RS-232** *(стр. 627)*
- Если в программе используется **SerialOpen()** с аргументом ***COMRS232***

Оба условия активируют интерфейс и оставляют его включенным без отсчета времени простоя. Если **SerialClose()** применяется после **SerialOpen(),** порт деактивируется и остается в таком состоянии до поступления символов.

В условиях нормальной эксплуатации порт отключается и ожидает ввода. После получения входящих сведений начинается отсчет 40-секундного времени ожидания, по истечении которого произойдет выключение. 40-секундный интервал ожидания обычно обходится при связи с *вспомогательным ПО регистратора данных (стр. 95)*, поскольку программа передает информацию как часть протокола, сообщающего CR1000, что можно отключать порт.

В «спящем» режиме при выключенном интерфейсе настройки прибора позволяют обнаружить активность и «разбудить» его, однако недостаток заключается в потере первого символа входящего потока данных. PakBus® учитывает это и использует «кольцевые пакеты», в начале которых располагаются дополнительные байты синхронизации. Благодаря этому **SerialOpen()** оставляет интерфейс включенным, так что входящие байты не теряются.

Если в CR1000 есть данные для отправки через порт RS-232, не являющиеся ответом на полученный пакет (например, они не имеют отношения к отправке маркера), устройство включит интерфейс, отправит данные и вернется в «спящий» режим без 40-секундного времени ожидания.

B: Как можно сослаться на специальные символы в строке?

A: Третье «измерение» строковой переменной предоставляет доступ к части строки после указанной позиции. Например, если

```
TempData = "STOP"
```

то

```
TempData(1,1,2) = "TOP"
TempData(1,1,3) = "OP"
TempData(1,1,1) = "STOP"
```

Для управления операциями с отдельными символами объявите строку размером 1. Эта односимвольная строка затем будет использоваться для поиска специальных символов. В следующем примере первый символ строки ***LargerString*** определяется и используется для управления логикой программы:

```
Public TempData As String * 1
  TempData = LargerString
  If TempData = "S" Then...
```

Отдельный символ может быть вызван из любой позиции строки. В следующем примере вызывается пятый символ строки:

```
Public TempData As String * 1
TempData = LargerString(1,1,5)
```

B: Как можно считать расширенные символы с помощью **SerialIn(), SerialInBlock()** и **SerialInRecord()**?

A: Откройте порт в двоичном режиме (режим 3) вместо режима активации PakBus (режим 0).

B: Тесты с применением осциллографа показали, что датчик реагирует быстро, однако данные задерживаются внутри CR1000 примерно на 30 мс. Символы в начале ответного сообщения датчика, поступающие уже через 5 мс, были недоступны для программы в течение ок. 30 мс; фактически в последовательном буфере не было данных в течение 30 мс.

A: Вследствие внутренней буферизации в CR1000 и/или внешних интерфейсах данные могут поступать в буфер последовательного порта с задержкой вплоть до 50 мс (в зависимости от используемого последовательного порта). Это следует учитывать при установке времени ожидания для команд **SerialIn()** и **SerialOut()** или пользовательских значений времени ожидания в структурах с применением команды **SerialInChk()**.

B: Каковы условия прерывания, предотвращающие сохранение входящих данных?

A: Условия прерывания:

- Получен аргумент ***TerminationChar***
- Выполнен аргумент ***MaxNumChars***
- Превышен аргумент ***TimeOut***

SerialIn() НЕ останавливает сохранение при получении нулевого (&h00) символа (если только нулевой символ не задан в качестве завершающего). В случае строковой переменной полученный нулевой символ завершит строку, однако символы после нулевого продолжат приниматься в области переменной, пока не будет выполнено одно из условий прерывания. В дальнейшем доступ к этим символам можно получить с помощью команды **MoveBytes()**, если необходимо.

B: Как можно использовать переменную, заданную с помощью **SerialIn()**, более чем в одной последовательности, при этом избегая ее появления в других последовательностях, если присутствуют старые данные?

A: Простая мера предосторожности заключается в отказе от использования целевой переменной более чем в одной последовательности во избежание ее появления там, где содержатся старые данные. Однако это не всегда возможно, и решение ключевой проблемы может быть более изящным.

Когда данные поступают независимо от выполнения программы CRBasic, как это происходит с потоковыми данными, необходимо принять меры по обеспечению своевременного обновления входящих данных для их дальнейшего использования в соответствующих процессах. Если задачи записи и считывания данных разделены, необходимо регулировать поток данных с помощью функций принудительного контроля, таких как использование флажков или точек весов с привязкой по времени, которые могут быть получены из таблицы данных.

Команда **SerialIn()** не обладает уникальными особенностями в плане понимания корректной процедуры записи и считывания глобальных переменных с помощью нескольких последовательностей. **SerialIn()** осуществляет запись в массив символов. Многие другие команды записывают данные в массив значений (символы, числа с плавающей точкой, длинные строки), среди них: **Move(), MoveBytes(), GetVariables(), SerialInRecord(), SerialInBlock().** Во всех случаях при записи в массив значений важно понимать, что именно считывается и осуществляется ли считывание асинхронно, иными словами, происходит ли считывание из другой задачи, запрашивающей данные в момент их записи, будь то другой прибор, считывающий данные, например *LoggerNet*, или другая последовательность/задача в пределах того же устройства. Если процесс достаточно быстр, как, например, команда **Move()** и есть асинхронный процесс, считывающий данные, это может иметь крайне негативные

последствия, поскольку «считывание старых данных» будет происходить не так часто, однако скрытый вред окажется существеннее именно ввиду редкости события.

7.9.18. Последовательный ввод/вывод: поддержка датчика SDI-12 — программные ресурсы

Связанные темы:
- *Поддержка датчика SDI-12 — общие сведения* *(стр. 72)*
- *Поддержка датчика SDI-12 — подробная информация* *(стр. 363)*
- *Последовательный интерфейс ввода/вывода: Поддержка датчика SDI-12 — программные ресурсы* *(стр. 267)*
- *Поддержка датчика SDI-12 — команды* *(стр. 555)*

В таблице *Определения терминала CR1000* *(стр. 76)* приведены назначения **C**-выводов для входа SDI-12. К каждому настроенному выводу можно подключить несколько датчиков SDI-12. Если к одному терминалу подключено несколько датчиков, каждый из них должен иметь уникальный адрес. Соответствующие стандарту SDI-12 вер. 1.3 датчики получают адреса от **0** до **9, от a** до **z** и от **A** до **Z**. Пример программы CRBasic, иллюстрирующий смену адреса SDI-12 в процессе работы, приведен в публикации Campbell Scientific *Регуляторы зарядного тока PS200/CH200 12 В*, доступной на сайте *www. campbellsci. com.*

CR1000 поддерживает связь с SDI-12 в двух режимах — прозрачном и программируемом.

- Прозрачный режим упрощает установку датчика и устранение проблем. Он позволяет вводить команды вручную и полностью анализировать реакцию датчика. В прозрачном режиме данные не записываются.
- В программируемом режиме автоматизируется значительная часть протокола SDI-12 и обеспечивается запись данных.

7.9.18.1. SDI-12: прозрачный режим

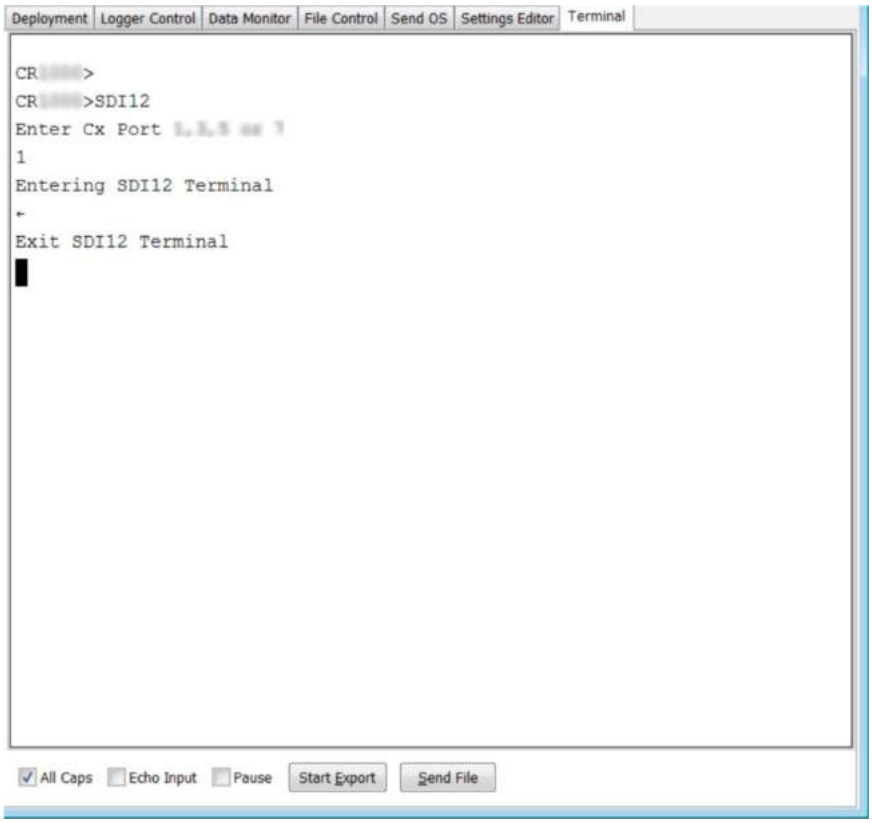
С помощью прозрачного режима операторы системы могут вручную запрашивать и вводить настройки измерительных приборов. Прозрачный режим удобен при устранении проблем в системах SDI-12, поскольку он позволяет напрямую связываться с измерительными приборами.

В прозрачном режиме может быть необходимо подождать подачи команд из программируемого режима для завершения операции перед отправкой ответа. Программы CR1000 не могут выполняться в прозрачном режиме. Для активации прозрачного режима может потребоваться разблокировка защиты CR1000.

Вход в прозрачный режим осуществляется в процессе телекоммуникационной связи ПК с CR1000 посредством программы-эмулятора терминала. Доступ легко получается через эмулятор терминала. В программе Campbell Scientific DevConfig есть служебная функция терминала, аналогично другим *вспомогательным программам регистратора данных* *(стр. 95)***.** Использование экранной клавиатуры невозможно.

Для входа в прозрачный режим SDI-12 необходимо активировать эмулятор терминала во вспомогательном ПО регистратора данных, как показано на рисунке *Вход в прозрачный режим SDI-12* *(стр. 268)***.** Нажмите и удерживайте **Enter**, пока CR1000 не выведет в ответ запрос **CR1000>.** Введите **SDI12** в поле запроса и нажмите **Enter**. В ответ на это запрос **Ввод порта Cx** выведет список доступных портов. Введите номер порта, присвоенный выводу, к которому подключен датчик SDI-12. Так, например, порт **1** указывается для вывода **C1**. Ответ **Ввод терминала SDI12** показывает, что прозрачный режим SDI-12 активен и готов передавать команды SDI-12 и отображать ответы.

Рисунок 72. Вход в прозрачный режим SDI-12



7.9.18.1.1. SDI-12: команды прозрачного режима

Команды имеют три компонента:

- Адрес датчика (**a**) — отдельный символ, он же является первым символом команды. Производитель обычно присваивает датчикам адрес по умолчанию «0». В команде запроса адреса используется подстановочный знак (?). Некоторые производители могут разрешить его использование в других командах.
- Тело команды (например: **M1**) представляет собой заглавную букву («команду») и следующие за ней буквенно-числовые квалификаторы.
- Завершение команды (**!**) — восклицательный знак.

Активный датчик реагирует на каждую команду. Отклики имеют несколько стандартных форм и завершаются символами <CR><LF> (разрыв строки/перевод строки).

Команды SDI-12 и отклики определяются группой поддержки SDI-12 (www.sdi-12.org) и представлены в сводном виде в таблице *Стандартный набор команд и откликов SDI-12* *(стр. 269)*. Производители датчиков определяют, какие команды поддерживаются. Наиболее частые команды подробно описаны в таблице *Команды SDI-12 для прозрачного режима* *(стр. 269)*.

Таблица 43. Команды SDI-12 для прозрачного режима

Имя команды	***Синтаксис команды***[1]	***Отклики***[2] ***с примечаниями***
Размыкание	Непрерывный интервал (12 мс минимум)	Н/п
Запрос адреса	**?!**	**a<CR><LF>**
Подтверждение активации	**a!**	**a<CR><LF>**
Смена адреса	**aAb!**	**b<CR><LF>** (поддержка этой команды требуется, только если датчик допускает программно изменяемые адреса)
Запуск параллельных процессов измерения	**aC!**	**atttnn<CR><LF>**
Дополнительные параллельные процессы измерения	**aC1! ... aC9!**	**atttnn<CR><LF>**
Дополнительные параллельные процессы измерения и запрос CRC	**aCC1! ... aCC9!**	**atttnn<CR><LF>**
Отправка данных	**aD0! ... aD9!**	**a<значения><CR><LF> или a<значения><CRC><CR><LF>**
Отправка идентификатора	**al!**	**allccccccccmmmmmmvvvxxx...xx<CR><LF>.** Например, **013CampbellCS1234003STD.03.01** означает следующее: адрес = 0, номер версии протокола SDI-12 = 1.3, производитель — Campbell Scientific, CS1234 — номер модели датчика (вымышленный в данном примере), 003 — номер версии датчика, STD.03.01 указывает, что номер модификации датчика равен .01.
Начало измерения[3]	**aM!**	**atttn<CR><LF>**
Начало измерения и запрос CRC[3]	**aMC!**	**atttn<CR><LF>**
Дополнительные измерения[3]	**aM1! ... aM9!**	**atttn<CR><LF>**
Дополнительные измерения и запрос CRC[3]	**aMC1! ... aMC9!**	**atttn<CR><LF>**
Непрерывные измерения	**aR0! ... aR9!**	**a<значения><CR><LF>** (формат аналогичен **D**-командам)
Непрерывные измерения и запрос CRC	**aRC0! ... aRC9!**	**a<значения><CRC><CR><LF>** (формат аналогичен **D**-командам)
Начало проверки[3]	**aV!**	**atttn<CR><LF>**

[1]Если завершающий элемент '**!**' отсутствует, команда не будет передана. Команда CRBasic **SDI12Recorder()**, тем не менее, продолжит собирать данные на основе ранее переданной **C!**-команды.

[2]Полную строку отклика можно получить с помощью команды **SDI12Recorder()** и объявления целевой переменной (***Destination***) в качестве **строки (String)**.

[3]Результатом этой команды может стать сервисный запрос.

SDI-12: команды адреса

Команды адреса и идентификатора запрашивают метаданные датчика. При использовании этих команд следует подключать только одно измерительное устройство.

?!

Запрашивает адрес датчика. Ответом является адрес, **a.**

Синтаксис:

?!

aAb!

Изменяет адрес датчика. **a** — текущий адрес, **b** — новый адрес. Ответом является новый адрес.

Синтаксис:

aAb!

aI!

Запрашивает идентификатор датчика. Отклик определяется производителем датчика, но обычно включает адрес датчика, версию SDI-12, имя производителя и сведения о модели датчика. Также может быть включен серийный номер или иная специфичная для датчика информация.

Синтаксис:

aI!

Пример ответа на команду **aI!**:

```
013NRSYSINC1000001.2101 <CR><LF>
```

где:

0 — это адрес SDI-12.
13 — версия SDI-12 (1.3).
NRSYSINC обозначает производителя.
100000 указывает на модель датчика.
1.2 — версия датчика.
101 — серийный номер датчика.

SDI-12: команды начала измерения

Ответы на команды измерения имеют следующий вид:

```
atttnn
```

где:

a — адрес датчика
ttt — время (в секундах) до открытия доступа к данным измерений
nn — количество возвращаемых значений при подаче одной или нескольких **D!**-команд.

aMv!

Запускает стандартное измерение. Квалификатор **v** — это переменная от 1 до 9. Если это поддерживается производителем датчика, **v** запрашивает данные варианта. Варианты могут включать альтернативные единицы (например, °C или °F), дополнительные значения (например, уровень и температуру) или диагностику внутренней батареи датчика.

Синтаксис:

aMv!

Так, например, ответ на команду **5M!** выглядит следующим образом:

```
500410
```

где:

5 — адрес датчика SDI-12.

004 обозначает, что данные будут доступны через 4 секунды.

10 указывает, что будет доступно 10 значений.

Команда **5M7!** вызывает аналогичный ответ, однако дополнительный символ «7» предписывает датчику вернуть значение напряжения внутренней батареи.

aC!

Запуск параллельных процессов измерения. CR1000 запрашивает измерение, продолжает выполнение программы и собирает запрошенные данные при следующем прогоне программы. Затем запрос измерения отправляется повторно, так что данные готовы при следующем сканировании. Частота сканирования регистратора данных должна быть настроена таким образом, чтобы итоговое расхождение времени измерения и времени сбора данных не повлияло на целостность данных. Эта команда впервые появилась в версии SDI-12 1.2.

Синтаксис:

aC!

Прерывание выполнения команды измерения SDI-12

Выполнение команды измерения (**M!** или **C!**) прерывается, когда на датчик подается любая другая действительная команда.

SDI-12: команда отправки данных

Команды отправки данных обычно подаются из CR1000 автоматически после команд измерения **aMv!** или **aCv!**. В прозрачном режиме необходимо передавать терминальные команды CR1000 определенными последовательностями. В автоматическом режиме, если в ответ на команду **aD0!** не возвращается ожидаемое число значений данных, регистратор данных передает **aD1!, aD2!** и т. д., пока не будут получены все данные. В прозрачном режиме следует действовать аналогичным образом. Ограничение заключается в том, что общее число символов, которое может быть возвращено в ответ на команду **aDv!** составляет 35 (75 для **aCv!**). Если число символов превосходит лимит, остаток отклика поступает с последующими командами **aDv!**, причем **v** увеличивается с каждой итерацией.

aDv!

Запрос данных с датчика.

Пример синтаксиса:

aD0!

SDI-12: команды непрерывного измерения (от aR0! до aR9!)

Датчики, постоянно участвующие в процессе мониторинга, например преобразователь углового положения в код, не требуют использования **M**-команды. Их показания могут считываться напрямую с помощью команды непрерывного измерения (от **R0!**до **R9!**). Например, если датчик работает в режиме постоянного измерения, **aR0!** возвращает текущее показание датчика. Ответы на **R**-команды форматируются аналогично ответам на команды отправки данных (**aDv!**). Основное различие заключается в том, что для **R**-команд не требуется предшествующей **M**-команды. Максимальное число символов, возвращаемых в части <значения> ответа, составляет **75.**

Каждая **R**-команда представляет собой независимое измерение. Например, для **aR5!** не требуется наличия предшествующих команд от **aR0!** до **aR4!**. Если датчик не может обеспечить непрерывное измерение, он должен вернуть свой адрес и следующие за ним символы **<CR><LF>** (разрыв строки/перевод строки) в ответ на **R**-команду. В случае запроса CRC символам **<CR><LF>** должно предшествовать значение CRC.

aRv!

Запрос непрерывных данных с датчика.

Пример синтаксиса:

```
aR5!
```

7.9.18.2. SDI-12: режим записи

CR1000 можно запрограммировать на работу в качестве записывающего устройства SDI-12 или датчика SDI-12.

В целях устранения неисправностей ответы на команды SDI-12 могут регистрироваться в программируемом режиме посредством размещения переменной, объявленной как **Строка** в параметрах переменной. Переменные, не объявленные как **Строки**, будут регистрировать только числовые данные.

Еще одним средством устранения неисправностей является отслеживающая служебная программа терминального режима, позволяющая выполнять мониторинг трафика SDI-12. Войдите в терминальный режим, как описано в разделе *SDI-12: прозрачный режим* *(стр. 267)*, введите CRLF (нажмите **<Enter>**) и подождите, пока появится запрос CR1000>. Наберите **W**, затем нажмите **<Enter>.** Напечатайте **9** в ответ на запрос **Выбор:, 100** — в ответ на запрос **Введите время ожидания (с):, Y** в ответ на **ASCII (Y)?.** После этого коммуникации SDI-12 будут открыты для просмотра.

Команда **SDI12Recorder()** автоматизирует подачу команд и интерпретацию ответов датчика. Команды, которые вводятся в команду **SDIRecorder()**, немного отличаются по своей функции от похожих команд, которые вводятся в прозрачном режиме. В прозрачном режиме, например, оператор вручную вводит **aM!** и **aD0!** для запуска измерения и получения данных, причем между запросом измерения и запросом данных предполагается достаточная временная задержка. В программируемом режиме CR1000 команда и расчет времени заложены в единую строку кода. Например, когда команда **SDI12Recorder()** программируется с помощью команды **M!** (обратите внимание, что адрес SDI-12 является отдельным параметром команды), CR1000 передает команды **aM!** *и* **aD0!** с достаточным временным интервалом между ними. CR1000 автоматически передает повторные команды и выполняет другие действия, обеспечивающие максимально стабильный и безопасный процесс измерения SDI-12. В таблице *Команды SDI-12Recorder()* *(стр. 272)* приведены действия CR1000, инициируемые некоторыми из команд **SDI12Recorder()**.

Если выполнение команды **SDI12Recorder()** завершилось неудачей, в качестве первой переменной загрузится значение **NAN**. Более подробную информацию см. в разделе *NAN и ±INF* *(стр. 482)*.

Имя команды	*SDIRecorder()* SDICommand *Аргумент*	*Переданная команда SDI-12* *Отклик датчика*[1] *Отклик CR1000* *Комментарии*
Запрос адреса	**?!**	CR1000: передает команду **a?!**. Только один датчик может быть назначен выводу **C**, настроенному в рамках SDI-12 для отправки ответа на эту команду. Датчики должны поддерживать эту команду.
Смена адреса	**Ab!**	CR1000: передает команду **aAb!**
Параллельные процессы измерения	**Cv!, CCv!**	CR1000: передает команду **aCv!** Датчик: возвращает ответ **atttnn** CR1000: если *ttt* = **0**, передает команду/команды **aDv!**. Если *nnn* = **0**, тогда первым элементом массива становится **NAN**. Датчик: в ответ возвращает данные CR1000: в ином случае, если **ttt** > 0, переходит к следующей программной команде CRBasic CR1000: при следующем выполнении **SDIRecorder()**, если промежуток времени < **ttt**, переходит к следующей команде CRBasic CR1000: в ином случае передает команду/команды **aDv!** Датчик: в ответ возвращает данные CR1000: передает команду **aCv!** (запрос данных для следующего сканирования)
Альтернативные параллельные процессы измерения	**Cv** (примечание — без завершающего «!»)[2]	***CR1000: проверяет ttt на истечение срока действия. Если значение ttt не истекло, загружает 1e9 в качестве первой переменной и переходит к следующей команде CRBasic. Если значение ttt истекло, передает команду/команды aDv!. См. раздел «Команда альтернативного запуска параллельных процессов измерения» (Cv) (стр. 273)*** Датчик: в ответ на команду/команды **aDv!** возвращает данные (при наличии таковых). Если данные отсутствуют, в качестве переменной загружается значение NAN. CR1000: переходит к следующей команде CRBasic (не передает повторно команду **aCv!**)
Отправка идентификатора	**I!**	CR1000: передает команду **aI!**
Начало измерения	**M!, Mv!, MCv!**	CR1000: передает команду **aMv!** Датчик: возвращает ответ **atttnn** CR1000: Если *nnn* = **0**, тогда первым элементом массива становится **NAN**. CR1000: ждет **ttt** [в течение 3] секунд (если не поступает сервисный запрос). Передает команду/команды **aDv!**. При поступлении сервисного запроса немедленно передает команду **aDv!**. Датчик: в ответ возвращает данные
Непрерывные измерения	**Rv!, RCv!**	CR1000: передает команду **aRv!**
Начало проверки	**V!**	CR1000: передает команду **aV!**

[1]В таблице *Команды SDI-12 для прозрачного режима (стр. 269)* полностью приведены ответы датчика.

[2]Воспользуйтесь заменой переменной в программе, чтобы использовать тот же экземпляр **SDI12Recorder()**, что и при передаче команды **aCV!** (см. пример CRBasic *Использование альтернативной параллельной команды (aC) (стр. 277)*).

[3]Обратите внимание, что значение **ttt** присуще только команде **SDIRecorder()**. Если используется вторая команда **SDIRecorder()**, в ней будет собственное значение **ttt.**

Примечание. **aCv** и **aCv!** являются различными командами — **aCv** не имеет завершающего «!».

Команда **SDIRecorder() aCv** упрощает использование стандартной для SDI-12 команды запуска параллельного процесса **(aCv!)** без встречно-параллельной последовательности измерений, нормальной для реализованной в CR1000 команды **aCv!.**

Представим практический случай, когда четыре температурных датчика SDI-12 должны раз в пять минут практически одновременно передавать показания измерений с помощью программы, производящей сканирование каждые пять секунд. Датчикам требуется 95 секунд для передачи данных в ответ на запрос измерения. Дополнительная сложность заключается в требовании минимизации потребления энергии, так что датчики должны отключаться после каждого измерения.

Данный случай является основным аспектом для рассмотрения стратегий нескольких измерений. Простейший способ измерения — подача команды измерения **M!** на каждый датчик, как показано в следующем примере CRBasic:

```
Public BatteryVolt
Public Temp1, Temp2, Temp3, Temp4

BeginProg
  Scan(5,Sec,0,0)

    Измерения вне SDI-12

    SDI12Recorder(Temp1,1,0,"M!",1.0,0)
    SDI12Recorder(Temp2,1,1,"M!",1.0,0)
    SDI12Recorder(Temp3,1,2,"M!",1.0,0)
    SDI12Recorder(Temp4,1,3,"M!",1.0,0)

  NextScan
EndProg
```

Однако в последовательности команд есть три проблемы:

1. Она не позволяет с требуемой частотой производить измерения по датчикам, не относящимся к SDI-12, поскольку выполнение команды **SDI12Recorder()** занимает слишком много времени.
2. При ней не достигается необходимая пятиминутная частота дискретизации, поскольку для выполнения каждой команды **SDI12Recorder()** требуется около 95 секунд, прежде чем запустится следующая команда **SDI12Recorder()**, и в результате реальная частота сканирования составляет примерно 6,5 минут.
3. Между измерениями по каждому из датчиков наблюдается временной сдвиг на 95 секунд.

Проблему 1 можно решить посредством переноса измерений SDI-12 в процесс сканирования **SlowSequence**. Это позволит подпрограмме SDI-12 выполняться без какого-либо воздействия на измерения других датчиков, а именно:

```
Public BatteryVolt
Public Temp(4)

BeginProg

Scan(5,Sec,0,0)
    Измерения вне SDI-12
NextScan

SlowSequence
  Scan(5,Min,0,0)
    SDI12Recorder(Temp(1),1,0,"M!",1.0,0)
    SDI12Recorder(Temp(2),1,1,"M!",1.0,0)
    SDI12Recorder(Temp(3),1,2,"M!",1.0,0)
```

```
    SDI12Recorder(Temp(4),1,3,"M!",1.0,0)
  NextScan
EndSequence

EndProg
```

Однако проблемы 2 и 3 еще не решены. При их решении может помочь использование команды параллельных процессов измерения, **C!.** Все измерения будут производиться примерно в одно и то же время, а время выполнения составит приблизительно 95 секунд, что с большим запасом соответствует требованию к частоте сканирования:

```
Public BatteryVolt
Public Temp(4)

BeginProg

  Scan(5,Sec,0,0)
    Измерения вне SDI-12
  NextScan

  SlowSequence
    Scan(5,Min,0,0)
    SDI12Recorder(Temp(1),1,0,"C!",1.0,0)
    SDI12Recorder(Temp(2),1,1,"C!",1.0,0)
    SDI12Recorder(Temp(3),1,2,"C!",1.0,0)
    SDI12Recorder(Temp(4),1,3,"C!",1.0,0)
  NextScan

EndProg
```

Однако в результате использования команды **C!** возникает новая проблема: высокое энергопотребление со стороны CR1000. Для данного практического случая характерен достаточно ограниченный бюджет в плане питания. Поскольку команда **C!** повторно подает запрос на измерение сразу после получения данных, датчики постоянно требуют значительных энергозатрат. Для решения этой проблемы необходимо начинать измерения с помощью команды **C!**, но останавливать их выполнение непосредственно перед получением следующей команды измерения (жестко запрограммированная часть подпрограммы **C!**), как только обработан запрос данных. Команда **C** команды **SDI12Recorder()** (не **C!**) обладает данным функционалом, как продемонстрировано в примере CRBasic *Использование альтернативной параллельной команды (aC)* *(стр. 277)***.** Также можно использовать модификацию данной программы, позволяющую производить практически одновременные измерения по датчикам SDI-12 без запроса дополнительных процессов измерения, что может потребоваться при измерении на основе событий.

Примечание. Если подключен только один датчик SDI-12, т. е. нет необходимости запускать параллельно несколько измерений по датчикам, может быть применен другой надежный метод выполнения измерений SDI-12 без воздействия на основной процесс сканирования, заключающийся в использовании команды CRBasic **SlowSequence** и команды SDI-12 **M!**. Выполнение основного процесса сканирования будет продолжаться в течение периода времени ***ttt***, возвращенного датчиком SDI-12. Вся хитрость заключается в синхронизации возвращенных значений SDI-12 с основным процессом сканирования.

---

aCv

Запуск альтернативных параллельных процессов измерения.

Синтаксис:

```
aCv
```

CRBasic: пример 55. Использование SDI12Sensor() для тестирования команды Cv

```
Этот пример программы демонстрирует принцип использования CRBasic для моделирования четырех
датчиков SDI-12. Данная
программа может использоваться для
выполнения измерений с целью тестирования примера CRBasic «Использование альтернативной
параллельной команды (aC) (стр. 277).

Public Temp(4)

DataTable(Temp,True,0)
   DataInterval(0,5,Min, 10)
   Sample(4,Temp() , FP2)
EndTable

BeginProg
   Scan(5,Sec,0,0)

     PanelTemp(Temp(1),250) Измерение температуры коммутационной панели CR1000 с целью
                            использования в качестве базы для
                            смоделированных температур Temp(2), Temp(3) и Temp(4).
     Temp(2) = Temp(1) + 5
     Temp(3) = Temp(1) + 10
     Temp(4) = Temp(1) + 15

     CallTable Temp

   NextScan

   SlowSequence
     Do
       Примечание. SDI12SensorSetup / SDI12SensorResponse необходимо обновлять
       после каждого успешного запроса SDI12Recorder().
       SDI12SensorSetup(1,1,0,95)
       Delay(1,95,Sec)
       SDI12SensorResponse(Temp(1))
     Loop
   EndSequence

   SlowSequence
     Do
       SDI12SensorSetup(1,3,1,95)
       Delay(1,95,Sec)
       SDI12SensorResponse(Temp(2))
     Loop
   EndSequence

   SlowSequence
     Do
       SDI12SensorSetup(1,5,2,95)
       Delay(1,95,Sec)
       SDI12SensorResponse(Temp(3))
     Loop
   EndSequence
```

```
  SlowSequence
    Do
      SDI12SensorSetup(1,7,3,95)
      Delay(1,95,Sec)
      SDI12SensorResponse(Temp(4))
    Loop
  EndSequence

EndProg
```

CRBasic: пример 56. Использование альтернативной параллельной команды (aC)

```
'Данный пример программы иллюстрирует использование специальной команды параллельного измерения
'SDI-12 (aC) в тех случаях, когда встречно параллельные измерения нежелательны, что вполне
'обычно для практических случаев с ограниченным бюджетом в плане питания. Для наиболее
'эффективного использования команды aC применяется логика контроля измерений.

'Объявление переменных
Dim X
Public RunSDI12
Public Cmd(4)
Public Temp_Tmp(4)
Public Retry(4)
Public IndDone(4)
Public Temp_Meas(4)
Public GroupDone

'Основная программа
BeginProg

'В качестве первой команды измерения предварительно устанавливается C!
 For X = 1 To 4
   cmd(X) = "C!"
 Next X

 'Установка пятисекундной частоты сканирования.
 Scan(5,Sec,0,0)

   'Прочие измерения

   'Установка пятиминутной частоты измерений SDI-12
   If TimeIntoInterval(0,5,Min) Then RunSDI12 = True

   'Начало последовательности измерений
   If RunSDI12 = True Then

     For X = 1 To 4
       Temp_Tmp(X) = 2e9                          'если 2e9 изменяется, указывает на изменение
     Next X
```

```
  Измерение по датчикам SDI-12
  SDI12Recorder(Temp_Tmp(1),1,0,cmd(1),1.0,0)
  SDI12Recorder(Temp_Tmp(2),1,1,cmd(2),1.0,0)
  SDI12Recorder(Temp_Tmp(3),1,2,cmd(3),1.0,0)
  SDI12Recorder(Temp_Tmp(4),1,3,cmd(4),1.0,0)

  Контрольное событие измерения
  For X = 1 To 4
     If cmd(X) = "C!" Then Retry(X) = Retry(X) + 1
     If Retry(X) > 2 Then IndDone(X) = -1

     Проверка ttt на истечение срока действия. Если значение ttt не истекло, загрузка «1e9» в
     качестве первой переменной,
     затем переход к следующей команде. Если значение ttt истекло, передача команды/команд
     aDv!.
     If ((Temp_Tmp(X) = 2e9) OR (Temp_Tmp(X) = 1e9)) Then
       cmd(X) = "C"                          Начало отправки «C»-команды.

     ElseIf(Temp_Tmp(X) = NAN) Then        Коммуникация завершилась неудачей, либо датчик не
                                           подключен
       cmd(X) = "C!"                       Начало измерения по

     Else 'C!/C command sequence complete
       Move(Temp_Meas(X),1,Temp_Tmp(X),1)  Копирование измерений в SDI_Val(10)
       cmd(X) = "C!"                       Запуск следующего измерения с помощью «C!»
       IndDone(X) = -1
     EndIf
  Next X

    Сводка успешных результатов по событиям измерения
    For X = 1 To 4
      GroupDone = GroupDone + IndDone(X)
  Next X

  Остановка текущего события измерения, сброс управляющих параметров
  If GroupDone = -4 Then
    RunSDI12 = False
    GroupDone = 0
    For X = 1 To 4
     IndDone(X) = 0
     Retry(X) = 0
    Next X
  Else
    GroupDone = 0
  EndIf
 EndIf                                      Окончание последовательности измерений

 NextScan

EndProg
```

SDI12Recorder() передает любую строку, заключенную в кавычки, в параметр ***Команда***. Если строка команды не является стандартной командой SDI-12, ответы записываются в переменные, присвоенные параметру ***Назначение***, до тех пор, пока переменная объявлена как **Строка**. В примере CRBasic *Использование расширенной команды SDI-12* *(стр. 279)* демонстрируется правильный код для отправки расширенной команды SDI- 12 и получения ответа. Функция расширенной команды не обладает встроенным средством для ответа с помощью дополнительных команд. Тем не менее в программе можно закодировать способность разбирать ответь и передавать последующие команды SDI-12 на основе пользовательского анализа отклика. Более подробная информация о разборе строк приведена в разделе *Основы программирования на входе* *(стр. 251)*.

CRBasic: пример 57. Использование расширенной команды SDI-12

```
'В этом примере программы демонстрируется использование расширенных команд SDI-12. В данном
'примере измерение температуры, tt.tt, передается в регулятор зарядки CH200 с помощью команды
'XTtt.tt!'. Ответом от CH200 должно быть значение 'OOK', если адрес SDI-12 равен 0.
'
'Объявление переменных
Public PTemp As Float
Public SDI12command As String
Public SDI12result As String

'Основная программа
BeginProg
   Scan(20,Sec,3,0)
     PanelTemp(PTemp,250)
     SDI12command = "XT" & FormatFloat(PTemp,"%4.2f") & "!"
     SDI12Recorder(SDI12result,1,0,SDI12command,1.0,0)
   NextScan
EndProg
```

7.9.18.3. SDI-12: режим датчика

CR1000 можно запрограммировать на работу в качестве записывающего устройства SDI-12 или датчика SDI-12.

В целях устранения неисправностей ответы на команды SDI-12 могут регистрироваться в программируемом режиме посредством размещения переменной, объявленной как **Строка** в параметрах переменной. Переменные, не объявленные как **Строки**, будут регистрировать только числовые данные.

Еще одним средством устранения неисправностей является отслеживающая служебная программа терминального режима, позволяющая выполнять мониторинг трафика SDI-12. Войдите в терминальный режим, как описано в разделе *SDI-12: прозрачный режим* *(стр. 267)*, введите CRLF (нажмите **<Enter>**) и подождите, пока появится запрос CR1000>. Наберите **W**, затем нажмите **<Enter>.** Напечатайте **9** в ответ на запрос **Выбор:, 100** — в ответ на запрос **Введите время ожидания (c):, Y в ответ на ASCII (Y)?**. После этого коммуникации SDI-12 будут открыты для просмотра.

С помощью пары команд **SDI12SensorSetup() / SDI12SensorResponse()** программируется поведение CR1000 в качестве датчика SDI-12. Практическое применение этой функции заключается в переносе данных из CR1000 в другие регистраторы данных Campbell Scientific с помощью однопроводного интерфейса (от настроенного под SDI-12 вывода к настроенному под SDI-12 выводу) либо в переносе данных в постороннее устройство записи SDI-12.

Подробная информация об использовании пары команд **SDI12SensorSetup() / SDI12SensorResponse()** приведена в *Справке по редактору CRBasic*. Прочие полезные советы:

Что касается параметра ***Reps*** в **SDI12SensorSetup(),** действительное значение ***Reps*** при ожидании команды **aMx!** варьируется в диапазоне от 0 до 9. Действительные значения ***Reps*** при ожидании команды **aCx!**: от 0 до 20. Параметр *Reps* не проходит проверку по диапазону на предмет действительных записей на момент компиляции. Когда в записывающее устройство SDI-12 поступает значение **atttn** в ответ на команду **aMx!** или **atttnn** в ответ на команду **aCx!**, используется только первый знак **n** или первые два знака **nn**. Например, если для ***Reps*** ошибочно запрограммировано значение 123, в записывающее устройство SDI-12 поступит только ответ **n** = 1 при подаче команды **aMx!**, либо ответ **nn** = 12 при подаче команды **aCx!**.

Когда устройство CR1000 запрограммировано как датчик SDI-12, будут отправляться ответы на команды SDI-12 **M, MC, C, CC, R, RC, V, ?** и **I**. В таблице *Команды SDI-12 для прозрачного режима* *(стр. 269)* представлен полный синтаксис команд. Применяются следующие правила:

1. Для CR1000 может быть присвоен только один адрес SDI-12 на каждый порт SDI-12. Например, CR1000 не будет одновременно отвечать на **0M!** И **1M!** через порт SDI-12 **C1**. Разные порты SDI-12, тем не менее, могут иметь уникальные адреса SDI-12. Рекомендуется использовать отдельную команду **SlowSequence** для каждого порта SDI-12, настроенного как датчик.
2. CR1000 обрабатывает дополнительные команды измерения (**aMx!**). Если записывающее устройство SDI-12 передает команды **aMx!**, как продемонстрировано в примере CRBasic *Настройка датчика SDI-12* *(стр. 280)*, результаты измерений возвращаются в соответствии со списком из таблицы *Настройка датчика SDI-12 — результаты* *(стр. 280)*.

CRBasic: пример 58. Настройка датчика SDI-12

```
'В этом примере программы демонстрируется использование пары команд SDI12SensorSetup()/
'SDI12SensorResponse() для программирования CR1000 в качестве эмулятора датчика SDI-12.
'Практическое применение этой функции заключается в переносе данных из CR1000 в совместимые с
'SDI-12 измерительные приборы, включая другие регистраторы данных Campbell Scientific, с помощью
'однопроводного интерфейса (из порт SDI-12 в порт SDI-12). Записывающий регистратор данных просто
'запрашивает данные с помощью команды aD0!.

Public PanelTemp
Public Batt_volt
Public SDI_Source(10)

BeginProg
   Scan(5,Sec,0,0)

    PanelTemp(PanelTemp,250)
    Battery(batt_volt)

    SDI_Source(1) = PanelTemp               'температура, °C
    SDI_Source(2) = batt_volt               'первичное питание, В пост. тока
    SDI_Source(3) = PanelTemp * 1.8 + 32    'температура, °F
    SDI_Source(4) = batt_volt               'первичное питание, В пост. тока
    SDI_Source(5) = PanelTemp               'температура, °C
    SDI_Source(6) = batt_volt * 1000        'первичное питание, мВ пост. тока
    SDI_Source(7) = PanelTemp * 1.8 + 32    'температура, °F
    SDI_Source(8) = batt_volt * 1000        'первичное питание, мВ пост. тока
    SDI_Source(9) = Status.SerialNumber     'серийный номер
    SDI_Source(10) = Status.LithiumBattery  'батарея резервного буфера данных, В

NextScan

   SlowSequence

    Do
      SDI12SensorSetup(10,1,0,1)
      Delay(1,500,mSec)
      SDI12SensorResponse(SDI_Source)
    Loop

   EndSequence
EndProg
```

Таблица 44. Пример CRBasic: настройка датчика SDI-12 — результаты

Команда измерения от записывающего устройства SDI-12	***Исходные переменные, доступные из устройства CR1000, функционирующего в качестве датчика SDI-12***	***Суть исходных переменных***
0M!	***Источник(1), Источник(2)***	Температура в °C, напряжение батареи
0M0!	Аналогично ***0M!***	
0M1!	***Источник(3), Источник(4)***	Температура в °F, напряжение батареи
0M2!	***Источник(5), Источник(6)***	Температура в °C, батарея (мВ)
0M3!	***Источник(7), Источник(8)***	Температура в °F, батарея (мВ)
0M4!	***Источник(9), Источник(10)***	Серийный номер, напряжение литиевой батареи

7.9.18.4. SDI-12: аспекты питания

Когда CR1000 передает команду на измерительный прибор SDI-12, активируются все измерительные приборы, подключенные к тому же порту SDI-12. Тем не менее ответ поступает только от измерительного прибора, к которому обращается регистратор данных. Все остальные измерительные приборы останутся в активном состоянии до истечения времени ожидания.

Пример:

Измерительный прибор: влажность

Потребление энергии:

- В состоянии ожидания: 0,25 мА
- Измерение: 120 мА
- Продолжительность измерения: 15 с
- В работе: 66 мА
- Время ожидания: 15 с

Измерительные приборы 1, 2, 3 и 4 подключаются к SDI-12 / управляющему порту **C1**.

С помощью строки времени в таблице *Пример профиля энергопотребления для сети измерительных приборов SDI-12* *(стр. 281)* проиллюстрирован пример 35-секундного профиля энергопотребления.

В большинстве случаев общее энергопотребление, составляющее 318 мА за 15 секунд, не является завышенным, однако при подключении 16 измерительных приборов к одному и тому же порту SDI-12 суммарная передаваемая мощность окажется избыточной. Распределение датчиков по нескольким выводам SDI-12 помогает снизить энергопотребление.

Таблица 45. Пример профиля энергопотребления для сети измерительных приборов SDI-12

Время (с)	Команда	Все измерительные приборы активны	Истечение времени ожидания	1 мА	2 мА	3 мА	4 мА	Всего мА
1	***1M!***	Да		120	66	66	66	318
2				120	66	66	66	318
•				•	•	•	•	•
•				•	•	•	•	•
•				•	•	•	•	•
14				120	66	66	66	318
15			Да	120	66	66	66	318
16	***1D0!***	Да		66	66	66	66	264
17				66	66	66	66	264
•				•	•	•	•	•
•				•	•	•	•	•
•				•	•	•	•	•
29				66	66	66	66	264
30			Да	66	66	66	66	264
31				0,25	0,25	0,25	0,25	1
•				•	•	•	•	
•				•	•	•	•	
•				•	•	•	•	
35				0,25	0,25	0,25	0,25	1

7.9.19. Строковые операции

Строковые операции выполняются с помощью строковых функций CRBasic, перечисленных в разделе *Строковые функции* *(стр. 574)*.

7.9.19.1. Строковые операторы

В таблице *Строковые операторы* *(стр. 282)* перечислены и описаны доступные строковые операторы. Строковые операторы чувствительны к регистру.

Таблица 46. Строковые операторы

Оператор	Описание
&	Выполняет конкатенацию строк. Принудительно преобразует числовые значения в строки перед конкатенацией. Пример 1 & 2 & 3 & «a» & 5 & 6 & 7 = «123a567»
+	Складывает числовые значения, пока не встретится строка. Как только встречается строка, она прибавляется к сумме числовых значений. Последующие числовые значения добавляются в качестве строк. Пример: 1 + 2 + 3 + «a» + 5 + 6 + 7 = «6a567»
-	«Отнимает» нулевое значение («») из конца ASCII-символа для преобразования в ASCII-код (тип данных «LONG»). Пример: «a» - «» = 97 Для ASCII-кодов первых символов в каждой строке выполняется сравнение. Если разница между кодами оказывается нулевой, сравниваются коды следующих символов. Если обнаруживается несовпадение кодов или нулевой символ (нулевой символ завершает все строки), возвращается разница между последними сравненными ASCII-кодами. Например: **Примечание.** ASCII-код для a = 97, b = 98, c = 99, d = 100, e = 101, и все строки оканчиваются нулевым символом. Разница между нулевыми символами (**NULL** и **NULL)** «abc» - «abc» = 0 Разница между **e** и **c** «abe» - «abc» = 2 Разница между **c** и **b** «ace» - «abe» = 1 Разница между **d** и **NULL** «abcd» - «abc» = 100
<, >, <>, <=, >=, =	Для ASCII-кодов первых символов в каждой строке выполняется сравнение. Если разница между кодами оказывается нулевой, сравниваются коды следующих символов. Если обнаруживается несовпадение кодов или нулевой символ (нулевой символ завершает все строки), выполняется затребованное сравнение. Если сравнение истинно, возвращается значение **–1** или **True** («истина»). Если ложно, возвращается значение **0** или **False** («ложь»). Например:

Выражение	Result
x = «abc» = «abc»	x = –1 или True
x = «abe» = «abc»	x = 0 или False
x = «ace» > «abe»	x = –1 или True

7.9.19.2. Конкатенация строк

Под конкатенацией подразумевается построение строк из других строк («abc123»), символов («a» или **chr()**), чисел или переменных. В таблице *Примеры конкатенации строк* *(стр. 284)* перечислены некоторые выражения и ожидаемые результаты. Пример CRBasic *Конкатенация чисел и строк* *(стр. 284)* иллюстрирует несколько вариантов конкатенации.

Когда при конкатенации нестроковых значений со строками встречается строка, все последующие операнды сначала преобразуются в строки, прежде чем будет выполнена операция сложения («+»). При работе со строками исключительное использование оператора & гарантирует, что ни одно значение строки не будет преобразовано в целое число.

Таблица 47. Примеры конкатенации строк

Выражение	*Комментарии*	*Result*
`Str(1) = 5.4 + 3 + " В"`	Сложение плавающих значений, конкатенация строк	`"8.4 В"`
`Str(2) = 5.4 & 3 & " В"`	Конкатенация плавающих значений и строк	`"5.43 В"`
`Lng(1) = "123"`	Преобразование строки в значение «long»	`123`
`Lng(2) = 1+2+"3"`	Добавление плавающих значений к строке / преобразование в значение «long»	`33`
`Lng(3) = "1"+2+3`	Конкатенация строки и плавающих значений	`123`
`Lng(4) = 1&2&"3"`	Конкатенация плавающих значений и строки	`123`

CRBasic: пример 59. Конкатенация чисел и строк

```
В этом примере программы демонстрируется конкатенация чисел и строк в переменные,
объявленные как Строки и Плавающие значения.
'
Объявление переменных
Public Num(12) As Float
Public Str(2) As String
Dim I

BeginProg
  Scan(1,Sec,0,0)

    I = 0 Присвоение нуля для I

    Тип данных следующих целевых переменных — «Плавающее значение»,
    поскольку массив Num() объявлен как плавающее значение.
    I += 1 Увеличение I на 1 для синхронизации последующих элементов массива Num()

    Как показано в следующем выражении, если все параметры являются числами, результатом
    использования «+» становится сумма чисел:
    Num(I) = 2 + 3 + 4                          '= 9

    Ниже приведены примеры использования «+» и «*», когда один или несколько параметров являются
    строками.
    Параметры обрабатываются в соответствии со стандартным порядком операций. При таком порядке
    операций, как только обрабатывается строка или «&», все последующие параметры
    будут обработаны (конкатенированы) как строки:
    I += 1
    Num(I) = "1" + 2 + 3 + 4                    '= 1234
    I += 1
    Num(I) = 1 + "2" + 3 + 4                    '= 1234
    I += 1
    Num(I) = 1 + 2 + "3" + 4                    '= 334
    I += 1
    Num(I) = 1 + 2 + 3 + "4"                    '= 64
```

```
    I += 1
    Num(I) = 1 + 2 + "3" + 4 + 5 + "6"       '= 33456
    I += 1
    Num(I) = 1 + 2 + "3" + (4 + 5) + "6"     ' = 3396
    I += 1
    Num(I) = 1 + 2 + "3" + 4 * 5 + "6"       '= 33206
    I += 1
    Num(I) = 1 & 2 + 3 + 4                   ' = 1234
    I += 1
    Num(I) = 1 + 2 + 3 & 4                   '= 64

    Если предпринимается попытка обработки нечисловой строки в плавающее целевое значение,
    операции прерываются в этот момент
    I += 1
    Num(I) = 1 + 2 + «hey» + 4 + 5 + «6»     '= 3
    I += 1
    Num(I) = 1 + 2 + «hey» + (4 + 5) + «6»   '= 3

    Аналогичные правила применяются, когда целевое значение принадлежит к типу данных «Строка»,
    за исключением
    случая, когда встречается нечисловая строка, как показано ниже. Тип данных
    следующих целевых переменных — «Строка», поскольку массив Str() объявлен как Строка.
    I = 0

    I += 1
    Str(I) = 1 + 2 + "hey" + 4 + 5 + "6"     = 3hey456
    I += 1
    Str(I) = 1 + 2 + "hey" + (4 + 5) + "6"   = 3hey96

 NextScan
EndProg
```

7.9.19.3. Нулевой символ в строке

Все строки автоматически завершаются нулевым символом. Нулевой символ аналогичен символу **Chr(0)** или считается одним из символов в строке. Присвоение одиночного символа подразумевает, что за этим символом следует нулевой, если только сам символ не является нулевым.

Таблица 48. Примеры нулевого символа в строке

Выражение	*Комментарии*	*Result*
`LongVar(5) = «#»-«»`	Вычитание нулевого символа, ASCII-код в результате	35
`LongVar(6) = StrComp(«#»,«»)`	Также вычитание нулевого символа	35

Пример:

Цель:

Вставка нулевого символа в строку, последующее преобразование строки.

Дано:

```
StringVar(3) = "123456789"
```

Выполнение:

```
StringVar(3,1,4) = ""                "123<NULL>56789"
```

Результат:

```
StringVar(4) = StringVar(3)          "123"
```

однако

```
StringVar(3) still = "123<NULL>56789",
```

таким образом,

```
StringVar(5) = StringVar(3,1,4+1)
'"56789"
StringVar(6) = StringVar(3) + 4 + StringVar(3,1,4+1)
'"123456789"
```

Некоторые интеллектуальные датчики отправляют строки, содержащие нулевые символы. Для управления строкой, содержащей нулевые символы (помимо завершающего), используется команда **MoveBytes()**.

7.9.19.4. Вставка строковых символов

Пример:

Цель:

Использование **MoveBytes()** для изменения **«123456789»** на **«123A56789»**

Дано:

```
StringVar(7) = "123456789"                          Результат:
"123456789"
```

попробуйте (не получится):

```
StringVar(7,1,4) = "A"                              Результат:
«123A<NULL>56789»
```

Вместо этого используйте:

```
StringVar(7) = MoveBytes(Strings(7,1,4),0,"A",0,1)  Результат:
«123A56789»
```

7.9.19.5. Извлечение строковых символов

Доступ к специальным символам в строке осуществляется с помощью «синтаксиса измерений», т.е., если указывается третье измерение строки, оно определяет положение символа.

Таблица 49. Извлечение строковых символов

Выражение	*Комментарии*	*Result*
`StringVar(3) = "Go Jazz"`	Загружает строку в переменную	`StringVar(3) = "Go Jazz"`
`StringVar(4) = StringVar(3,1,4)`	Извлекает отдельный символ	`StringVar(4) = "J"`

7.9.19.6. Использование ASCII- / ANSII-кодов в строке

Таблица 50. Примеры использования ASCII- / ANSII-кодов		
Выражение	***Комментарии***	***Result***
`LongVar (7) = ASCII("#")`		35
`LongVar (8) = ASCII("*")`		42
`LongVar (9) = "#"`	Невозможно преобразовать в значение «Long» с использованием нулевого символа	NAN
`LongVar (1) = "#"-""`	Можно преобразовать в значение «Long» без нулевого символа	35

7.9.19.7. Форматирование строк

Таблица 51 Примеры форматирования строк	
Выражение	***Result***
`Str(1)=123e4`	`1230000`
`Str(2)=FormatFloat(123e4,"%12.2f")`	`1230000.00`
`Str(3)=FormatFloat(Va1ues(2)," The battery is %.3g Volts ")`	`“The battery is 12.4 Volts”`
`Str(4)=Strings(3,1,InStr(1,Strings(3),"The battery is ",4))`	`12.4 Volts`
`Str(5)=Strings(3,1,InStr(1,Strings(3),"is ",2) + 3)`	`12.4 Volts`
`Str(6)=Rep1ace("The battery is 12.4 Volts"," is "," = ")`	`The battery = 12.4 Volts`
`Str(7)=LTrim("The battery is 12.4 Volts")`	`The battery is 12.4 Volts`
`Str(8)=RTrim("The battery is 12.4 Volts")`	`The battery is 12.4 Volts`
`Str(9)=Trim("The battery is 12.4 Volts")`	`The battery is 12.4 Volts`
`Str(10)=UpperCase("The battery is 12.4 Volts")`	`THE BATTERY IS 12.4 VOLTS`
`Str(12)=Left("The battery is 12.4 Volts",5)`	`The b`
`Str(13)=Right("The battery is 12.4 Volts",7)`	`Volts`

CRBasic: пример 60. Форматирование строк

```
В этом примере программы демонстрируется форматирование строковых переменных. Для запуска
демонстрации следует отправить эту программу в CR1000. Форматирование строк будет выполнено
автоматически.

Цель:
Извлечение фрагмента «12.4 В» из строки «Батарея — 12.4 В»

Public StringVar As String

BeginProg
   Обратите внимание на символ продолжения строки «_»
   StringVar() = Mid("The battery is 12.4 Volts", _
   InStr(1,"The battery is 12.4 Volts"," is ",2)+3,Len("The battery is 12.4 Volts"))
EndProg
```

7.9.19.8. Форматирование строки с шестнадцатеричными переменными

Таблица 52. Форматирование шестнадцатеричных переменных — примеры		
Выражение	***Комментарий***	***Result***
`CRLFNumeric(1) = &H0d0a`	Добавление начального нуля к шестнадцатеричному значению, шаг 1	3338
`StringVar(20) = "0" & Hex(CRLFNumeric)`	Добавление начального нуля к шестнадцатеричному значению, шаг 2	0D0A
`CRLFNumeric(2) = HexToDec(Strings(20))`	Превращение шестнадцатеричной строки в плавающее значение	3338.00

7.9.20. Подпрограммы

Подпрограмма представляет собой группу программных команд, вызываемую в рамках основной программы, но выполняемую вне ее. Подпрограммы используются для следующих целей:

- Сокращение длины программы. Код подпрограммы может выполняться несколько раз за прогон программы.
- Облегчение интеграции проверенных сегментов кода в новые программы.
- Сортировка программ по разделам для оптимизации организационной структуры.

С помощью команды **Call()** основная программа может вызвать подпрограмму из любого места программы.

Подпрограмма обладает доступом ко всем *глобальным переменным* *(стр. 517)***.** *Локальные переменные* *(стр. 519)* подпрограммы объявляются в рамках команды подпрограммы. Локальные переменные могут иметь псевдоним (согласно OS 26, 4/2013), однако не отображаются в таблице **Public.** Глобальные и локальные переменные могут иметь одно и то же имя, при этом не конфликтуя. Если глобальные переменные передаются в локальные переменные другого типа, применяются те же правила преобразования, что и к переменным, объявленным как **Public** или **Dim**. Типы преобразования описаны в разделе *Выражения с числовыми типами данных* *(стр. 162)***.**

Примечание. Во избежание программных конфликтов следует передавать информацию в локальные переменные и/или определять несколько глобальных переменных и использовать их исключительно в рамках подпрограммы.

В примере CRBasic *Подпрограмма с глобальными и локальными переменными* *(стр. 288)* демонстрируется использование глобальных и локальных переменных. Переменные **counter()** и ***pi_product*** являются глобальными. Переменная ***i_sub*** является глобальной, однако используется исключительно подпрограммой обработки (***process***). Переменные **j()** и ***OutVar*** являются локальными, поскольку они объявлены как параметры в команде **Sub(),**

```
Sub process(j(4) AS Long,OutVar).
```

Переменная **j()** представляет собой массив из четырех элементов, переменная **OutVar** — из одного элемента. Оператор вызова,

```
Call ProcessSub (counter(1),pi_product)
```

передает пять значений в подпрограмму: **pi_product** и четыре элемента массива **counter().** Массив **counter()** используется для передачи значений в подпрограмму и их извлечения из подпрограммы. Переменная **pi_product** используется для извлечения значения из подпрограммы.

Call() передает значения всех перечисленных переменных в подпрограмму. Значения передаются обратно в основной процесс сканирования по окончании подпрограммы.

CRBasic: пример 61. Подпрограмма с глобальными и локальными переменными

```
'В этом примере программы демонстрируется использование глобальных и локальных переменных в
'подпрограммах.

'Глобальными являются переменные, объявленные в любом месте подпрограммы как «Public» или «Dim».
'Локальные — это те переменные, которые объявлены в рамках команды Sub().

'Функция программы: Передача двух переменных в подпрограмму. Подпрограмма увеличивает каждую
'переменную раз в секунду, умножает на пи, а затем передает результаты обратно в основную
'программу с целью сохранения в таблице данных.

'Глобальные переменные (используются только вне подпрограммы по желанию)
'Объявление счетчика в основном процессе сканирования.
Public counter(2) As Long

'Объявление произведения ПИ * counter(2).
Public pi_product(2) As Float

'Глобальная переменная (используется только в подпрограмме по желанию)
'Инкрементор «For / Next», используемый в подпрограмме.
Public i_sub As Long

'Объявление таблицы данных
DataTable(pi_results,True, -1)
  Sample(1,counter(),IEEE4)
EndTable

'Объявление подпрограммы
'Объявление j(4) как локального массива (может использоваться только в подпрограмме)
Sub ProcessSub (j(2) As Long,OutVar(2) As Float)
  For i_sub = 1 To 2
     j(i_sub) = j(i_sub) + 1
     'Обработка для демонстрации функциональности
     OutVar(i_sub) = j(i_sub) * 4 * ATN(1)     'Совет: 4 * ATN(1) = пи с точностью IEEE4)
  Next i_sub
EndSub

BeginProg
  counter(1) = 1
  counter(2) = 2
  Scan(1,Sec,0,0)

  'Передача массива Counter() в массив j(), pi_product() в OutVar()
     Call ProcessSub (counter(),pi_product())
     CallTable pi_results

NextScan
EndProg
```

7.9.21. TCP/IP — подробная информация

Связанные темы:

- *TCP/IP — общие сведения* *(стр. 91)*
- *TCP/IP — подробная информация* *(стр. 423)*
- *TCP/IP — команды* *(стр. 593)*
- *Подключения TCP/IP — список* *(стр. 652)*

Следующие протоколы TCP/IP поддерживаются в рамках CR1000 при использовании *сетевых подключений* (стр. 652) с постоянным стеком IP или при использовании сотового модема с активированным ключом PPP/IP. Более подробная информация о некоторых из этих протоколов приведена в следующих разделах.

- DHCP
- DNS
- FTP
- HTML
- HTTP
- микро-последовательный сервер;
- NTCIP;
- NTP;
- PakBus через TCP/IP;
- Ping;
- POP3;
- SMTP
- SNMP;
- Telnet;
- *Web API* (стр. 423)
- XML

Наиболее актуальная информация о реализации этих протоколов содержится в *Справке по редактору CRBasic*. Список команд CRBasic можно найти в приложении *TCP/IP* (стр. 593).

Читать далее. Специальная информация касательно использования цифровых сотовых модемов для TCP/IP находится в руководствах Campbell Scientific по данным модемам. Информацию о доступных устройствах TCP/IP/PPP, в том числе номера моделей, см. в приложении *Сетевые подключения* (стр. 652). Подробную информацию об использовании устройств TCP/IP/PPP можно найти в соответствующих руководствах (доступны по адресу *www.campbellsci.com http://www.campbellsci.com)* и в *Справке по редактору CRBasic*.

7.9.21.1. PakBus через TCP/IP и обратный вызов

После конфигурации аппаратного обеспечения возможна коммуникация PakBus® по TCP/IP. Среди этих функций:

- Отправка программ
- Вызов программ
- Установка времени CR1000
- Сбор данных
- Отображение текущей записи в таблице данных

Обратный вызов данных и связь между регистраторами данных также возможны по TCP/IP. Подробная информация и примеры программ для обратного вызова и связи между регистраторами данных доступны в руководстве по сетевым подключениям. Список номеров моделей для сетевых подключений находится в приложении Сетевые подключения (стр. 652).

7.9.21.2. Веб-сервер HTTP по умолчанию

В CR1000 по умолчанию есть домашняя страница, встроенная в операционную систему. Доступ к домашней странице осуществляется по следующему адресу:

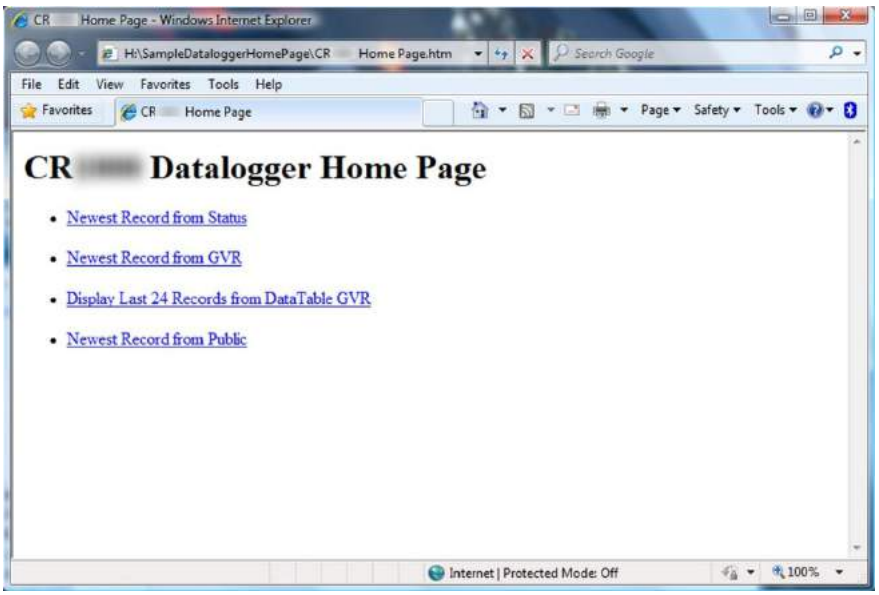
```
http :\\ipaddress : 80
```

Примечание. Порт 80 подразумевается, если нет конкретного указания порта.

Как показано на рисунке *Предварительно сконфигурированная домашняя страница HTML* *(стр. 291)*, эта страница предоставляет ссылки на самые новые записи во всех таблицах, включая таблицу состояния (**Status**), общую таблицу (**Public**) и таблицы данных. Также доступны ссылки на последние 24 записи в каждой из таблиц данных. Если в таблице данных хранится менее 24 записей, по ссылке будут отображены все данные из таблицы.

Ссылки на новые записи автоматически обновляются каждые 10 секунд. Ссылку **Последние 24 записи** необходимо обновлять вручную, чтобы отобразить новые данные. Также автоматически создаются ссылки на любые файлы HTML, XML и JPEG, найденные на дисках CR1000. Чтобы скопировать файлы на эти диски, выберите **Управление файлами** в меню *ПО поддержки регистратора данных* *(стр. 512)*.

Рисунок 73. Предварительно сконфигурированная домашняя страница HTML



7.9.21.3. Пользовательский веб-сервер HTTP

Несмотря на то, что домашняя страница по умолчанию недоступна для редактирования, ее можно заменить HTML-кодом настроенной пользователем веб-страницы. Чтобы заменить домашнюю страницу по умолчанию, сохраните новую домашнюю страницу под именем *default.html* и скопируйте ее в регистратор данных. Ее можно скопировать на диск CR1000 через **Управление файлами**. При удалении *default.html* CR1000 будет использовать изначальную домашнюю страницу, заданную по умолчанию.

CR1000 можно запрограммировать на генерацию кода HTML или XML, который может просматриваться с помощью веб-браузера. В примере CRBasic *HTML* *(стр. 293)* демонстрируется использование команд CRBasic **WebPageBegin() / WebPageEnd** и **HTTPOut()** для создания кода HTML. Обратите внимание, что для HTML-кода, требующего применения кавычек, используется **CHR(34)**, в то время как обычные кавычки определяют начало и окончание буквенно-числовых строк внутри скобок в команде **HTTPOut()**.

Дополнительная информация приведена в *Справке по редактору CRBasic.*

В данном примере программы домашняя страница по умолчанию заменяется с помощью **WebPageBegin** путем создания файла с именем default.html. Внешний вид новой домашней страницы по умолчанию, созданной в программе, показан на рисунке *Домашняя страница, созданная с помощью команды WebPageBegin()* *(стр. 292)*.

Логотип Campbell Scientific на веб-странице находится в файле **SHIELDWEB2.JPG**, который необходимо передать с ПК в CR1000 b сохранит на диск CPU: посредством **Управления файлами** в ПО поддержки регистратора данных.

Вторая веб-страница, показанная на рисунке *Пользовательская веб-страница числового отслеживания* *(стр. 293)* и названная «monitor.html», создана в примере программы, содержащей ссылки на таблицы данных CR1000.

*Рисунок 74. Домашняя страница, созданная с помощью команды **WebPageBegin()***

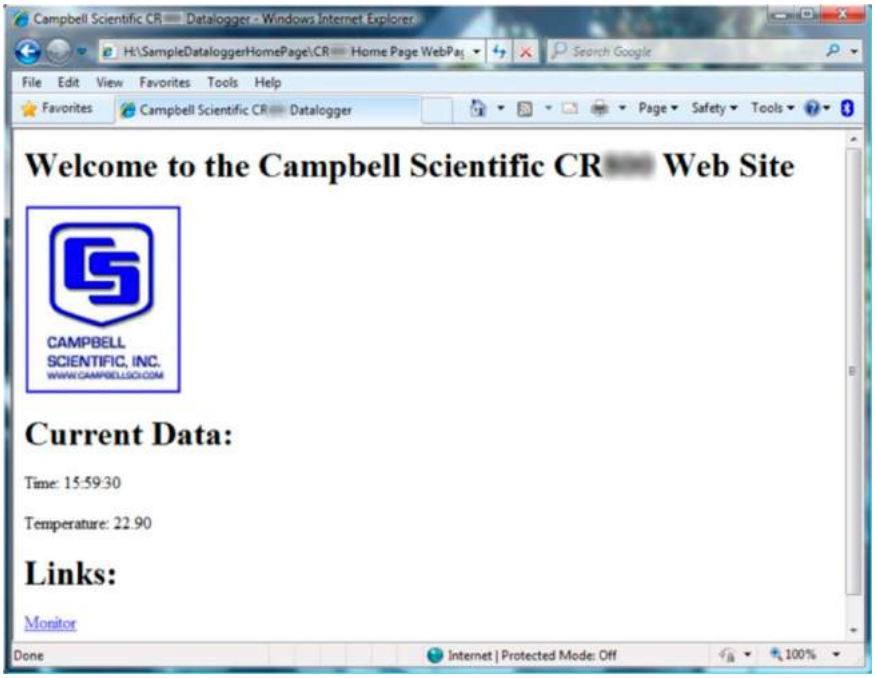
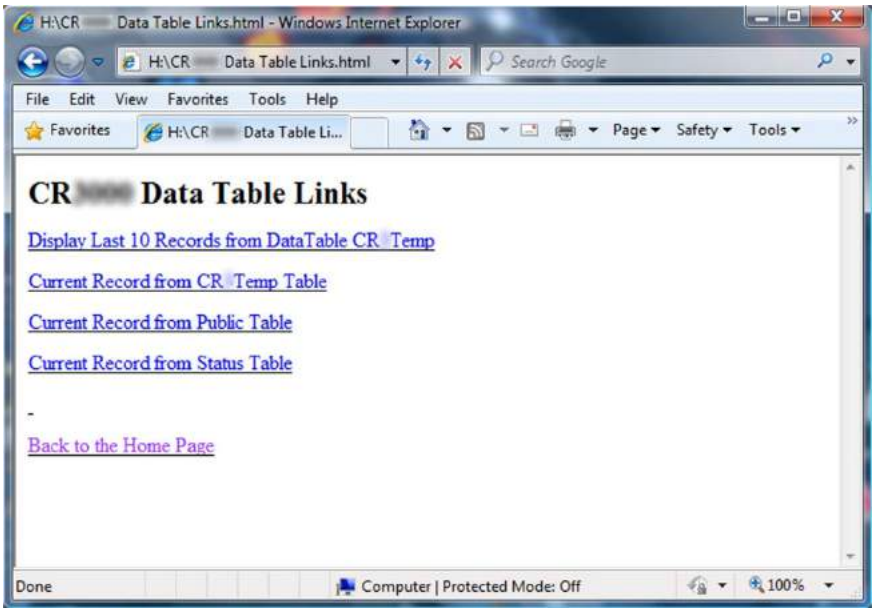


Рисунок 75. Пользовательская веб-страница числового отслеживания



CRBasic: пример 62. Пользовательская веб-страница HTML

```
Данный пример программы иллюстрирует создание пользовательской веб-страницы, находящейся в
CR1000. В этом примере программы стандартная домашняя страница заменяется с помощью WebPageBegin
путем создания файла с именем default.html. Изображение на веб-странице (в данном случае это
логотип Campbell Scientific) находится в файле SHIELDWEB2.JPG. Файл изображения должен быть
скопирован на диск CPU: устройства CR1000 с помощью функции Управления файлами в ПО поддержки
регистратора данных. Затем создается вторая веб-страница, содержащая ссылки на таблицы данных
CR1000.

ПРИМЕЧАНИЕ. Символ «_», используемый в конце некоторых строк, позволяет перенести кодовый
оператор на следующую строку.

Dim Commands As String * 200
Public Time(9), RefTemp,
Public Minutes As String, Seconds As String, Temperature As String

DataTable(CRTemp,True,-1)
   DataInterval(0,1,Min,10)
   Sample(1,RefTemp,FP2)
   Average(1,RefTemp,FP2,False)
EndTable

HTML-страница по умолчанию
WebPageBegin("default.html",Commands)
   HTTPOut("<html>")
   HTTPOut("<style>body {background-color: oldlace}</style>")
   HTTPOut("<body><title>Campbell Scientific CR1000 Datalogger</title>")
   HTTPOut("<h2>Welcome To the Campbell Scientific CR1000 Web Site!</h2>")
   HTTPOut("<tr><td style=" + CHR(34) +"width: 290px" + CHR(34) + ">")
   HTTPOut("<a href=" + CHR(34) + "http://www.campbellsci.com" + CHR(34) + ">")
   HTTPOut("<img src="+ CHR(34) +"/CPU/SHIELDWEB2.jpg"+ CHR(34) + "width=" + _
      CHR(34) +"128"+CHR(34)+"height="+CHR(34)+"155"+ CHR(34) + "class=" + _
      CHR(34) +"style1"+ CHR(34) +"/></a></td>")
   HTTPOut("<p><h2> Current Data:</h2></p>")
   HTTPOut("<p>Time: " + time(4) + ":" + minutes + ":" + seconds + "</p>")
```

```
   HTTPOut(”<p>Temperature: ” + Temperature + "</p>")
   HTTPOut(”<p><h2> Links:</h2></p>")
   HTTPOut("<p><a href="+ CHR(34) +"monitor.htm"l"+ CHR(34) + ">Monitor</a></p>")
   HTTPOut("</body>")
   HTTPOut("</html>")
WebPageEnd

'Веб-страница отслеживания
WebPageBegin("monitor.html",Commands)
   HTTPOut("<html>")
   HTTPOut("<style>body {background-color: oldlace}</style>")
   HTTPOut("<body>")
   HTTPOut("<title>Monitor CR1000 Datalogger Tables</title>")
   HTTPOut("<p><h2>CR1000 Data Table Links</h2></p>")
   HTTPOut("<p><a href="+ CHR(34) + "command=TableDisplay&table=CRTemp&records=10" + _
      CHR(34)+">Display Last 10 Records from DataTable CR1Temp</a></p>")
   HTTPOut("<p><a href="+ CHR(34) + "command=NewestRecord&table=CRTemp"+ CHR(34) + _
      ">Current Record from CRTemp Table</a></p>")
   HTTPOut("<p><a href="+ CHR(34) + "command=NewestRecord&table=Public"+ CHR(34) + _
      ">Current Record from Public Table</a></p>")
   HTTPOut("<p><a href="+ CHR(34) + "command=NewestRecord&table=Status" + CHR(34) + _
      ">Current Record from Status Table</a></p>")
   HTTPOut("<br><p><a href="+ CHR(34) +"default.html"+ CHR(34) + ">Back to the Home Page _
      </a></p>")
   HTTPOut("</body>")
   HTTPOut("</html>")
WebPageEnd

BeginProg
   Scan(1,Sec,3,0)
      PanelTemp(RefTemp,250)
      RealTime (Time())
      Minutes = FormatFloat(Time(5),"%02 .0f")
      Seconds = FormatFloat(Time(6),"%02.0f")
      Temperature = FormatFloat(RefTemp, "%02.02f")
      CallTable(CRTemp)
   NextScan
EndProg
```

7.9.21.4. Сервер FTP

На CR1000 работает автоматический FTP-сервер. Это позволяет с помощью *Windows® Explorer®* получать доступ к файловой системе CR1000 через FTP, причем диски CR1000 сопоставляются с директориями или папками. Корневой директорией CR1000 может быть любой диск, однако обычно диску USR: отдается предпочтение. USR: это диск, созданный путем распределения памяти в поле **USR: размер диска** на вкладке **Развертывание | Расширенное** сервиса CR1000 в *DevConfig*. Файлы могут копироваться и передаваться между дисками. С помощью FTP можно удалять файлы.

7.9.21.5. FTP-клиент

CR1000 может выступать в роли FTP- клиента, отправляющего или получающего файл с сервера FTP, например другого регистратора данных или веб-камеры. Это выполняется с помощью команды CRBasic **FTPClient().** Подробная информация и примеры программ приведены в руководстве по сетевому подключению Campbell Scientific (см. приложение *Сетевые подключения* *(стр. 652)*), доступном на сайте *www.campbellsci.com* или в *Справке по редактору CRBasic*.

7.9.21.6. Telnet

Telnet используется для получения доступа к тем же командам, которые доступны в *терминальном эмуляторе ПО поддержки* *(стр. 530)*. Для запуска сеанса *Telnet* откройте командную строку DOS и наберите:

```
Telnet xxx.xxx.xxx.xxx <Enter>
```

где xxx.xxx.xxx.xxx — это IP-адрес сетевого устройства, подключенного к CR1000.

7.9.21.7. SNMP

Простой протокол сетевого управления (SNMP) является частью набора IP-протоколов, используемых в рамках NTCIP и RWIS для отслеживания дорожной ситуации. CR1000 поддерживает SNMP, если подключено сетевое стройство.

7.9.21.8. Проверка связи (IP)

Команда ping может использоваться для проверки доступности IP-адреса сетевого устройства, подключенного к CR1000. Чтобы воспользоваться инструментом Ping, откройте командную строку на компьютере, подключенном к сети, и наберите:

```
ping xxx.xxx.xxx.xxx <Enter>
```

где xxx.xxx.xxx.xxx — это IP-адрес сетевого устройства, подключенного к CR1000.

7.9.21.9. Микро-последовательный сервер

CR1000 можно сконфигурировать таким образом, чтобы разрешить последовательную связь через порт TCP/IP. Это полезно при коммуникации с последовательным датчиком через Ethernet с помощью микро-последовательного сервера (сторонний последовательный интерфейс Ethernet), к которому подключен последовательный датчик. Более подробную информацию см. в руководстве по сетевым подключениям и в *Справке по редактору CRBasic* для команды **TCPOpen()**. Информация о доступных сетевых подключениях приведена в приложении *Сетевые подключения* *(стр. 652)*.

7.9.21.10. Modbus TCP/IP

CR1000 может осуществлять коммуникацию Modbus по TCP/IP с помощью интерфейса Modbus TCP/IP. Для настройки Modbus TCP/IP укажите порт 502 как ***ComPort*** в командах **ModBusMaster()** и **ModBusSlave()**. Более подробная информация приведена в *Справке по редактору CRBasic*. См. *Modbus* *(стр. 411)*.

7.9.21.11. DHCP

При подключении к серверу со списком IP-адресов, доступных для присвоения, CR1000 автоматически запрашивает и получает IP-адрес по протоколу динамической конфигурации хоста (DHCP). Как только адрес присвоен, используйте команду *DevConfig, PakBusGraph, Connect* или экранную клавиатуру (CR1000KD Keyboard Display), чтобы найти в таблице состояния CR1000 (**Status**) присвоенный IP-адрес. Он отображается под именем поля ***IPInfo***.

7.9.21.12. DNS

CR1000 предоставляет клиент сервера доменных имен (DNS), который может запрашивать у DNS-сервера определение сопоставления IP-адреса с именем хоста. При положительном ответе имя хоста может равноправно использоваться с IP-адресом в некоторых командах регистратора данных.

7.9.21.13. SMTP

Простой протокол электронной почты (SMTP) является стандартным для передачи электронной почты. CR1000 можно запрограммировать на отправку электронных сообщений в соответствии со стандартным графиком или при возникновении определенных событий.

7.9.22. Вектор ветра

Команда **WindVector()** обрабатывает измерения скорости и направления ветра, высчитывая среднюю скорость, среднюю амплитуду вектора и среднее направление вектора в интервале хранения данных. Поддерживаются измерения с полярных (скорость и направление ветра) или ортогональных (зафиксированные восточные и северные винты) датчиков. Направление вектора и стандартное отклонение направления вектора могут быть рассчитаны для скорости ветра как взвешенно, так и не взвешенно.

7.9.22.1. Параметры OutputOpt

В команде CR1000 **WindVector()** параметр ***OutputOpt*** определяет обработанные данные, подлежащие хранению. Результатом для всех опций вывода является массив значений, элементы которого в качестве индекса имеют **_WVc(n)**, где **n** — номер элемента. Массив использует имя переменной ***Speed/East*** (скорость/восток) в качестве основы. В таблице *Опции OutputOpt* *(стр. 296)* перечислены и описаны опции ***OutputOpt***.

Таблица 53. WindVector(): опции OutputOpt

Опция	*Описание (WVc() — массив вывода)*
0	WVc(1): средняя горизонтальная скорость ветра (S) WVc(2): единичный вектор среднего направления ветра (Θ1) WVc(3): стандартное отклонение направления ветра σ(Θ1). Стандартное отклонение рассчитывается с помощью алгоритма Ямартино. Данная опция соответствует требованиям EPA касательно использования в моделях линейного распределения по Гауссу для моделирования струйного выброса.
1	WVc(1): средняя горизонтальная скорость ветра (S) WVc(2): единичный вектор среднего направления ветра (Θ1)
2	WVc(1): средняя горизонтальная скорость ветра (S) WVc(2): итоговая средняя горизонтальная скорость ветра (U) WVc(3): итоговое среднее направление ветра (Θu) WVc(4): стандартное отклонение направления ветра σ(Θu). Это стандартное отклонение рассчитывается с помощью взвешенного алгоритма скорости ветра Campbell Scientific. Использование итогового среднего горизонтального направления ветра не рекомендуется для моделей линейного распределения по Гауссу, однако это значение может применяться для моделирования направления переноса в модели траектории переменной.
3	WVc(1): единичный вектор среднего направления ветра (Θ1)

Таблица 53. WindVector(): опции OutputOpt	
Опция	***Описание (WVc() — массив вывода)***
4	WVc(1): единичный вектор среднего направления ветра (Θ1) WVc(2): стандартное отклонение направления ветра σ(Θu). Это стандартное отклонение рассчитывается с помощью взвешенного алгоритма скорости ветра Campbell Scientific. Использование итогового среднего горизонтального направления ветра не рекомендуется для моделей линейного распределения по Гауссу, однако это значение может применяться для моделирования направления переноса в модели траектории переменной.

7.9.22.2. Обработка вектора ветра

WindVector() использует нулевое измерение скорости ветра только при обработке скалярной скорости ветра. Поскольку для векторов требуется амплитуда и направление, измерения при нулевой скорости ветра не используются в расчетах скорости или направления ветра. Это означает, например, что рассчитанные вручную часовые направления вектора на основе 15-минутных направлений вектора не совпадают с рассчитанными в CR1000 часовыми направлениями вектора. Для корректного вычисления вручную часового направления вектора на основе 15-минутных направлений вектора требуется соответствующее взвешивание 15-минутных направлений вектора в соответствии с числом действительных (ненулевая скорость ветра) значений направления ветра.

Примечание. Для анемометров чашечного типа характерно механическое смещение, добавляемое к каждому измерению. Числовое смещение обычно закодировано в программе CRBasic с целью компенсации механического смещения. В таком случае измерение будет равно смещению, только если скорость ветра равна нулю; следовательно, часто включается дополнительный код для обнуления измерению, чтобы функция **WindVector()** могла отклонять измерения при нулевой скорости ветра.

Стандартное отклонение может обрабатываться одним из двух способов: 1) использование каждого значения, выбранного в течение периода хранения данных (введите ***0*** для параметра ***Подынтервал***), либо 2) усреднение стандартных отклонений после обработки на основе более коротких подынтервалов периода хранения данных. При усреднении стандартных отклонений подынтервалов минимизируется воздействие изгибов при легком ветре, а также поступает более полная информация по периодам переноса (см. публикацию EPA «On-site Meteorological Program Guidance for Regulatory Modeling Applications» («Локальная метеорологическая программа: руководство для моделирования в нормативной среде»).

Стандартное отклонение горизонтальных колебаний ветра на основе подынтервалов рассчитывается следующим образом:

$$\sigma(\Theta) = [((\sigma\Theta_1)^2 + (\sigma\Theta_2)^{2 \cdots +} (\sigma\Theta_M)^2) / M]^{1/2}$$

где: $\sigma(\Theta)$ — стандартное отклонение за период хранения данных, а $\sigma\Theta_1 \ldots \sigma\Theta_M$ — стандартные отклонения интервалов. Подынтервал указан в виде числа прогонов сканирования. Число прогонов сканирования для подынтервала задается следующим образом:

Требуемый подынтервал (сек) / частота сканирования (сек)

Например, если частота сканирования равна 1 секунде, а интервал вывода данных составляет 60 минут, стандартное отклонение рассчитывается на основании всех 3600 прогонов сканирования, если подынтервал равен 0. Если задан подынтервал в 900 прогонов сканирования (15 минут), стандартное отклонение равно среднему для стандартных отклонений четырех подынтервалов. Если последний подынтервал не содержит указанного числа прогонов сканирования, применяется процедура взвешивания.

Согласно рекомендации EPA, расчет часового стандартного отклонения горизонтального направления ветра (сигма тета) должно производиться на основе четырех пятнадцатиминутных интервалов.

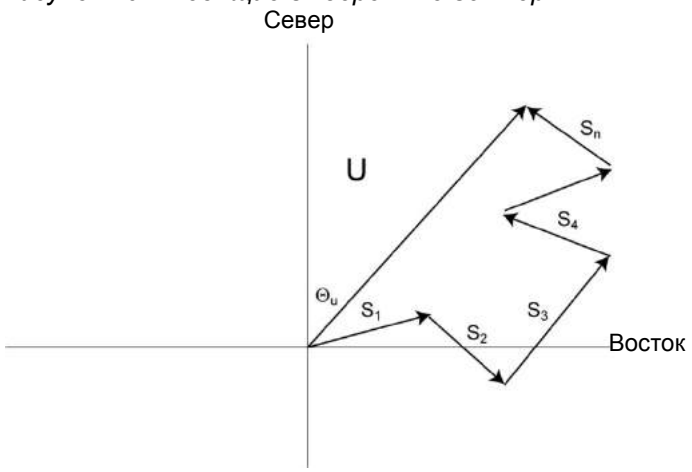
7.9.22.2.1. Измеренные необработанные данные

- S_i: горизонтальная скорость ветра
- Θ_i: горизонтальное направление ветра
- Ue_i: составляющая скорости ветра «запад-восток»
- Un_i: составляющая скорости ветра «север-юг»
- N: количество значений

7.9.22.2.2. Расчеты

Входящие выборочные векторы

Рисунок 76. Входящие выборочные векторы



На рисунке *Ввод выборки векторов* *(стр. 298)* короткие векторы, сцепленные по принципу «голова-хвост», являются входящими выборочными векторами, описываемыми параметрами s_i и Θ_i (выборочная скорость и направление) или Ue_i и Un_i (составляющие скорости выборочного вектора «восток» и «север»). В конце интервала хранения данных (T) сумма выборочных векторов описывается с помощью вектора с амплитудой U и направлением Θu.
Если интервал ввода выборочных значений равен t, количество выборочных значений в интервале хранения данных T *определяется как* N = T /t. *Средняя амплитуда вектора равна* $\bar{U}$ = U / N.

Скалярная средняя горизонтальная скорость ветра, S:

$$S = (\sum S_i) / N$$

где в случае ортогональных датчиков:

$$S_i = (Ue_i^2 + Un_i^2)^{1/2}$$

Единичный вектор среднего направления ветра,

$$\Theta_1 = \text{arctg}\ (Ux / Uy)$$

где

$$Ux = (\Sigma \sin\Theta_i) / N$$

$$Uy = (\Sigma \cos\Theta_i) / N$$

или, в случае ортогональных датчиков

$$Ux = (\Sigma (Ue_i / U_i) / N$$

$$Uy = (\Sigma (Un_i / U_i) / N$$

где

$$U_i = (Ue_i^2 + Un_i^2)^{1/2}$$

Стандартное отклонение направления ветра (алгоритм Ямартино)

$$\sigma(\Theta_1) = \arcsin(\varepsilon)[1 + 0{,}1547\varepsilon^3]$$

где

$$\varepsilon = [1 - ((Ux)^2 + (Uy)^2)]^{1/2}$$

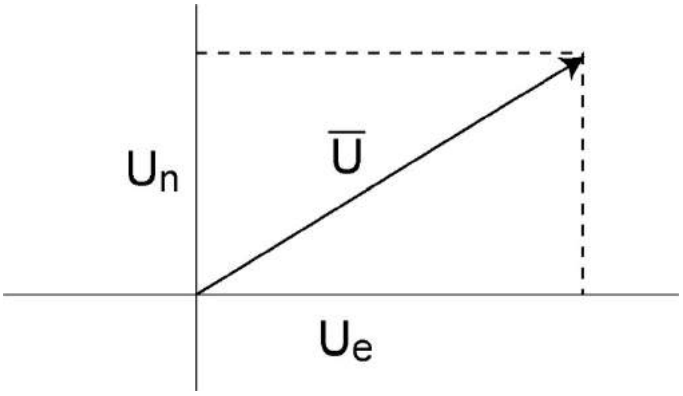
Ux и Uy определены выше.

Средний вектор ветра

Итоговая средняя горизонтальная скорость ветра, $\bar{U}$:

$$\bar{U} = (Ue^2 + Un^2)^{1/2}$$

Рисунок 77. Графическое изображение среднего вектора ветра



где для полярных датчиков:

$$Ue = (\Sigma s_i \sin\Theta_i) / N$$

$$Un = (\Sigma s_i \cos\Theta_i) / N$$

или, в случае ортогональных датчиков:

$$Ue = (\Sigma Ue_i) / N$$

$$Un = (\Sigma Un_i) / N$$

Итоговое среднее направление ветра, Θu:

$$\Theta u = arctg\ (Ue / Un)$$

Стандартное отклонение направления ветра, σ(Θu), по алгоритму Campbell Scientific:

$$\sigma(\Theta u) = 81(1 - \bar{U} / S)^{1/2}$$

При разработке алгоритма для σ (Θu) следует обратить внимание, как показано на рисунке *Стандартное отклонение направления* *(стр. 300)*, что

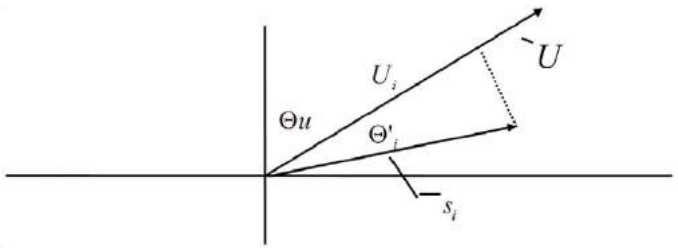
$$\cos(\Theta_i') = U_i / s_i$$

где

$$\Theta_i' = \Theta_i - \Theta u$$

Стандартное отклонение направления

Рисунок 78. Стандартное отклонение направления



Ряд Тейлора для косинусоидной функции, сокращенный после второго члена:

$$\cos(\Theta_{ii}^{'}) \cong 1 - (\Theta_i')^2 / 2$$

Для отклонений менее 40 градусов погрешность приближения составляет менее 1 %. Для отклонений в 60 градусов погрешность равна 10 %.

Выборочное значение скорости может быть выражено как отклонение от средней скорости,

$$s_i = s_i' + S$$

Приравниваем два выражения для Cos (θ') и используем предыдущее уравнение для s_i:

$$1 - (\Theta_i')^2 / 2 = U_i / (s_i' + S)$$

Находя решение для $(\Theta_i')^2$, получим:

$$(\Theta_i')^2 = 2 - 2U_i / S - (\Theta_i')^2 s_i' / S + 2s_i' / S$$

При суммировании $(\Theta_i')^2$ для N выборочных значений и делении на N получается дисперсия для Θu.

Примечание. Сумма последнего члена равна 0.

$$(\sigma(\Theta u))^2 = (\sum_{i=1}^{N}(\Theta_i')^2 / N) = 2(1 - \bar{U}/S) - \sum_{i=1}^{N}((\Theta_i)^2 s_i')/ NS$$

Значение члена

$$\Sigma((\Theta_i')^2 s_i') / NS$$

равно 0, если отклонения скорости не коррелируют с отклонением направления. Это допущение было проверено в ходе тестов данных ветра специалистами Campbell Scientific; Лаборатории исследования воздушных ресурсов, NOAA, Айдахо-Фолс, штат Айдахо; а также MERDI, Бьютт, штат Монтана. В ходе этих тестов максимальная разница в

$$\sigma(\Theta u) = (\Sigma((\Theta_i')^2 / N)^{1/2}$$

и

$$\sigma(\Theta u) = (2(1 - \bar{U} / S))^{1/2}$$

не превышала несколько градусов.

Для приведения в итоговый вид радианы преобразуются в градусы (57,296 градуса/радиан).

$$\sigma(\Theta u) = (2(1 - \bar{U} / S))^{1/2} = 81(1 - \bar{U} / S)^{1/2}$$

8. Эксплуатация

Подробнее:
- *Краткое руководство* *(стр. 41)*
- *Технические характеристики* *(стр. 97)*
- *Установка* *(стр. 99)*
- *Эксплуатация* *(стр. 303)*

8.1. Измерения — подробная информация

Связанные темы:
- *Датчики — краткое руководство* *(стр. 42)*
- *Измерения — общие сведения* *(стр. 62)*
- *Измерения — подробная информация* *(стр. 303)*
- *Датчики — списки* *(стр. 649)*

Благодаря некоторым функциям CR1000 обладает достаточной гибкостью для измерения с использованием большинства типов датчиков. Обратитесь к специалисту Campbell Scientific по внедрению, если требуется помощь в оценке совместимости CR1000 с конкретным приложением или типом датчика. Для некоторых датчиков требуется точный уровень тока возбуждения или источник питания. См. раздел *Переключаемое выходное напряжение — подробная информация* *(стр. 103)*.

8.1.1. Хронометраж — подробная информация

Связанные темы:
- *Хронометраж — общие сведения* *(стр. 75)*
- *Хронометраж — подробная информация* *(стр. 303)*

Измерение времени является ключевой функцией регистратора CR1000. Измерение времени с помощью встроенных часов позволяет регистратору CR1000 присваивать временные метки данным, измерять интервалы между событиями и назначать время запуска функций управления.

8.1.1.1. Временные метки

Измерение без точной привязки по времени имеет мало смысла. Данные в CR1000 хранятся с временными метками. Степень соответствия временной метки фактическому времени выполнения измерения зависит от нескольких факторов.

Временная метка в стандартных программах CRBasic соответствует времени начала текущего прогона сканирования, зарегистрированного внутренними часами CR1000. Если сканирование начинается в 15:00:00, выходящим данным, относящимся к этому прогону сканирования, будет присвоена временная метка **15:00:00**, независимо от продолжительности сканирования или конкретного времени измерения в процессе сканирования. Существует вероятность, что сканирование уже будет выполняться в течение определенного времени до момента измерения. Например, сканирование может начаться в 15:00:00, выполнить времязатратный код, а затем произвести измерение в 15:00:00.51. Если команда **CallTable()** вызывается из конструкции **Scan() / NextScan**, измерению будет присвоена временная метка **15:00:00**, что в итоге даст сдвиг временной метки на 510 мс.

В большинстве случаев сдвиг временной метки не является проблемой, поскольку:

- Выполнение программы обычно не требует много времени, так что сдвиг временной метки составляет всего несколько миллисекунд.

Большинство требований к измерениями допускают сдвиг на несколько миллисекунд.

- Данные, преобразованные в средние, максимальные, минимальные значения и т. д., являются элементами нескольких измерений. Связанные с ними временные метки отражают только момент выполнения последнего измерения и расчетов в плане обработки, так что значимость точного времени измерения конкретного выборочного значения снижается.

Сдвиг временной метки может негативно повлиять на приложения, предназначенные для измерения и хранения выборочных данных и требующие точных временных меток. Сдвига можно избежать за счет:

- Выполнения измерений в процессе сканирования до запуска времязатратного кода.
- Программирования CR1000 таким образом, чтобы временная метка отражала системное время, а не время сканирования. Если **CallTable()** выполняется из конструкции **Scan() / NextScan**, как это обычно и происходит, временная метка отражает время сканирования. При выполнении команды **CallTable()** вне конструкции **Scan() / NextScan**, временная метка будет отражать системное время вместо времени сканирования. В примере CRBasic *Присвоение временных меток с системным временем* *(стр. 304)* продемонстрированы базовые требования к коду. Команда **DataTime()** — это нововведение, упрощающее присвоение временных меток, отражающих системное время. Более подробную информацию см. в разделе *Объявления таблиц данных* *(стр. 540)* и в *Справке по редактору CRBasic*.

CRBasic: пример 63. Присвоение временных меток с системным временем

```
'В этом примере программы демонстрируется присвоение временных меток данных с системным временем
вместо используемого по умолчанию времени сканирования (момента начала сканирования).
'
'Объявление переменных
Public value

'Объявление таблицы данных
DataTable(Test,True,1000)
   Sample(1,Value, FP2)
EndTable

SequentialMode

BeginProg
   Scan(1,Sec,10,0)

      'Задержка – в операционной программе задержка может быть вызвана другим кодом
      Delay(1,500,mSec)

      'Измеренное значение – может быть любым аналоговым измерением
      PanelTemp(Value,0)

      'Немедленный вызов SlowSequence для выполнения CallTable()
      TriggerSequence(1,0)

   NextScan

'Возможность сохранения данных с разницей в 510 мс относительно времени сканирования и с
временной меткой s.51.
   SlowSequence
      Do
       WaitTriggerSequence
       CallTable(Test)
      Loop

EndProg
```

Остальные команды CRBasic, предназначенные для обработки времени, регулируются теми же правилами. Более подробная информация о конкретных командах доступна в *Справке по редактору CRBasic*.

8.1.2. Аналоговые измерения — подробная информация

Связанные темы:
- *Аналоговые измерения — общие сведения* *(стр. 63)*
- *Аналоговые измерения — подробная информация* *(стр. 305)*

CR1000 измеряет следующие типы аналогового выхода датчиков:

- Напряжение
  - Несимметричное
  - Дифференциальное
- Ток (с помощью резистивного шунта)
- Сопротивление
- Подключение с полным мостом
- Полумостовое подключение

Датчик подключается к клеммам **H/L]**, настраиваемым под дифференциальные (**DIFF**) или несимметричные (**SE**) входы. Например, дифференциальный канал 1 включает клеммы **1H** и **1L**, причем клемма **1H** предназначена для высоких значений, а **1L** — для низких.

8.1.2.1. Измерения напряжения — подробная информация

Связанные темы:
- Измерения по напряжению — технические характеристики
- *Измерения по напряжению — общие сведения* *(стр. 63)*
- *Измерения по напряжению — подробная информация* *(стр. 305)*

8.1.2.1.1. Механика измерения напряжения

Последовательность измерений

Аналоговое измерение напряжения, изображенное на рисунке *Упрощенная последовательность измерений напряжения* *(стр. 306)*, выполняется следующим образом:

1. Переключение
2. Успокоение
3. Усиление
4. Интеграция
5. А/Ц (последовательное приближение)
6. Масштабированное измерение с множителем и смещением
7. Масштабированное значение в памяти

РИСУНОК. Упрощенная последовательность измерения напряжения — 8 10 30

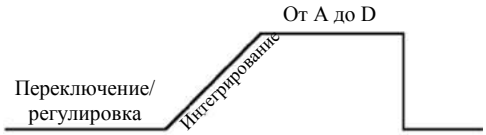


Рисунок 79. Упрощенная последовательность измерения напряжения

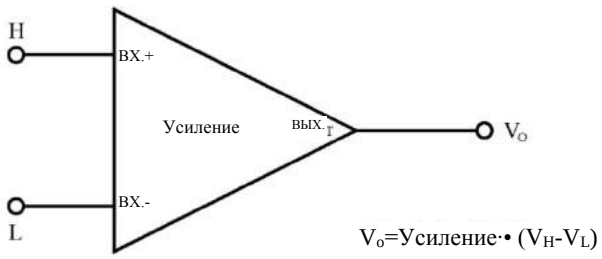
Измерения напряжения производятся с помощью последовательного приближения *A/Ц-преобразователя* *(стр. 507)* с целью достижения разрешающей способности в 14 бит. Еще до начала применения А/Ц преобразования сигнал усиливается с помощью высокоомного измерительного усилителя с программируемым усилением (PGIA). См. рисунок *Программируемый входной усилитель (PGIA)* *(стр. 306)***.** Программа CRBasic контролирует уровень усиления и конфигурацию усилителя — несимметричный или дифференциальный вход. Внутренние мультиплексоры направляют сигнал из отдельных клемм в PGIA.

Временные рамки задач измерения тщательно контролируются. Расписание измерений определяется в момент компиляции и загружается в память.

При использовании двух различных команд измерения напряжения в рамках одного и того же диапазона напряжения требуется примерно в два раза больше времени, чем на применение одной команды, повторенной дважды.

Параметры, перечисленные в таблице *Параметры CRBasic, варьирующие последовательность и временные рамки измерений* *(стр. 307)***,** обеспечивают вариативность последовательности и временных рамок для команд измерения напряжения.

Рисунок 80. Программируемый входной усилитель (PGIA)



Измерение напряжения выполняется следующим образом:

1. Устанавливается усиление PGIA в рамках выбранного диапазона напряжения в соответствии с параметром ***Range*** («Диапазон») команды измерения CRBasic.
2. Ток возбуждения подается и доводится до уровня, заданного параметром ***ExmV***.
3. Выбранные клеммы (***InChan***) мультиплексируются в PGIA, определяется задержка для указанного времени успокоения (***SettlingTime***).
4. Сигнал интегрируется (см. раздел *Интеграция измерения* *(стр. 307)*), выполняется А/Ц-преобразование.
5. Действия повторяются для инверсии тока возбуждения и входящего сигнала согласно параметрам ***RevEx*** и ***RevDiff***.
6. К измеренному результату применяется множитель (***Mult***) и смещение (***Offset***).

CR1000 измеряет аналоговое напряжение путем интеграции входящего сигнала в течение фиксированного отрезка времени, а затем сохраняя интегрированное значение в процессе последовательного приближения при аналого-цифровом преобразовании (А/Ц). CR1000 может выполнять и сохранять измерения, данные которых поступают максимум из 8 дифференциальных или 16 несимметричных каналов, настроенных через **H/L**-клеммы на минимальный интервал сканирования, равный 10 мс (100 Гц), с применением технологий программирования быстрого измерения, описанных в разделе *Измерения: повышенные аналоговые частоты* *(стр. 229)***.** Максимальная скорость преобразования составляет 2000 в секунду (2 кГц) для измерений, выполненных по несимметричному каналу.

Таблица 54. Параметры CRBasic, варьирующие последовательность и временные рамки измерений

Параметр CRBasic	*Описание*
MeasOfs	Корректное смещение заземления для несимметричных измерений.
SettlingTime	Время успокоения на входе датчика.
Целое (Integ)	Длительность интеграции входящего сигнала.
RevDiff	Инверсия дифференциальных входов высокого и низкого уровня.
RevEx	Инверсия полярности напряжения возбуждения.

Интеграция измерения

Благодаря интеграции сигнала исчезают помехи, являющиеся причиной погрешности измерения. Медленная интеграция устраняет больше помех, чем быстрая интеграция. Время интеграции можно изменять, чтобы устранить помехи в линии питания от сети с частотой 50 Гц и 60 Гц.

Время от времени предпочтительной может становиться быстрая интеграция, например:

- для минимизации сдвига между последовательными измерениями;
- для максимизации пропускной способности;
- для максимизации срока службы блока питания CR1000;
- для минимизации поляризации полярных датчиков, например используемых для измерения проводимости, влажности почвы или влажности листа. Поляризация может стать причиной погрешности измерения или снижения эффективности датчика;
- повышения точности измерения линейно регулируемого дифференциального трансформатора (LVDT). Индуцированное напряжение в LVDT со временем ослабевает по мере того, как ток в первичной катушке переходит с индуктора в контур последовательного сопротивления; результатом долгой интеграции может стать затухание большинства сигналов до завершения измерения.

Несимметричные измерения — подробная информация

Связанные темы:

- *Несимметричные измерения — общие сведения* *(стр. 65)*
- *Несимметричные измерения — подробная информация* *(стр. 307)*

Согласно рисунку *Программируемый входной усилитель (PGIA)* *(стр. 306)***,** при несимметричном измерении сигнал высокого уровня (H) направляется в V+. Сигнал низкого уровня (L) автоматически подключается к заземлению сигнала

изнутри, причем сигнал заземляется (⏚) через коммутационную панель. V+ соответствует нечетным или четным клеммам **SE** на коммутационной панели CR1000. Несимметричная конфигурация используется для следующих команд CRBasic:

- **VoltSE()**
- **BrHalf()**
- **BrHalf3W()**
- **TCSE()**
- **Therm107()**
- **Therm108()**
- **Therm109()**
- **Thermistor**()

Связанные темы:
- *Дифференциальные измерения — общие сведения* *(стр. 66)*
- *Дифференциальные измерения — подробная информация* *(стр. 308)*

Дифференциальные измерения — подробная информация

Согласно рисунку *Программируемый входной усилитель (PGIA)* *(стр. 306)***,** при дифференциальном измерении сигнал высокого уровня (H) направляется в V+, а сигнал низкого уровня (L) — в V–**.**

Клемма **H** дифференциала контактной пары **H/L** соответствует V+. Клемма **L** соответствует V–. Дифференциальная конфигурация используется для следующих команд CRBasic:

- **VoltDiff()**
- **BrFull()**
- **BrFull6W()**
- **BrHalf4W()**
- **TCDiff()**

8.1.2.1.2. Ограничения измерения напряжения

Внимание! При длительном воздействии напряжения свыше ±8,6 В на контакты, настроенные под аналоговый вход, могут быть нарушены все аналоговые измерения.

Предупреждение! При длительном воздействии напряжения свыше ±16 В на контакты, настроенные под аналоговый вход, будет повреждена электрическая схема CR1000.

Диапазоны напряжения

Связанные темы:
- Измерения по напряжению — технические характеристики
- *Измерения по напряжению — общие сведения* *(стр. 63)*
- *Измерения по напряжению — подробная информация* *(стр. 305)*

Как правило, используется наименьший постоянный входной диапазон, полностью охватывающий возможные выходные показания датчика. Это обеспечивает максимальную разрешающую способность и точность измерений. Для CR6 характерны фиксированные входные диапазоны измерений напряжения, а также функция автоматического определения подходящего входного диапазона напряжения для конкретного измерения. В таблице *Аналоговое напряжение: входные диапазоны и опции* *(стр. 309)* перечислены эти входные диапазоны и коды.

В постоянных диапазонах входного напряжения предусмотрен запас, который равен примерно 9 % от диапазона. Иными словами, превышение входного диапазона ±2500 мВ произойдет примерно при выходе за значения 2725 мВ и -2725 мВ. CR1000 указывает на превышение диапазона измерения, возвращая значение **NAN** (не является числом).

Автоматическое определение диапазона

В случае сигналов, для которых не характерны быстрые колебания, аргумент диапазона ***AutoRange*** позволяет CR1000 автоматически выбирать диапазон напряжения. ***AutoRange*** производит два измерения. Первое измерение определяет диапазон, который будет использоваться. Для этого выполняется интеграция 250 мкс в диапазон ±5000 мВ. Второе измерение производится с использованием диапазона, определенного на основе первого. В обоих измерениях используется время успокоения, указанное в параметре ***SettlingTime***. Автоматическое определение диапазона оптимизирует разрешающую способность, однако требует больше времени, чем измерение на основании постоянного диапазона, ввиду необходимости двух последовательностей измерения.

При автоматическом определении диапазона будет возвращено значение **NAN** («не является числом»), если напряжение превысит диапазон, заданный в первом измерении. Во избежание проблем с сигналом на границе диапазона функция ***AutoRange*** выбирает следующий наибольший диапазон, если сигнал превышает 90 % диапазона.

Автоматическое определение диапазона следует использовать для сигнала, который периодически выходит за рамки определенного диапазона; например, в случае термопары типа J обычно измеряется температура ниже 476 °C (диапазон ±25 мВ), однако временами температура достигает 500 °C (диапазон ±250 мВ).

AutoRange не следует использовать для сигналов с высокой частотой колебаний, особенно для сигналов, быстро проходящих через несколько диапазонов напряжения. Существует вероятность, что сигнал способен менять диапазоны в интервале между внутренней проверкой диапазона и фактическим измерением.

Таблица 55. Аналоговое напряжение: входные диапазоны и опции

Диапазон кода	*Описание*
mV5000	измеряет напряжение в пределах ±5000 мВ
mV2500[1]	измеряет напряжение в пределах ±2500 мВ
mV250[2]	измеряет напряжение в пределах ±250 мВ
mV25[2]	измеряет напряжение в пределах ±25 мВ
mV7_5[2]	измеряет напряжение в пределах ±7,5 мВ
mV2_5[2]	измеряет напряжение в пределах ±2,5 мВ
AutoRange[3]	регистратор данных определяет наиболее подходящий диапазон

[1] В конце добавляется ***C***, чтобы активировать обнаружение синфазного нулевого сигнала/размыкания и установить полный диапазон возбуждения (~2700 мВ) (пример: ***mV2500***)

[2] В конце добавляется ***C***, чтобы активировать обнаружение синфазного нулевого сигнала/размыкания (пример: ***mV25C***)

[3] В конце добавляется ***C***, чтобы активировать обнаружение синфазного нулевого сигнала/размыкания для диапазонов ≤ ±250 мВ, либо только синфазного нулевого сигнала для диапазонов > ±250 мВ (пример: ***AutoRangeC***)

Входные пределы / синфазный диапазон

Связанные темы:
- Измерения по напряжению — технические характеристики
- *Измерения по напряжению — общие сведения* *(стр. 63)*
- *Измерения по напряжению — подробная информация* *(стр. 305)*

Примечание. Содержащаяся в данном разделе расширенная информация не является необходимой для обычной эксплуатации CR1000.

Краткое описание

- Входные пределы напряжения для измерения равны ±5 В пост. тока. Входные пределы как техническая характеристика представлены в разделе *Технические характеристики* *(стр. 97)***.**
- Синфазный диапазон не является фиксированным числом. Его *значение* меняется в зависимости от амплитуды входного напряжения.
- CR1000 обладает функциями, помогающими смягчить некоторые аспекты влияния сигналов, выходящих за рамки технической характеристики *входных пределов* или синфазного диапазона.

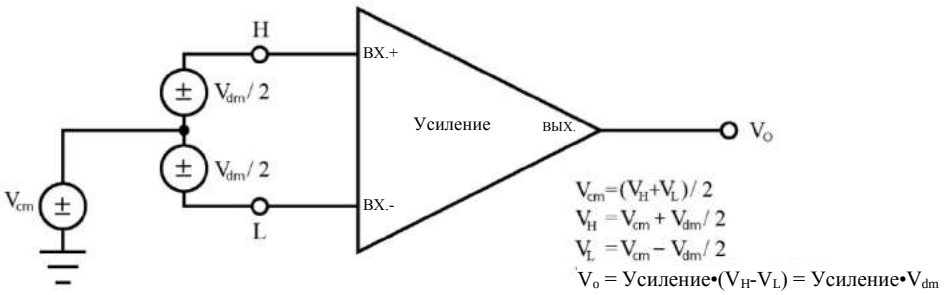
Согласно рисунку *PGIA с разложением входящего сигнала* *(стр. 311)*, PGIA обрабатывает разность потенциалов между V+ и V–. При этом игнорируется синфазное напряжение, а также напряжение, являющееся общим для обоих входов. На рисунке показано подаваемое напряжение, разложенное на синфазное напряжение (V_{cm}) и дифференциальный компонент (V_{dm}) сигнала напряжения. V_{cm} является средним значением напряжений на входах V+ и V–. Таким образом, $V_{cm} = (V+ + V–)/2$ или остаточному напряжению на входах, если $V_{dm} = 0$. Суммарное напряжение на входах V+ и V– задается как $V+ = V_{cm} + V_{dm}/2$ и $V– = V_{cm} – V_{dm}/2$ соответственно.

PGIA игнорирует или отбрасывает значения синфазного напряжения, пока напряжение на V+ и V– соответствует технической характеристике *входных пределов*, которые в случае CR6 составляют ±5 В пост. тока относительно земли. Если входное напряжение, будь то V+ или V– либо оба одновременно, находится вне пределов ±5 В пост. тока, могут возникнуть скрытые погрешности измерения. Параметр *Синфазный диапазон* определяет диапазон синфазных напряжений, которые не должны становиться причиной скрытых погрешностей измерения. *Синфазный диапазон* отличается от *Входных пределов*, если дифференциальное напряжение заметно. Следующее соотношение выводится на основе рисунка PGIA:

$$\text{Синфазный диапазон} = \pm 5 \text{ В пост. тока} – |V_{dm}/2|.$$

Отсюда следует, что синфазный диапазон не выражается фиксированным числом, а вместо этого уменьшается при повышении дифференциального напряжения. В случае сравнительно малых дифференциальных напряжений (относительно входных пределов) синфазный диапазон существенно эквивалентен параметру *Входные пределы*. В случае дифференциального сигнала, равного 5000 мВ, синфазный диапазон сокращается до ±2,5 В пост. тока, в то время как Входные пределы *всегда составляют ±5 В пост. тока.* Следовательно, параметр *Входные пределы* используется для определения допустимого диапазона напряжения на входах V+ и V– в PGIA.

Рисунок 81. PGIA с разложением входящего сигнала



8.1.2.1.3. Качество измерения напряжения

Читать далее. В следующих технических документах, доступных по адресу *www.campbellsci.com/app- notes (http://www.campbellsci.com/app-notes)*, углубленно рассматриваются определенные темы, касающиеся качества измерения напряжения:

- *Предотвращение и устранение проблем, связанных с помехами при измерении*
- *Преимущества инверсии входящего сигнала и тока возбуждения для измерения напряжения*
- *Погрешность измерения напряжения, внутренняя калибровка, логометрические измерения*
- *Оценка погрешности измерения для команд логометрических измерений.*

В следующих темах обсуждаются методы общего улучшения измерений напряжения. Соответствующая информация по особым случаям измерения напряжения (*термопары (стр. 327), токовые контуры (стр. 337), сопротивление (стр. 337), перегрузка (стр. 342)*) доступна в разделах, посвященных этим измерениям.

Несимметричное или дифференциальное?

Выбор подходящего измерения между дифференциальным и несимметричным обычно является бесспорно важнейшим критерием в плане качества измерения напряжения. Для принятия решения требуется найти компромисс между погрешностью и точностью, устранением помех, скоростью измерения, доступными инструментальными средствами измерения и бюджетными ограничениями.

В общих чертах аналоговое напряжение наилучшим образом измеряется в дифференциальной форме, поскольку такие измерения включают перечисленные ниже функции снижения помех, не являющиеся частью несимметричных измерений.

- Пассивное подавление шумов
  - Без смещения опорного напряжения
  - Подавление синфазных шумов с фильтрацией емкостно-связанных помех
- Активное подавление шумов
  - Инверсия входящего сигнала
    - Подробнее см. раздел *Инверсия входящего сигнала и тока возбуждения (стр. 325)*
    - Двукратно увеличивает время интеграции инвертированного входящего сигнала

Среди причин использования несимметричных измерений выделяются, тем не менее:

- Недостаточное количество дифференциальных клемм. Дифференциальные измерения используют в два раза больше клемм **H/L]**, чем несимметричные измерения.
- Требуется быстрая выборка. Несимметричные измерения выполняются примерно в два раза быстрее дифференциальных измерений.
- Датчик не предназначен для дифференциальных измерений. Многие датчики Campbell Scientific не предназначены для дифференциальных измерений, но погрешность несимметричного измерения обычно компенсируется большим запрограммированным возбуждением и/или выходным напряжением датчика.

Возможность нормального измерения для датчиков с высоким соотношением сигнал/шум, таких как датчик относительной влажности с полным выходным диапазоном от 0 до 1000 мВ, по несимметричному методу без существенного снижения точности.

В случае датчиков с низким соотношением сигнал/шум, таких как термопары, следует применять дифференциальное измерение. Однако, если выполняемое измерение не требует высокой точности (например, при измерении с помощью термопар локальных температур, способных превысить 2500 °C), может оказаться подходящим и несимметричное измерение. Если датчики требуют дифференциального измерения, но соответствующие входные клеммы недоступны, необходимо приобрести аналоговый мультиплексор для увеличения дифференциального входного номинала. Информацию о доступных мультиплексорах можно найти в приложении *Аналоговые мультиплексоры* *(стр. 646)***.**

Поскольку несимметричные измерения CR1000 используют заземление, любое изменение в электрическом потенциале земли между датчиком и CR1000 приведет к погрешности измерений. Например, если рабочий спай медно-константановой термопары, используемой для измерения температуры почвы, не изолирован, а потенциал земли на датчике на 1 мВ выше, чем в точке заземления CR1000, измеренное напряжение будет на 1 мВ больше истинного значения на выходе термопары, что дает превышение температуры в отчете примерно на 25 °C. Общая проблема с разностью потенциалов земли возникает в тех случаях, когда внешняя схема формирования сигнала питается от того же источника, что и CR1000 (например, от электрической розетки сети переменного тока). Несмотря на подключение к одной и той же земле, разница между отдачей тока и сопротивлением проводов может привести к различному потенциалу земли двух измерительных приборов. Поэтому в качестве меры предосторожности следует выполнять дифференциальное измерение на аналоговом выводе из внешнего блока преобразования сигналов; дифференциальное измерение ДОЛЖНО применяться в тех случаях, когда точно известно, что низкоуровневый сигнал отличается от земли.

Электронный шум

Электронный «шум» может стать причиной существенной погрешности измерения напряжения, особенно когда измеряются напряжения менее 200 мВ. Пока соблюдаются входные ограничения, PGIA игнорирует напряжения, включая шум, которые являются общими для каждой из сторон дифференциальной входной пары. Это синфазное напряжение. Игнорирование (отклонение или сброс) синфазного напряжения является важной функцией дифференциального конфигурации входа, оптимизирующей измерения напряжения.

На рисунке *PGIA с Разложением входящего сигнала* *(стр. 311)* представлены синфазный (V-cm) и дифференциальный (Vdm) компоненты сигнала напряжения. Vcm является средним значением напряжений на входах V+ и V–. Таким образом, Vcm = (V+ + V–)/2 или остаточному напряжению на входах, если Vdm = 0. Суммарное напряжение на входах V+ и V– задается как V+ =

Vcm + Vdm/2 и VL = Vcm – Vdm/2 соответственно.

Погрешность измерений

Читать далее. Оценка погрешности углубленно рассматривается в техническом документе *Анализ погрешности измерений*, доступном по адресу *www.campbellsci.com/app-notes (http://www.campbellsci.com/app-notes).*

Погрешность представляет собой разницу между измеренным и истинным значением. На погрешность влияет множество факторов. В данном разделе описывается влияние, которое оказывают процент от показаний, смещение и разрешающая способность на погрешность измерения сигнала с датчика аналогового напряжения. Погрешность определяется следующим образом:

погрешность = процент от показаний + смещение

где проценты от показаний сгруппированы в таблице *Погрешность измерения аналогового напряжения* *(стр. 313)*, а смещения — в таблице *Смещения измерения аналогового напряжения* *(стр. 313)*.

Примечание. Погрешность, описываемая в данном разделе, а также связанные технические характеристики CR1000 не включают погрешности, вызванные самим датчиком или передачей сигнала датчика в CR1000.

Таблица 56. Погрешность измерения аналогового напряжения[1]

От 0 до 40 °C	*От –25 до 50 °C*	*От –55 до 85 °C*[2]
±(0,06 % от показаний + смещение)	±(0,12 % от показаний + смещение)	±(0,18 % от показаний + смещение)

[1] Предполагается, что CR1000 соответствует заводским техническим характеристикам

[2] Доступно только в случае приобретения расширенной опции температуры (-XT)

Таблица 57. Смещения измерения аналогового напряжения

Дифференциальное измерение с инверсией входящего сигнала	*Дифференциальное измерение без инверсии входящего сигнала*	*Несимметричное*
1,5 • Базовое разрешение + 1,0 мкВ	3 • Базовое разрешение + 2,0 мкВ	3 • Базовое разрешение + 3,0 мкВ

Примечание. Значение базового разрешения находится в таблице *Разрешение измерения аналогового напряжения* *(стр. 313)*.

Таблица 58. Разрешение измерения аналогового напряжения

Диапазон входного напряжения (мВ)	*Дифференциальное измерение с инверсией входящего сигнала (мкВ)*	*Базовое разрешение (мкВ)*
±5000	667	1333
±2500	333	667
±250	33,3	66,7
25	3.33	6.7

7,5	1,0	2,0
2,5	0,33	0,67
Примечание. Полное табличное представление разрешений измерения представлено в разделе *Технические характеристики* *(стр. 97)*		

Например, на рисунке *Пример диапазона погрешности измерения напряжения* *(стр. 314)* показаны колебания погрешности по мере изменения входного напряжения в диапазоне ±2500 мВ. Процент от показаний является основным компонентом, так что погрешность увеличивается при снижении входного напряжения. Смещение невелико, однако может оказаться существенным в случае, когда напряжение сигнала датчика крайне мало, что характерно для термопар.

Смещение зависит от типа измерения и диапазона входного напряжения. Уравнения смещений сгруппированы в таблице *Смещения измерения аналогового напряжения* *(стр. 313)***.** Например, для дифференциального измерения с инверсией входящего сигнала в рамках входного диапазона ±5000 мВ напряжение смещения рассчитывается следующим образом:

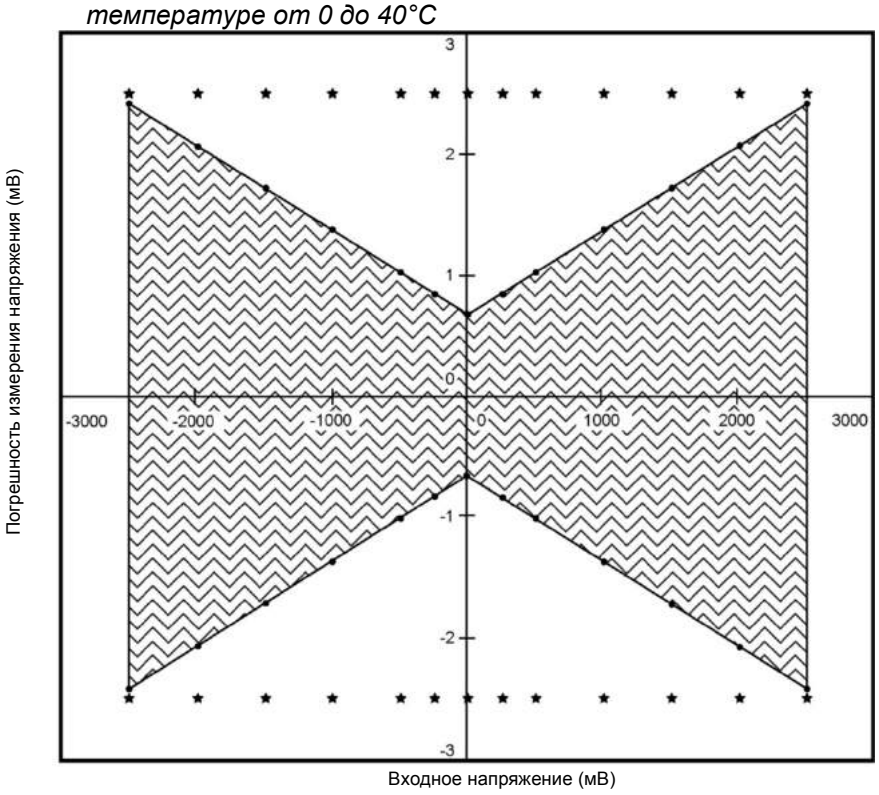
смещение = 1,5 • Базовое разрешение + 1,0 мкВ

= (1,5 • 667 мкВ) + 1,0 мкВ

= 1001,5 мкВ

где базовое разрешение выражается опубликованным значением разрешения, взятым из таблицы *Разрешение измерения аналогового напряжения* *(стр. 313)***.**

Рисунок 82. Пример диапазона погрешности измерения напряжения, включая влияние процента от показаний и смещения, для дифференциального измерения с инверсией входящего сигнала при температуре от 0 до 40°C



Пример погрешности измерения

Следующий пример иллюстрирует влияние процента от показаний и смещения на погрешность измерения. В случае сильного сигнала влияние смещения обычно незаметно:

Пример:

- Напряжение сигнала датчика: ≈2500 мВ
- Команда измерения CRBasic: **VoltDiff()**
- Запрограммированный диапазон входного напряжения (***Range***): ***mV2500*** (±2500 мВ)
- Инверсия измерения входящего сигнала (***RevDiff***): True ***(истина)***
- Температура электрической схемы CR1000: 10 °C

Погрешность измерения вычисляется следующим образом:

погрешность = процент от показаний + смещение

где

процент от показаний = 2500 мВ • ±0,06 %

= ±1,5 мВ

и

смещение = (1,5 • 667 мкВ) + 1 мкВ

= 1,00 мВ

Таким образом,

погрешность = ±1,5 мВ + 1,00 мВ

= ±2,5 мВ

Интеграция

Встроенная электрическая схема CR1000 позволяет выполнять аналоговую интеграцию напряжений и производить измерение до *А/Ц-преобразования* *(стр. 507)*. Благодаря интеграции аналогового сигнала исчезают помехи, являющиеся причиной погрешности измерения. Медленная интеграция устраняет больше помех, чем быстрая интеграция. Если продолжительность интеграции совпадает с длительностью одного цикла помех в сети переменного тока, помехи отфильтровываются. В таблице *Интеграция аналогового измерения* *(стр. 316)* перечислены допустимые аргументы продолжительности интеграции.

Ускоренная интеграция может быть выбрана для достижения следующих целей:

- Минимизация временного сдвига между последовательными измерениями
- Максимизация пропускной способности
- Максимизация срока службы блока питания CR1000
- Минимизация поляризации полярных датчиков, например используемых для измерения проводимости, влажности почвы или влажности листа. Поляризация может стать причиной погрешности измерения или снижения эффективности датчика.
- Повышение точности измерения линейно регулируемого дифференциального трансформатора (LVDT). Индуцированное напряжение в LVDT со временем ослабевает по мере того, как ток в первичной катушке переходит с индуктора в контур последовательного сопротивления; результатом долгой интеграции может стать затухание большинства сигналов до завершения измерения.

Читать далее. См. технический документ «Предотвращение и устранение проблем с помехами измерения» на сайте *www.campbellsci.com.*

Амплитуда частотной характеристики аналогового интегратора имеет форму функции SIN(x)/x с провалами (нулями коэффициента передачи) в значениях продолжительности интеграции, равных 1/(целое кратное). Следовательно, помехи в точках 1/(целое кратное) продолжительности интеграции успешно отсекаются аналоговым интегратором. Если задана инверсия дифференциальных входящих сигналов или тока возбуждения, на каждое измерение приходится два отдельных процесса интеграции; если заявлены обе инверсии, количество отдельных процессов интеграции возрастает до четырех.

Таблица 59. Интеграция аналогового измерения

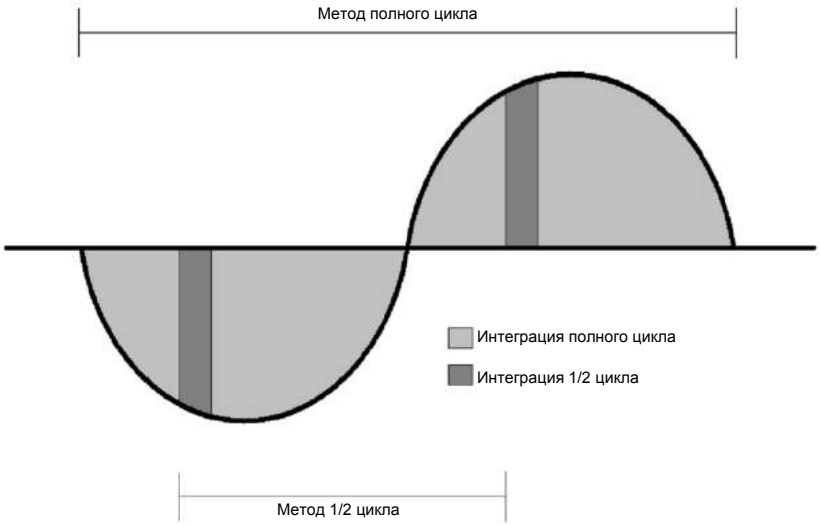
Время интеграции (мс)	***Аргумент параметра интеграции***	***Комментарии***
от 0 до 16000 мкс	***0 to 16000***	250 мкс считается высокой скоростью и нормальным минимумом
16,667 мс	***_60Hz***	Фильтрует помехи с частотой 60 Гц
20 мс	***_50Hz:***	Фильтрует помехи с частотой 50 Гц

Подавление помех в шине питания переменного тока

При питании от энергосистемы или электросети (50 или 60 Гц, 230 или 120 В перем. тока) могут возникнуть электрические помехи при целых значениях, кратных 50 или 60 Гц. Слабые сигналы аналогового напряжения, такие как в термопарах и пиранометрах, отличаются частичной уязвимостью. Измерения напряжения в CR1000 могут быть запрограммированы на подавление (фильтрацию) помех, соответствующих частоте 50 или 60 Гц. Помехи устраняются благодаря соотнесению времени интегрирования сигналов с длиной цикла помех переменного тока, как показано на рисунке *Методы подавления помех в шине переменного тока* *(стр. 316)*.

РИСУНОК. Методы подавления помех в шине питания переменного тока — 8 10 30

Рисунок 83. Методы подавления помех в шине питания переменного тока



CR1000 подавляет помехи в шине питания переменного тока для всех диапазонов напряжения, за исключением ***mV5000*** и ***mV2500***, посредством интеграции измерения ровно в один полный цикл переменного тока перед *А/Ц-преобразованием* *(стр. 507)* согласно таблице *Подавление помех переменного тока для слабых сигналов* *(стр. 317)*.

Таблица 60. Подавление помех переменного тока для слабых сигналов[1]		
Частота шины питания переменного тока	***Длительность интеграции измерения***	***Код интеграции CRBasic***
60 Гц	16,667 мс	***_60Hz***
50 Гц	20 мс	***_50Hz:***
[1] Применимо для всех входных диапазонов аналогового напряжения, кроме ***mV2500*** и ***mV5000***.		

Если выполняется подавление помех при измерении в диапазонах 2500 мВ (***mV2500***) и 5000 мВ (***mV5000***), CR1000 производит два быстрых измерения, разделенных по времени половиной линейного цикла. Половина цикла при частоте 60 Гц составляет 8333 мкс, поэтому второе измерение должно начинаться через 8333 мкс после начала интеграции первого измерения. Время А/Ц-преобразования примерно равно 170 мкс, вследствие чего максимальное время успокоения на входе составляет ок. 8160 мкс (8333 мкс – 170 мкс). Если максимальное время успокоения на входе превышено, подавление помех в шине с частотой 60 Гц не произойдет. Для подавления помех при частоте 50 Гц максимальное время успокоения на входе составляет примерно 9830 мкс (10000 мкс – 170 мкс). CR1000 не предотвращает и не выводит предупреждение при установке времени успокоения вне пределов половины цикла. В таблице Подавление помех переменного тока для сильных сигналов *(стр. 317)* перечислены подробные данные метода подавления помех в шине переменного тока для половины цикла.

Таблица 61. Подавление помех переменного тока для сильных сигналов[1]				
Частота шины питания переменного тока	***Время интеграции измерения***	***Код интеграции CRBasic***	***Время успокоения по умолчанию***	***Максимально рекомендуемое время успокоения***[2]
60 Гц	250 мкс•2	***_60Hz***	3000 мкс	8330 мкс
50 Гц	250 мкс•2	***_50Hz:***	3000 мкс	10000 мкс

[1] Применимо для входных диапазонов аналогового напряжения ***mV2500*** и ***mV5000***.

[2] Время возбуждения и время успокоения идентичны в измерениях, где требуется ток возбуждения. CR1000 не может подавать ток возбуждения на контакты **VX** в процессе А/Ц-преобразования. Метод половины цикла с током возбуждения ограничивает продолжительность рекомендуемого времени возбуждения и успокоения значениями первого измерения половины цикла. CR1000 не предотвращает и не выводит предупреждение при установке времени успокоения вне пределов половины цикла. Например, можно запрограммировать время успокоения вплоть до 50000 мкс, однако CR1000 будет производить измерения следующим образом:

1. CR1000 подает ток возбуждения, ожидает в течение 50000 мкс, а затем выполняет первое измерение.
2. В ходе А/Ц-преобразования CR1000 отключает ток возбуждения примерно на 170 мкс.
3. Ток возбуждения снова подается на половину цикла, затем выполняется второе измерение.

Иными словами, если в CR1000 запрограммировано использование методов подавления для половины цикла (50 или 60 Гц), датчик не регистрирует постоянный ток возбуждения с продолжительностью, заданной как время успокоения перед вторым измерением — если указанное время успокоения превышает половину цикла. В результате второй процесс возбуждения сокращается. В зависимости от используемого датчика сокращенный второй процесс возбуждения может вызвать погрешности измерения.

Время успокоения сигнала

Время успокоения позволяет аналоговому сигналу напряжения стабилизироваться максимально близко к истинной амплитуде до выполнения измерения. В целях минимизации погрешностей измерения успокоение сигнала применяется, когда сигнал испытывает одно или несколько указанных ниже влияний:

- небольшие внезапные скачки, вызванные внутренним мультиплексированием, соединяющим контакт CR1000 с электронной схемой измерения;

- сравнительно большие внезапные скачки, причиной которых является проводник тока возбуждения, смежный с проводником сигнала (при наличии), вследствие емкостной связи при измерении по мостовой схеме;
- диэлектрическое поглощение. Для интеграций с частотой 50 или 60 Гц требуется сравнительно большое время возврата внутреннего интеграционного конденсатора в исходное состояние перед выполнением следующего измерения.

Частота успокоения сигнала определяется константой времени успокоения на входе, являющейся функцией от исходного сопротивления и фиксированной входной емкости (3,3 nfd) CR1000.

Восходящая и нисходящая волны сигнала являются экспоненциальными. На рисунке *Входное напряжение: возрастание и затухание перехода* *(стр. 318)* показаны восходящие и нисходящие волны успокоения, стремящиеся к истинной амплитуде сигнала, V_{so}. Параметр ***SettlingTime*** команды аналогового измерения позволяет настраивать время успокоения в рамках команды измерения с разрешением в 100 мкс вплоть до 50000 мкс.

Запрограммированное время успокоения является функцией от аргументов, заданных в параметрах ***SettlingTime*** и ***Integ*** команды измерения. Комбинации аргументов и итоговые значения времени успокоения перечислены в таблице *Время успокоения для измерения CRBasic* *(стр. 318)*. Значения времени успокоения по умолчанию (если параметр ***SettlingTime = 0***) обеспечивают достаточное успокоение в большинстве случаев. Дополнительное время успокоения зачастую программируется при измерении высокорезистивных (высокоомных) датчиков, либо в случае подключения датчиков к входным клеммам с помощью длинных проводов.

Время измерения для заданной команды увеличивается по мере увеличения времени успокоения. Например, увеличение времени успокоения на 1 мс для команды с мостовым подключением и инверсией входящего сигнала и тока возбуждения приводит к увеличению времени, необходимого CR1000 на выполнение команды, на 4 мс.

Рисунок 84. Входное напряжение: возрастание и затухание перехода

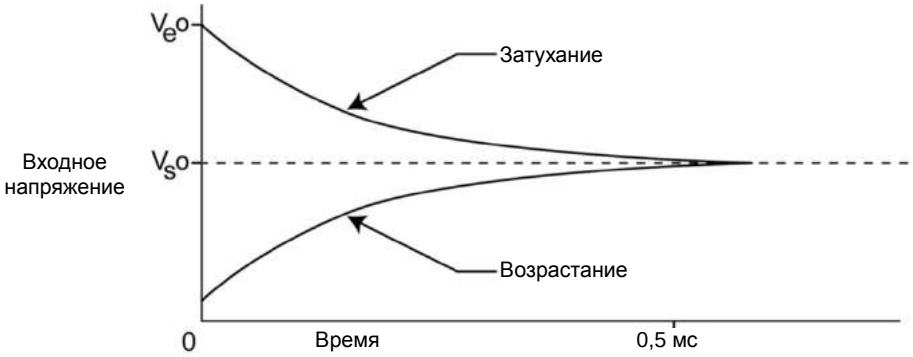


Таблица 62. Время успокоения для измерения CRBasic

SettlingTime: ***Аргумент***	Integ: ***Аргумент***	***Итоговое время успокоения***[1]
0	***250***	450 мкс
0	***_50Hz:***	3 мс
0	***_60Hz***	3 мс
целое число ≥ ***100***	целое число	мкс согласно аргументу параметра ***SettlingTime***

Таблица 62. Время успокоения для измерения CRBasic		
SettlingTime: ***Аргумент***	Integ: ***Аргумент***	***Итоговое время успокоения***[1]
[1] 450 мкс — минимальное время успокоения, требуемое для соответствия техническим характеристикам разрешения CR1000.		

Погрешности успокоения

Если для датчиков требуются длинные провода, необходимо принять во внимание следующие общие методики для минимизации погрешностей успокоения:

- Не использовать кабели с проводниками в ПВХ-изоляции. Для ПВХ характерна высокая диэлектрическая константа, увеличивающая время успокоения на входе.
- По возможности кабели тока возбуждения и сигнальные кабели должны находиться в разных оболочках, чтобы минимизировать риск внезапных скачков.
- Если скорость измерения не является приоритетом, можно выделить дополнительное время, чтобы гарантировать достаточное время успокоения. Требуемое время успокоения может быть измерено с помощью CR1000.
- В сложных случаях погрешность успокоения может быть измерена, как описано в разделе *Измерение времени успокоения* *(стр. 319)*.

Измерение времени успокоения

Время успокоения для конкретного датчика и кабеля может быть измерено с помощью CR1000. В результате программирования ряда измерений с увеличивающимися значениями времени успокоения можно будет определить, каким будет следующее значение времени успокоения при незаметном изменении измеренного напряжения. Запрограммированное время успокоения в этой точке указывает на необходимое для комбинации датчик/кабель время успокоения.

В примере CRBasic *Измерение времени успокоения* *(стр. 319)* представлен код CRBasic, помогающий определить время успокоения для датчика давления с использованием высокоемкостного полупроводника. Код состоит из ряда измерений, выполненных по схеме полного моста (**BrFull()**), с увеличивающимся временем успокоения. Датчик давления находится в устойчивом состоянии, так что изменения в измеренном напряжении могут быть обусловлены временем успокоения, а не изменениями давления. Раздел *Программирование* *(стр. 122)* может помочь в понимании примера кода CRBasic.

Первые шесть измерений показаны в таблице *Первые шесть значений данных времени успокоения* *(стр. 321)*. Каждая линия на рисунке *Время успокоения для датчика давления* *(стр. 321)* содержит все двадцать значений **PT()** mV/V (левая ось) для заданного номера записи, а также среднее значение, с помощью которого измерения выражаются в качестве процентной доли от итогового показания (правая ось). Показание установилось на уровне 99,5 % от итогового значения к четырнадцатому измерению, что отражено переменной PT(14). Это допустимая погрешность для приложения, так что время успокоения, равное 1400 мкс, определяется как приемлемое.

CRBasic: пример 64. Измерение времени успокоения

```
'Этот пример программы демонстрирует измерение времени успокоения посредством
'последовательного повторения отдельного измерения несколько раз. В данном случае программа
'измеряет температуру коммутационной панели CR1000.

Public RefTemp 'Объявление переменной для получения команды.

BeginProg
   Scan(1,Sec,3,0)
      PanelTemp(RefTemp, 250) 'Команда для выполнения измерения
   NextScan
EndProg'измеряет время успокоения датчика на основании дифференциального
'измерения напряжения

Public PT (20)                                  'Переменная для сохранения измерений

DataTable(Settle,True,100)
   Sample(20,PT(),IEEE4)
EndTable

BeginProg
   Scan(1,Sec,3,0)

      BrFull(PT(1),1,mV7.5,1,Vx1,2500,True,True,100, 250,1.0,0)
      BrFull(PT(2),1,mV7.5,1,Vx1,2500,True,True,200, 250,1.0,0)
      BrFull(PT(3),1,mV7.5,1,Vx1,2500,True,True,300, 250,1.0,0)
      BrFull (PT(4),1,mV7.5,1,Vx1,2500,True,True,400, 250,1.0,0)
      BrFull (PT(5),1,mV7.5,1,Vx1,2500,True,True,500, 250,1.0,0)
      BrFull (PT(6),1,mV7.5,1,Vx1,2500,True,True,600, 250,1.0,0)
      BrFull (PT(7),1,mV7.5,1,Vx1,2500,True,True,700, 250,1.0,0)
      BrFull (PT(8),1,mV7.5,1,Vx1,2500,True,True,800, 250,1.0,0)
      BrFull (PT(9),1,mV7.5,1,Vx1,2500,True,True,900, 250,1.0,0)
      BrFull (PT(10),1,mV7.5,1,Vx1,2500,True,True,1000, 250,1.0,0)
      BrFull (PT(11),1,mV7.5,1,Vx1,2500,True,True,1100, 250,1.0,0)
      BrFull (PT(12),1,mV7.5,1,Vx1,2500,True,True,1200, 250,1.0,0)
      BrFull (PT(13),1,mV7.5,1,Vx1,2500,True,True,1300, 250,1.0,0)
      BrFull (PT(14),1,mV7.5,1,Vx1,2500,True,True,1400, 250,1.0,0)
      BrFull (PT(15),1,mV7.5,1,Vx1,2500,True,True,1500, 250,1.0,0)
      BrFull (PT(16),1,mV7.5,1,Vx1,2500,True,True,1600, 250,1.0,0)
      BrFull (PT(17),1,mV7.5,1,Vx1,2500,True,True,1700, 250,1.0,0)
      BrFull (PT(18),1,mV7.5,1,Vx1,2500,True,True,1800, 250,1.0,0)
      BrFull (PT(19),1,mV7.5,1,Vx1,2500,True,True,1900, 250,1.0,0)
      BrFull (PT(20),1,mV7.5,1,Vx1,2500,True,True,2000, 250,1.0,0)

      CallTable Settle

   NextScan
EndProg
```

Рисунок 85. Время успокоения для датчика давления

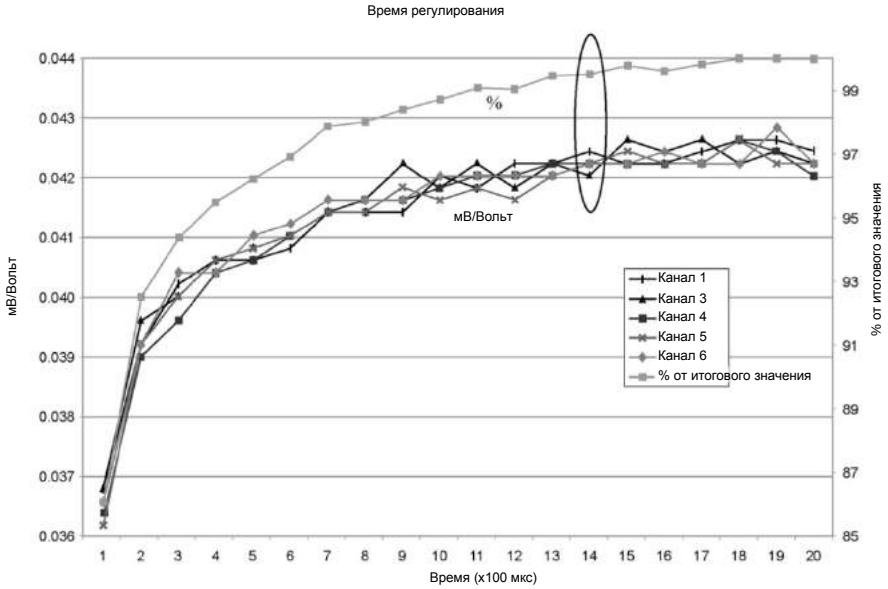


Таблица 63. Первые шесть значений данных времени успокоения

TIMESTAMP	*REC*	*PT(1)*	*PT(2)*	*PT(3)*	*PT(4)*	*PT(5)*	*PT(6)*
		Smp	*Smp*	*Smp*	*Smp*	*Smp*	*Smp*
1/3/2000 23:34	0	0.03638599	0.03901386	0.04022673	0.04042887	0.04103531	0.04123745
1/3/2000 23:34	1	0.03658813	0.03921601	0.04002459	0.04042887	0.04103531	0.0414396
1/3/2000 23:34	2	0.03638599	0.03941815	0.04002459	0.04063102	0.04042887	0.04123745
1/3/2000 23:34	3	0.03658813	0.03941815	0.03982244	0.04042887	0.04103531	0.04103531
1/3/2000 23:34	4	0.03679027	0.03921601	0.04022673	0.04063102	0.04063102	0.04083316

Обнаружение размыкания на входе

Примечание. Информация, приведенная в следующих разделах, по большей части является сугубо технической и не требуется для обычной эксплуатации CR1000. Эта информация включена с целью обеспечить понимание функции обнаружения размыкания на входе в CR1000.

Краткое описание

• В CR1000 доступна опция обнаружения размыкания на входе, например поврежденного датчика или слабого соединения.
• Опция выбирается путем добавления ***C*** к коду диапазона.
• При использовании этой опции результат измерения на разомкнутом соединении будет **NAN** (не является числом).

Полезной опцией, доступной для несимметричного и дифференциального измерения, является обнаружение размыкания на входе вследствие поврежденного или отсоединенного кабеля датчика. Так предотвращаются не выявляемые иным способом погрешности измерения. Коды диапазона с ***C*** на конце позволяют обнаруживать размыкание на входе для всех входных диапазонов, кроме ±5000 мВ (см. таблицу *Входные диапазоны аналогового напряжения с CMN / OID* *(стр. 309)*).

Результатом добавления в конец кода диапазона (***Range***) символа ***C*** становится 50-микросекундное внутреннее подключение входа V+ PGIA к источнику высокого перенапряжения. Вход V– подключается к заземлению. После отключения входов истинный входящий сигнал может прийти в состояние успокоения, и измерение выполняется в нормальном режиме. Если подключен связанный датчик, измеряется напряжение сигнала. Если вход (дифференциальный) разомкнут, измерение выйдет за рамки диапазона, поскольку поданное перенапряжение будет до сих пор присутствовать на входе, а в качестве результата будет выведено значение **NAN.**

Коды диапазона и допустимые амплитуды перенапряжения приведены в таблице *Опция кода диапазона: C, перенапряжение* *(стр. 322)***.**

Опция ***C*** может не работать или работать некорректно в следующих случаях:

- Если контур на входе не является действительно разомкнутым, как это может произойти в случае мокрого обрезанного конца кабеля, размыкание контура может остаться необнаруженным, поскольку входной конденсатор сбрасывает напряжение до нормального посредством внешней утечки в землю в пределах заданного для измерения времени успокоения. Проблема усугубляется, если выбрано более долгое время успокоения, поскольку входному концентратору требуется больше времени, чтобы разрядиться до «нормального» уровня.
- Если разомкнутый контур находится на конце очень длинного кабеля, тестовый импульс (300 мВ) может не создать в кабеле (имеющем высокую емкость) достаточное напряжение, чтобы сгенерировать результат «NAN» или определенное напряжение сигнала ошибки. Кроме того, кабель может сыграть роль антенны и спровоцировать помехи, которые также не будут считываться как напряжение сигнала ошибки.
- Датчик может «противиться» воздействию тестового импульса при подключении в выходу, даже на 100 мкс. Риск повреждения мал или отсутствует, однако на выходе датчика могут быть созданы временные колебания. Программирование более долгого времени успокоения для команды измерения CRBasic, позволяющего колебаниям затухнуть, прежде чем будет выполнено А/Ц-преобразование, может минимизировать проблему.

Таблица 64. Опция кода диапазона: C, перенапряжение

Диапазон входного тока	***Перенапряжение***
±2,5 мВ ±7,5 мВ ±25 мВ ±250 мВ	300 мВ
±2500 мВ	***C***-опция с пояснением[1]
±5000 мВ	***C***-опция недоступна

[1] При использовании ***C*** на клемму H кратковременно подается напряжение выше 2500 мВ, в то время как клемма L остается подключенной к земле. Итоговое синфазное напряжение составляет 1250 мВ, что недопустимо для нулевого остаточного синфазного напряжения, однако приемлемо в целях упрощения обнаружения типа размыкания на входе. Для этого требуется включение оператора **If / Then / Else** в программу CRBasic, что позволит протестировать результаты измерения. Например:

• Результат измерения **VoltDiff()** с использованием значения ***mV2500C*** в качестве кода диапазона (***Range***) может быть проверен на индикацию размыкания на входе при результате > 2500 мВ.

• Результат измерения **BrHalf()**, **X**, может быть проверен с использованием кода диапазона ***mV2500C*** на наличие значений >***1***. Результат **X > 1** указывает на размыкание на входе при первичном измерении, V1, где X = V1/Vx, а Vx — напряжение возбуждения. Аналогичная стратегия может использоваться для других измерений с мостовой схемой.

Компенсация напряжения смещения

Связанные темы
• *Автоматическая калибровка — общие сведения (стр. 92)*
• *Автоматическая калибровка — подробная информация (стр. 344)*
• *Автоматическая калибровка — ошибки (стр. 490)*
• *Компенсация напряжения смещения (стр. 323)*
• *Заводская калибровка (стр. 94)*
• *Заводская калибровка или процедуры ремонта (стр. 476)*

Краткое описание
Напряжение смещения при измерении неизбежно, однако может быть минимизировано.

Напряжение смещения возникает ввиду следующих причин:
• Токи заземления
• Прямой термоэлектрический эффект
• Остаточное напряжение после предыдущего измерения

Способы решения проблемы:
• Подключение заземления к клеммам заземления (**G**)
• Использование инверсии входящего сигнала (***RevDiff* = *True***) при дифференциальных измерениях
• Автоматическая компенсация смещения при дифференциальных измерениях, если ***RevDiff= False***
• Автоматическая компенсация смещения при несимметричных измерениях, если ***MeasOff* = *False***
• Улучшенная компенсация смещения, если ***MeasOff* = *True***
• Инверсия тока возбуждения (***RevEx* = *True***)
• Увеличенное время успокоения

Смещение напряжения может стать причиной существенной погрешности. Например, смещение на 3 мкс при сигнале в 2500 мВ вызывает погрешность в 0,00012 %, однако аналогичное смещение при сигнале в 0,25 мВ вызывает погрешность 1,2 %. Первичными источниками смещения напряжения являются токи заземления и прямой термоэлектрический эффект.

Несимметричные измерения уязвимы к перепадам напряжения на клемме заземления, вызванным возвратными токами от другого устройства, запитанного от коммутационной панели CR1000, такого как телекоммуникационный модем стороннего производителя или датчик, потребляющий значительное количество энергии. Токи > 5 мА, как правило, нежелательны. Ошибки можно избежать посредством назначения заземлений от этих сторонних устройств клемме заземления **G** на коммутационной панели CR1000 вместо использования клеммы заземления сигнала (⏚). Токи заземления могут быть вызваны вследствие возбуждения резистивных мостовых датчиков, однако в таком случае обычно не возникает погрешность смещения. Эти токи, как правило, перемещаются, только если подается напряжение возбуждения. Возвратные токи, связанные с напряжением возбуждения, не могут воздействовать на другие несимметричные измерения, поскольку ток возбуждения обычно отключается перед тем, как CR1000 переходит к следующему измерению. Тем не менее, если программа CRBasic составлена таким образом, что клемма возбуждения активируется при несвязанном измерении слабого напряжения, может возникнуть погрешность смещения.

Прямой диэлектрический эффект приводит к появлению слабого термоиндуцированного напряжения на стыках разнородных металлов, что типично для электронных устройств. Дифференциальные измерения менее подвержены этому воздействию (по сравнению с несимметричными) благодаря пассивному гашению напряжения между сопоставленными парами высокого и низкого уровня, такими как **1H/1L**. Потому дифференциальный метод следует использовать при измерении критического низкоуровневого напряжения (особое внимание уделяется напряжению менее 200 мВ), например на выходе пиранометров и термопар. Дифференциальные измерения

обладают еще одним преимуществом — опцией инверсии входящего сигнала, ***RevDiff***. Если параметр ***RevDiff*** имеет значение ***True*** (истина), выполняются два дифференциальных измерения, причем первое имеет положительную полярность, а второе инвертируется. При вычитании измерений с противоположной полярностью сбрасываются некоторые из значений напряжения смещения, связанных с измерением.

При несимметричных и дифференциальных измерениях без инверсии входящего сигнала используется измерение напряжения смещения с заземлением входов PGIA. В случае дифференциальных измерений без инверсии входящего сигнала это измерение напряжения смещения выполняется как часть обычной процедуры автокалибровки CR1000. В рамках команд несимметричного измерения **VoltSE()** и **TCSe()** параметр ***MeasOff*** определяет, выполняется ли измерение напряжения смещения в начале команды или как часть внутренней калибровки. Эта опция предоставляет возможность взвешивания скорости измерения применительно к погрешности измерения. Если параметр ***MeasOff = True***, измерение несимметричного напряжения смещения производится в начале команды **VoltSE()**. Если параметр ***MeasOff = False***, измерение напряжения смещения выполняется в процессе внутренней калибровки. В случае напряжения смещения с медленными колебаниями выбор параметра ***MeasOff = True*** для команды **VoltSE()** приводит к улучшению эффективности напряжения смещения.

Для возбуждения датчика в процессе логометрического измерения используется напряжение или ток возбуждения. При инверсии полярности возбуждения также сокращается погрешность напряжения смещения. При установке для параметра ***RevEx*** значения ***True*** (истина) для измерения на программном уровне задается инверсия возбуждения. Инверсия возбуждения приводит к изменению полярности измеренного напряжения, так что два измерения с противоположной полярностью могут быть вычтены и разделены на 2, чтобы сократить смещение аналогичным образом, как при инверсии входящего сигнала для дифференциальных измерений. Команды логометрического дифференциального измерения позволяют устанавливать для параметров ***RevDiff*** и ***RevEx*** значение ***True***. В результате получаются четыре последовательности измерений:

- положительная полярность возбуждения с положительной полярностью дифференциального входа
- отрицательная полярность возбуждения с положительной полярностью дифференциального входа
- положительная полярность возбуждения с отрицательной полярностью дифференциального входа
- положительная полярность возбуждения, а затем отрицательная полярность возбуждения дифференциального входа

В случае логометрических несимметричных измерений, например **BrHalf(),** установка параметра ***RevEx = True*** приводит к вычитанию и делению пополам двух измерений с противоположной полярностью возбуждения в целях снижения напряжения смещения. Если при логометрическом несимметричном измерении установлен параметр ***RevEx = False***, измерение напряжения смещения производится в процессе внутренней калибровки.

Если в рамках команды несимметричного измерения производится измерение последовательности сигналов аналогового напряжения, как это, например, происходит, когда для **VoltSE()** запрограммировано значение ***Reps = 2*** или выше, измерения на последовательных клеммах могут испытать влияние смещения, амплитуда которого является функцией напряжения предыдущего измерения. В то время как это смещение обычно крайне мало и незаметно при измерении сильных сигналов, для измерения очень слабых сигналов это может стать причиной существенной погрешности или результата **NAN**. Этот эффект вызывается диэлектрическим поглощением интегрирующего конденсатора и не может быть устранен посредством конструкции контура. Возможные способы решения проблемы:

- Программирование увеличенного времени успокоения.
- Использование отдельной команды для каждой входной клеммы, в результате чего интегрирующий контур возвращается в исходное состояние перед интеграцией.
- Предотвращение расположения входа с очень высоким напряжением перед входом с очень низким напряжением в последовательности измерений, если используется команда отдельного измерения.

В таблице *Опции компенсации напряжения смещения* *(стр. 325)* перечислены некоторые из доступных инструментальных средств, помогающих минимизировать воздействие напряжения смещения.

Таблица 65. Опции компенсации напряжения смещения

Команда измерения CRBasic	***Инверсия входящего сигнала*** (RevDiff =True)	***Инверсия тока возбуждения*** (RevEx = True)	***Определение смещения в процессе измерения*** (MeasOff = True)	***Измерение смещения в процессе фоновой калибровки*** (RevDiff = False) (RevEx = False) (MeasOff = False)
VoltDiff()	✓			✓
VoltSe()			✓	✓
TCDiff()	✓			✓
TCSe()			✓	✓
BrHalf()		✓		✓
BrHalf3W()		✓		✓
Therm107()		✓		✓
Therm108()		✓		✓
Therm109()		✓		✓
BrHalf4W()	✓	✓		✓
BrFull()	✓	✓		✓
BrFull6W()	✓	✓		✓
AM25T()	✓	✓		✓

Инверсия входящего сигнала и тока возбуждения

При инверсии входящих сигналов (дифференциальные измерения) или полярности напряжения возбуждения (мостовые измерения) сбрасывается смещение паразитного напряжения. Например, если в электрической схеме измерения присутствует смещение на 3 мкс, сигнал с напряжением 5 мВ измеряется как 5,003 мВ. При инверсии входящего сигнала или тока возбуждения второе вспомогательное измерение выполняется для 4,997 мВ. После вычитания второго вспомогательного измерения из первого и деления результата на 2 смещение устраняется:

```
5.003 mV - (-4.997 mV) = 10.000 mV
10.000 mV / 2 = 5.000 mV
```

Когда CR1000 инвертирует полярность дифференциальных входящих сигналов или тока возбуждения, после инверсии возникает задержка, равная времени успокоения перед первым вспомогательным измерением. Таким образом, при использовании ***RevDiff*** или ***RevEx*** возникают две задержки в одном измерении. Если оба параметра — ***RevDiff*** и ***RevEx*** — имеют значение ***True***, выполняется четыре вспомогательных измерения: положительный и отрицательный ток возбуждения с входящими сигналами, направленными сначала в одну, а затем в обратную сторону. Автоматическая процедура выглядит следующим образом:

1. Переключение на клеммы, задействованные в измерении
2. Настройка тока возбуждения, успокоение, затем **измерение**

3. Инверсия возбуждения, успокоение, затем **измерение**
4. Инверсия возбуждения, инверсия входящих клемм, успокоение, **измерение**
5. Инверсия возбуждения, успокоение, **измерение**

На одно **измерение** приходится четыре задержки. CR1000 обрабатывает четыре вспомогательных измерения в рамках одного отчетного измерения. В случае инверсии тока возбуждения время возбуждения для каждого из вариантов полярности в точности совпадает; это гарантирует, что ионные датчики не будут поляризоваться при повторных измерениях.

Читать далее. Технический документ «Преимущества инверсии входящих сигналов и тока возбуждения для измерений напряжения» доступен на сайте *www.campbellsci.com.*

Напряжение смещения опорного заземления

Если включен параметр ***MeasOff*** (= ***True***), CR1000 измеряет напряжение смещения опорного заземления перед выполнением каждого измерения **VoltSe()** или **TCSe()**. Это напряжение смещения вычитается из последующего измерения.

Фоновая калибровка

Если параметры ***RevDiff, RevEx*** или ***MeasOff*** отключены (= ***False***), компенсация напряжения смещения продолжает выполняться автоматически, хотя и с меньшей эффективностью, благодаря использованию измерений из автоматической фоновой калибровки. Деактивация параметров ***RevDiff, RevEx*** или ***MeasOff*** ускоряет измерение; однако увеличение скорости достигается за счет снижения точности, поскольку:

1 ***RevDiff, RevEx*** и ***MeasOff*** обладают большей эффективностью.

2 Фоновая калибровка выполняется только периодически, так что между смещениями фоновой калибровки и измерениями, к которым они применяются, возникает больше временных сдвигов.

Примечание. Если длительность измерений должна быть минимальной, чтобы максимизировать частоту измерений, рекомендуется отключить параметры ***RevDiff, RevEx*** и ***MeasOff***, особенно при слабой предрасположенности температуры модуля CR1000 и возвратных токов к изменению.

Временной сдвиг между измерениями

Временной сдвиг между последующими измерениями напряжения является функцией от времени успокоения и интеграции, А/Ц-преобразования и числа, указанного в качестве параметра Reps в команде VoltDiff() или VoltSE(). Достаточно хорошее приближение:

временной сдвиг = время успокоения + время интеграции + время А/Ц-преобразования[1] + reps[2]

[1] Время А/Ц-преобразования, равное 15 мкс.

[2] Если ***Reps > 1*** (несколько измерений в рамках одной команды), дополнительное время не требуется. Если ***Reps = 1*** в последовательных командах для напряжения, добавляется по 15 мкс на команду.

8.1.2.2. Измерения термопар — подробная информация

Связанные темы:
• Измерения термопар — подробная информация
• Измерения термопар — команды

Измерения термопар представляют собой частный случай измерения напряжения.

Примечание. Термопары недороги и просты в эксплуатации. Тем не менее они могут вызвать затруднения в процессе получения точных температурных данных, особенно если используются внешние свободные спаи. Campbell Scientific **настоятельно рекомендует** внимательно ознакомиться с разделом *Анализ погрешностей* *(стр. 327)***.** Введение в измерения термопар находится в разделе *Практическое упражнение: измерение термопары* *(стр. 46)***.**

Микровольтовое разрешение CR1000 в сочетании с функциональной возможностью измерения напряжения с низким уровнем помех отлично подходят для измерения термопар. Термопара состоит из двух проводов, выполненных из разных металлов или сплавов и соединенных на одном из концов с целью образования измерительного спая. На противоположном конце каждый кабель подключается к клеммам устройства измерения напряжения, например CR1000. Эти соединения представляют собой свободный спай. Если температура обоих спаев (измерительного и свободного) различается, в проводах возникает напряжение, прямо пропорциональное этой разнице. Этот феномен называется прямым термоэлектрическим эффектом. В результате измерения напряжения между положительным и отрицательным контактами устройства измерения напряжения получается непосредственное значение температурной разницы между измерительным и свободным спаями. Наличие третьего металла (например, припоя или контактов CR1000) между двумя проводами из разнородных металлов приводит к созданию паразитного спая термопары, влияние которого прекращается при достижении обоими проводами одинаковой температуры. Как следствие, оба провода свободного спая располагаются в непосредственной близости друг от друга, чтобы их температура оставалась одинаковой. Информация о температуре свободного спая позволяет определить напряжение компенсации свободного спая в соответствии с температурной разницей между свободным спаем и 0 °C. Это напряжение компенсации может использоваться в сочетании с измеренным напряжением термопары для расчета абсолютной температуры спая термопары. В целях упрощения измерений термопар в коммутационную панель CR1000 встроен терморезистор, позволяющий измерять температуру свободного спая с помощью команды **PanelTemp().**

Команды термопары TCDiff() и **TCSe()** определяют температуру термопары, применяя следующую последовательность действий. Сначала определяется температура (°C) свободного спая. Затем рассчитывается напряжение компенсации свободного спая на основании температурной разницы между свободным спаем и 0 °C. Если свободный спай располагается на контактах аналогового входа CR1000, температура без проблем измеряется с помощью команды **PanelTemp()**. Измеряется фактическое напряжение термопары, которое затем объединяется с напряжением компенсации свободного спая. Результат используется для определения температуры спая термопары по методу полиномиального приближения значений калибровки термопар NIST.

8.1.2.2.1. Анализ погрешностей термопары

Погрешность измерения температуры термопары складывается из суммы погрешностей измерения температуры свободного спая и погрешности аппроксимации значений температуры к напряжению с помощью полинома, неидеальной природы самой термопары (отклонение от стандартов, опубликованных в Монографии NIST 175), погрешности измерения напряжения термопары, а также погрешности аппроксимации значений

напряжения к температуре с помощью полинома (разница между стандартом NIST и полиномиальными приближениями CR1000). Приведенное ниже описание погрешностей ограничивается только погрешностями калибровки и измерения, не включая погрешности, возникающие при установке или сопоставлении датчика и типа термопары с измеряемой средой.

Погрешность температуры панели

Терморезистор температуры панели (Betatherm 10K3A1A) находится непосредственно под панелью в середине двух рядов контактов аналогового входа. Согласно техническим характеристикам, он обладает совместимостью с точностью до 0,1 °C для диапазона температур от 0 до 70 °C. Ниже точки замерзания и при более высоких температурах характеристики снижаются. Учитывая возможную погрешность при измерении дополняющего резистора, а также уравнение Стейнхарта-Харта, используемое для вычисления температуры на основании сопротивления, погрешность измерения температуры панели рассчитывается, как показано на рисунке *Сводка по погрешностям температуры панели* *(стр. 329)***.** В общих словах, погрешность оценивается как ± 0,1 °C в диапазоне от 0 до 40 °C, ± 0,3 °C в диапазоне от –25 до 50 °C и ± 0,8 °C в диапазоне от –55 до 85 °C.

Погрешность измерения опорной температуры складывается из погрешности измерения температуры терморезистора и разницы температур между терморезистором панели и клеммами, к которым подключена термопара. При выполнении измерений термопары необходимо всегда использовать крышку клеммной колодки. Она изолирует контакты от сквозного воздуха и быстрых перепадов температуры, а также проводимого тепла в целях снижения градиентов температуры. В типичной конструкции, когда CR1000 находится в атмосферозащищенном кожухе, не подвергающемся резким перепадам температуры или неравномерному воздействию солнечных лучей, разница температур между контактами и терморезистором составляет менее 0,2 °C.

При внешнем воздействии градиент температур на входной панели может быть значительно хуже. К примеру, CR1000 находится в камере с регулируемой температурой. Термопары измерили температуру изолированной алюминиевой планки, расположенной вне камеры, на концевых клеммах и в середине каждой аналоговой клеммной колодки. Температура этой планки также была измерена другим регистратором данных. Разница между температурой, измеренной с помощью одной из термопар, и фактической температурой планки обусловлена температурной разницей между клеммами, к которым подключена термопара, и опорной температурой терморезистора (показатели скорректированы с учетом погрешности терморезистора). На рисунке *Градиент температуры панели (от низкой температуры до высокой)* *(стр. 329)* представлены погрешности после изменения температуры камеры с низкой на высокую приблизительно в течение 15 минут. На рисунке *Градиент температуры панели (от высокой температуры до низкой)* *(стр. 330)* показаны результаты снижения температуры. При резких перепадах температуры показания терморезистора обычно отстают от температуры клемм, поскольку он расположен глубже внутри CR1000.

Рисунок 86. Сводка по погрешностям температуры панели

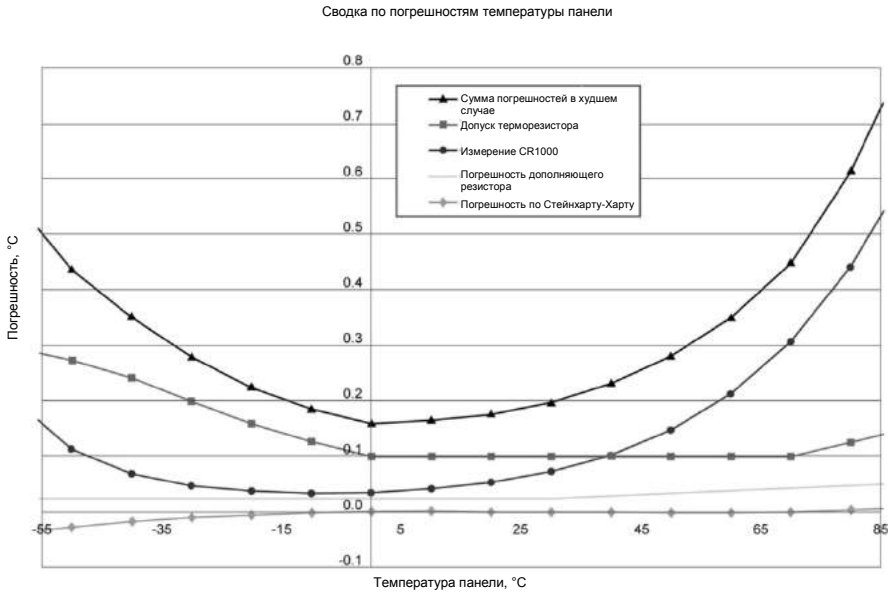


Рисунок 87. Градиенты температуры панели (от низкой температуры до высокой)

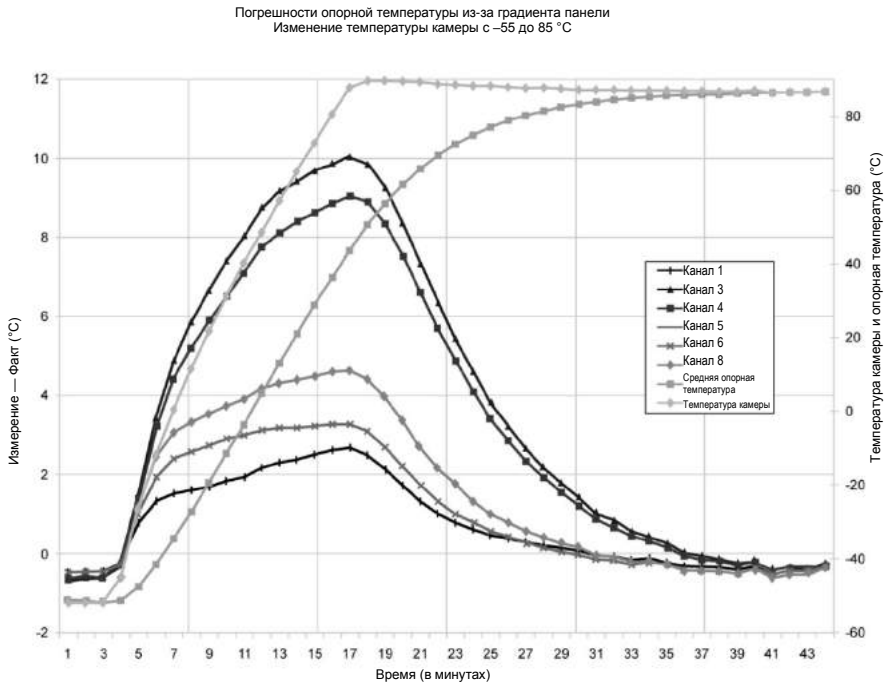
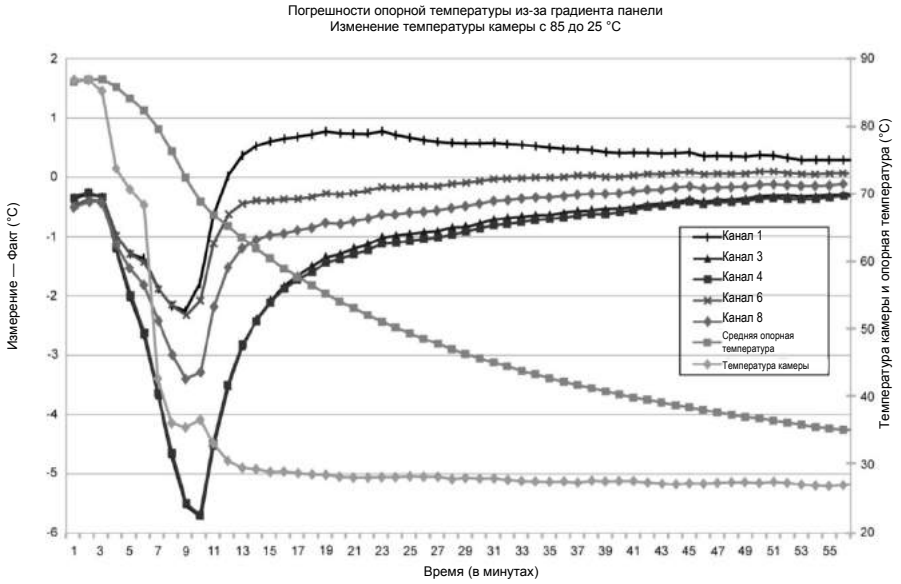


Рисунок 88. Градиенты температуры панели (от высокой температуры до низкой)



Пределы погрешности термопары

Стандартом, в котором выходное напряжение термопары представлено в виде функции от температуры (свободный спай при 0 °C), является Монография 175 Национального института стандартов и технологии (NIST, 1993 г.). Американский национальный институт стандартов (ANSI) установил пределы погрешности для проводов термопары, принятые в качестве отраслевого стандарта (ANSI MC 96.1, 1975 г.). В таблице *Пределы погрешности для провода термопары* *(стр. 331)* представлены установленные ANSI пределы погрешности для стандартных и специальных проводов термопары допустимых для CR1000 типов.

Когда температура обоих спаев термопары равна, напряжение, согласно закону промежуточных металлов, не генерируется. Как следствие, в термопаре не может быть погрешности смещения; любое отклонение от стандарта (предполагается, что провода однородны, и нет дополнительных спаев) обусловлено погрешностью наклона. Учитывая это, постоянные температурные ограничения погрешности (например, ±1,0 °C для типа T в отличие от погрешности наклона, равной 0,75 % от температуры) в приведенной выше таблице очевидно выше, чем привычные значения при определении температуры в диапазоне окружающей среды (т. е. свободный спай при 0 °C показывает относительно близкие значения к измеряемой температуре, так что абсолютная погрешность — произведение температурной разницы и погрешности наклона — должна быть ближе к процентной погрешности, а не к фиксированной). Аналогично: поскольку погрешность калибровки термопары представляет собой погрешность наклона, точность может возрасти в случае приближения температуры свободного спая к температуре измерения. По той же причине дифференциальные измерения температуры в рамках небольшого градиента температуры могут быть крайне точными.

Чтобы выполнить количественную оценку погрешности термопары, когда значения свободного спая не зафиксированы на отметке 0 °C, необходимы пределы погрешности для коэффициента прямого термоэлектрического эффекта (переход напряжения термопары / температурная кривая) для различных термопар. При отсутствии данной информации рациональным подходом будет применение процентной погрешности (с возможным добавлением 0,25 %) к разнице температуры, измеряемой термопарой.

Таблица 66. Пределы погрешности для провода термопары (свободный спай при 0 °C)			
Термопара	***Температура***	***Пределы погрешности (выбирается большее значение)***	
Тип	***Диапазон, °C***	***Стандартный***	***Особая***
T	от –200 до 0	± 1,0 °C или 1,5 %	
	от 0 до 350	± 1,0 °C или 0,75 %	± 0,5 °C или 0,4 %
J	от 0 до 750	± 2,2 °C или 0,75 %	± 1,1 °C или 0,4 %
E	от –200 до 0	± 1,7 °C или 1,0 %	
	от 0 до 900	± 1,7 °C или 0,5 %	± 1,0 °C или 0,4 %
K	от –200 до 0	± 2,2 °C или 2,0 %	
	от 0 до 1250	± 2,2 °C или 0,75 %	± 1,1 °C или 0,4 %
R или S	от 0 до 1450	± 1,5 °C или 0,25 %	± 0,6 °C или 0,1 %
B	от 800 до 1700	± 0,5 %	не установлено

Погрешность измерения напряжения термопары

Выходные значения термопары крайне малы — от 10 до 70 мкс на °C. Если при программировании не используются входные диапазоны с высоким разрешением, точность и чувствительность CR1000 заметно снижаются. В таблице *Диапазон напряжения для максимального разрешения термопары (стр. 331)* перечислены диапазоны высокого разрешения, доступные для различных типов термопар и температурных диапазонов. Следующие четыре примера расчетов погрешности входа термопары иллюстрируют принцип воздействия выбранного диапазона входного напряжения на точность измерений. На рисунке *Расчет погрешности на входе (стр. 332)* показано, откуда поступают различные значения для выполнения расчета. Более подробную информацию о точности измерений и расчетах погрешности см. в разделе «Точность измерения».

Если измерительный спай термопары находится в электрическом контакте с измеряемым объектом (или имеет возможность установить такой контакт), во избежание образования петли заземления следует выполнять дифференциальное измерение.

Таблица 67. Диапазон напряжения для максимального разрешения термопары				
Опорная температура при 20 °C				
Тип термопары и температурный диапазон (°C)	***Температурный диапазон (°C) для входного диапазона ±2,5 мВ***	***Температурный диапазон (°C) для входного диапазона ±7,5 мВ***	***Температурный диапазон (°C) для входного диапазона ±25 мВ***	***Температурный диапазон (°C) для входного диапазона ±250 мВ***
T: от -270 до 400	от -45 до 75	от -270 до 180	от -270 до 400	не используется
E: от -270 до 1000	от -20 до 60	от -120 до 130	от -270 до 365	> 365
K: от -270 до 1372	от -40 до 80	от -270 до 200	от -270 до 620	> 620
J: от -210 до 1200	от -25 до 65	от -145 до 155	от -210 до 475	> 475
B: от -0 до 1820	от 0 до 710	от 0 до 1265	от 0 до 1820	не используется

Таблица 67. Диапазон напряжения для максимального разрешения термопары				
Опорная температура при 20°C				
Тип термопары и температурный диапазон (°C)	***Температурный диапазон (°C) для входного диапазона ±2,5 мВ***	***Температурный диапазон (°C) для входного диапазона ±7,5 мВ***	***Температурный диапазон (°C) для входного диапазона ±25 мВ***	***Температурный диапазон (°C) для входного диапазона ±250 мВ***
R: от -50 до 1768	от -50 до 320	от -50 до 770	от -50 до 1768	не используется
S: от -50 до 1768	от -50 до 330	от -50 до 820	от -50 до 1768	не используется
N: от -270 до 1300	от -80 до 105	от -270 до 260	от -270 до 725	> 725

Рисунок 89. Расчет погрешности на входе

Особенности измерения термопары

Условия:
Температура = 45 °C
Опорная температура = 25 °C
Разность температур = 20 °C
Выходной множитель при 45 °C = 42,4 мкВ $°C^{-1}$
Выход термопары = 20 °C * 42,4 мкВ $°C^{-1}$ = 830,7 мкВ

Технические характеристики CR1000

ДИАПАЗОНЫ И РАЗРЕШЕНИЕ. Базовое разрешение (Basic Res) — это разрешающая способность отдельного А/Ц-преобразования. Разрешение дифференциальных измерений с инверсией входящего сигнала составляет половину от базового.

Входной диапазон (мВ)[1]	Дифференциальное разрешение (мкВ)[2]	Базовое разрешение (мкВ)
±5000	667	1333
±2500	333	667
±250	33.3	66.7
±25	3.33	6.7
±7.5	1.0	2.0
±2.5	0.33	0.67

[1] Запас, примерно равный 9%, предусмотрен для всех диапазонов, и за счет этого гарантируется, что значения, разбросанные по всему диапазону, не выйдут за его пределы.
[2] Разрешение для дифференциального измерения с инверсией входящего сигнала.

ПОГРЕШНОСТЬ[3]:
±(0,06% от показаний + смещение), от 0° до 40°C
±(0,12% от показаний + смещение), от 25° до 50°C
±(0,18% от показаний + смещение), от 40° до 85°C (только -XT)

[3] Погрешность не учитывает помехи датчика и помехи при измерении. Смещения определяются следующим образом:

Смещение для дифференциального измерения с инверсией входящего сигнала = 1,5·базовое разрешение +1,0 мкВ
Смещение для дифференциального измерения без инверсии входящего сигнала = 3·базовое разрешение +2,0 мкВ
Смещение для несимметричного измерения = 3·базовое разрешение + 3,0 мкВ

Пример 1. Расчет погрешности на входе

Погрешность (мкВ) = **компонент усиления** + компонент смещения
= (830,7 мкВ *0,12%) + (1,5 * 0,67 мкВ + 1,0 мкВ)
= 0,997 мкВ + 2,01 мкВ
= 3,01 мкВ (= 0,071 °C)

Примеры погрешности на входе: термопара типа T при 45 °C

Данные примеры показывают, что в температурном диапазоне окружающей среды погрешность смещения на входе значительно выше, чем погрешность усиления на входе, поскольку используется небольшой входной диапазон.

Условия:

Температура модуля CR1000, от –25 до 50 °C

Температура = 45 °C

Опорная температура = 25 °C

Разность температур = 20 °C

Выходной множитель термопары при 45 °C = 42,4 мкВ $°C^{-1}$

Выход термопары = 20 °C • 42,4 мкВ $°C^{-1}$ = 830,7 мкВ

Входной диапазон = ±2,5 мВ

Расчет погрешностей с инверсией входящего сигнала = True

Погрешность (мкВ) = компонент усиления + компонент смещения

= (830,7 мкВ • 0,12%) + (1,5 • 0,67 мкВ + 1,0 мкВ)

= 0,997 мкВ + 2,01 мкВ

= 3,01 мкВ (= 0,071 °C)

Расчет погрешностей с инверсией входящего сигнала = False

Погрешность (мкВ) = компонент усиления + компонент смещения

= (830,7 мкВ • 0,12%) + (3 • 0,67 мкВ + 2,0 мкВ)

= 0,997 мкВ + 4,01 мкВ

= 5,01 мкВ (= 0,12 °C)

Примеры погрешности на входе: термопара типа K при 1300 °C

Погрешность температуры, обусловленная неточностью измерения напряжения термопары, возрастает при экстремальных значениях температуры, особенно когда для температуры и типа термопары требуется использование диапазона ±200|250 мВ. Допустим, например, что для измерения температур около 1300 °C используются термопары типа K (хромель-алюмелевые).

Данные примеры показывают, что при экстремальных значениях температуры погрешность смещения на входе значительно ниже, чем погрешность усиления на входе, поскольку требуется больший входной диапазон.

Условия

Температура модуля CR1000, от –25 до 50 °C

Температура = 1300 °C

Опорная температура = 25 °C

Разность температур = 1275 °C

Выходной множитель термопары при 1300 °C = 34,9 мкВ $°C^{-1}$

Выход термопары = 1275 °C • 34,9 мкВ $°C^{-1}$ = 44500 мкВ

Входной диапазон = ±250 мВ

Расчет погрешностей с инверсией входящего сигнала = True

Погрешность (мкВ) = компонент усиления + компонент смещения

= (44500 мкВ • 0,12%) + (1.5 • 66,7 мкВ + 1,0 мкВ)

= 53,4 мкВ + 101,0 мкВ

= 154 мкВ (= 4,41 °C)

Расчет погрешностей с инверсией входящего сигнала = False

Погрешность (мкВ) = компонент усиления + компонент смещения

= (44500 мкВ • 0,12%) + (3 • 66,7 мкВ + 2,0 мкВ)

= 53,4 мкВ + 200 мкВ

= 7,25 мкВ (= 7,25 °C)

Погрешность петли заземления

Если измерительный спай термопары находится в электрическом контакте с измеряемым объектом (или имеет возможность установить такой контакт), во избежание образования петли заземления следует выполнять дифференциальное измерение.

Погрешность вследствие помех

Как правило, входящие помехи в диапазоне ±2_5 мВ при дифференциальном измерении с интеграцией в течение 16,67 мс и инверсией входящего сигнала составляют 0,19 мкВ (среднекв.). Для термопары типа T (приблизительно 40 мкВ/°C), это составит 0,005 °C.

Примечание. Это среднеквадратичное значение; отдельные показания могут отличаться в большую сторону.

Полиномиальная погрешность термопары

В Монографии 175 NIST приведены полиномы высшего порядка для расчета выходного напряжения указанной термопары в рамках широкого диапазона температур. В целях ускорения обработки и улучшения функциональных возможностей CR1000 по вычислению и хранению для преобразования напряжения в температуру в рамках диапазона, охватываемого каждым из типов термопар, используется четыре отдельных полинома шестого порядка. В таблице *Пределы погрешности для полиномов термопары в CR1000* *(стр. 334)* приведены пределы погрешности для полиномов термопары.

Таблица 68. Пределы погрешности для полиномов термопары в CR1000

Тип термопары	*Диапазон, °C*			*Пределы погрешности (°C) относительно стандартов NIST*
T	**-270**	—	**400**	
	-270	—	-200	18 при -270
	-200	—	-100	±0,08
	-100	—	100	±0,001
	100	—	400	±0,015
J	-150	—	760	±0,008
	-100	—	300	±0,002
E	**-240**	—	**1000**	
	-240	—	-130	±0,4
	-130	—	200	±0,005

Таблица 68 Пределы погрешности для полиномов термопары в CR1000				
Тип термопары	***Диапазон, °C***			***Пределы погрешности (°C) относительно стандартов NIST***
	200	—	1000	±0,02
K	**-50**	**—**	**1372**	
	-50	—	950	±0,01
	950	—	1372	±0,04

Погрешность свободного спая

Команды термопар **TCDiff()** и **TCSe()** содержат параметр ***TRef***, позволяющий включить температуру свободного спая в измерение. Напряжение компенсации свободного спая рассчитывается с помощью параметра ***TRef***, входящего в команду термопары и отражающего разницу температур между свободным спаем и 0 °C. Полиномы, используемые для определения напряжения компенсации свободного спая, не покрывают весь диапазон термопары, как показано в таблицах *Пределы погрешности для полиномов термопары в CR1000* *(стр. 334)* и *Диапазон компенсации опорной температуры и полиномиальная погрешность* *(стр. 335)*. Существенные погрешности напряжения компенсации температуры свободного спая приводят к выходу температуры свободного спая за пределы диапазонов аппроксимации с помощью полиномов.

Измерение температуры свободного спая может выполняться на базе команды **PanelTemp()** или любого другого измерения температуры свободного спая. Стандартный и расширенный (-XT) рабочие диапазоны для CR1000 равны от –25 до 50 °C и от –55 до 85 °C соответственно. Эти диапазоны также применяются при измерении температуры свободного спая с использованием **PanelTemp().**

Если опорная температура выходит за пределы диапазона аппроксимации с помощью полинома, существуют два источника погрешности. Наиболее значимая погрешность наблюдается в рассчитанном напряжении компенсации; однако небольшие погрешности могут быть также вызваны нелинейностью коэффициента прямого термоэлектрического эффекта.

Таблица 69. Диапазон компенсации опорной температуры и погрешность		
Тип термопары	***Диапазон, °C***	***Пределы погрешности, °C[1]***
T	от -100 до 100	± 0,001
E	от -150 до 206	± 0,005
J	от -150 до 296	± 0,005
K	от -50 до 100	± 0,01
[1] Относительно стандарта ITS-90, приведенного в Монографии 175 NIST		

Сводка по погрешностям термопары

Погрешности линеаризации температуры термопары и опорной температуры крайне малы, а погрешность измерения напряжения несущественна.

Амплитуда погрешности, описанная в разделе *Анализ погрешностей* *(стр. 327),* свидетельствует о том, что основными источниками погрешностями при измерении термопары обычно являются:

- Стандартная (допускаемая отраслью) погрешность изготовления провода термопары
- Опорная температура

В таблице *Примеры погрешности термопары (стр. 336)* сгруппированы относительные значения амплитуды для

этих типов погрешности. Здесь показан пример худшего случая, при котором:

- Температура в 45 °C измеряется с помощью термопары типа T, а все погрешности максимальны и суммируются.
- Опорная температура = 25 °C, однако отображается как 25,1 °C.
- Контакт, к которому подключена термопара, на 0,05 °C холоднее опорного терморезистора (погрешность 0,15 °C).

Таблица 70. Примеры погрешности термопары

Источник	***Погрешность: °C : % от общей погрешности***			
	Отдельное дифференциальное измерение Интеграция: 250 мкс		***Инвертированное дифференциальное измерение Интеграция: подавление частот 50/60 Гц***	
	Погрешность термопары по ANSI (1 °C)	***Погрешность термопары: 1% наклона***	***Погрешность термопары по ANSI (1 °C)***	***Погрешность термопары: 1% наклона***
Опорная температура	0,15° : 11,5 %	0,15° : 29,9 %	0,15° : 12,2 %	0,15° : 34,7 %
Выход термопары	1,0° : 76,8 %	0,2° : 39,8 %	1,0° : 81,1 %	0,2° : 46,3 %
Измерение напряжения	0,12° : 9,2 %	0,12° : 23,9 %	0,07° : 5,7 %	0,07° : 16,2 %
Помехи	0,03° : 2,3 %	0,03° : 6,2 %	0,01° : 0,8 %	0,01° : 2,3 %
Опорная линеаризация	0,001° : 0,1 %	0,001° : 0,2 %	0,001° : 0,1 %	0,001° : 0,25 %
Выходная линеаризация	0,001° : 0,1 %	0,001° : 0,2 %	0,001° : 0,1 %	0,001° : 0,25 %
Общая погрешность	1,302° : 100 %	0,502° : 100 %	1,232° : 100 %	0,432° : 100 %

8.1.2.2.2. Использование внешнего свободного спая

Внешний спай в изолированном корпусе зачастую используется для упрощения подключений термопар. Он может снизить расход ресурса провода термопары, если измерения производятся на удалении от CR1000. Преобразование внешнего спая в свободный, что предпочтительно для большинства случаев, выполняется за счет протягивания медного провода от спая к CR1000. Как вариант, может использоваться распределительная коробка, позволяющая соединить удлинитель термопары с термопарами, используя при этом команду **PanelTemp()** для определения температуры свободного спая.

Удлинитель термопары имеет меньший температурный диапазон по сравнению со стандартным проводом термопары, однако пределы погрешности в рамках этого диапазона остаются неизменными. Одной из ситуаций, в которой удлинитель термопары имеет преимущество, является выход температуры распределительной коробки за пределы диапазона компенсации свободного спая, заданного CR1000. Этот фактор имеет большое значение только при использовании термопар типа K, поскольку верхний предел диапазона аппроксимации опорной компенсации с помощью полинома равен 100 °C, а верхний предел для провода-удлинителя равен 200 °C. Для других типов термопар диапазон аппроксимации опорной компенсации с помощью полинома равен или больше диапазона удлинителя. В любом случае, погрешность может возникнуть, если в распределительной коробке присутствует градиент температур.

На рисунке *Схема распределительной коробки термопары (стр. 337)*

представлена типичная распределительная коробка, в которой свободный спай относится к CR1000. Клеммные колодки и провод термопары изготовлены из разных металлов. Таким образом, если между A и A' или B и B' присутствует градиент температуры, распределительная коробка будет работать как еще одна подключенная последовательно термопара, в результате чего возникнет погрешность измерения напряжения в CR1000. Это напряжение термоэлектрического смещения также является важным фактором, когда распределительная коробка используется в качестве свободного спая. Это смещение может быть минимизировано путем увеличения теплопроводности и уменьшения расстояния между двумя точками. Наилучшее решение при подключении удлинителя к проводу термопары — использовать соединительные разъемы, обеспечивающие контакт между двумя проводами.

Если внешняя распределительная коробка является также свободным спаем, значения температуры в точках A, A', B и B' должны быть максимально близки (изотермическое условие), чтобы было возможно измерить действительную опорную температуру и избежать возникновения напряжения термоэлектрического смещения. Распределительная коробка должна содержать элементы, обладающие высоким коэффициентом теплопроводности и способные быстро выравнивать градиенты температуры, воздействию которых подвергается коробка. Нет необходимости в разработке коробки с постоянной температурой. Желательно, чтобы коробка медленно реагировала на внешние температурные колебания. Если распределительная коробка устанавливается в полевых условиях, необходимо обеспечить защиту от излучения. Также необходимо внимательно следить, чтобы градиент температуры не возникал вследствие эффекта теплопроводности входящих проводов. CR1000 может использоваться для измерения градиентов температуры в распределительной коробке.

Рисунок 90. Схема распределительной коробки термопары

8.1.2.3. Измерения тока — подробная информация

Связанные темы:
• *Измерения тока — общие сведения* *(стр. 66)*
• *Измерения тока — подробная информация* *(стр. 337)*

Датчики токовых контуров (от 4 до 20 мА, например) подробно рассматриваются в следующих публикациях, доступных на сайте *www.campbellsci.com/app-notes* (*http://www.campbellsci.com/app-notes*):

- *Преобразователи тока на выходе, измеряемые с помощью регистраторов данных Campbell Scientific (2MI-B)*
- *Входной модуль клеммы токового шунта на 100 Ом CURS100*

8.1.2.4. Измерения сопротивления — подробная информация

Связанные темы:
• Измерения сопротивления — технические характеристики
• *Измерение сопротивления — общие сведения* *(стр. 67)*
• *Измерение сопротивления — подробная информация* *(стр. 337)*
• *Измерение сопротивления — команды* *(стр. 551)*

Сопротивление может быть вычислено за счет подачи точного известного напряжения на контур моста сопротивления и измерения возвратного напряжения.

К командам CRBasic для измерения сопротивления относятся:

BrHalf() — полумостовое подключение
BrHalf3W() — трехжильное полумостовое подключение
BrHalf4W() — четырехжильное полумостовое подключение
BrFull() — четырехжильное подключение с полным мостом
BrFull6W() — шестижильное подключение с полным мостом

Читать далее. Доступные модули дополнения для моста сопротивления перечислены в приложении *Преобразователи сигнала (стр. 647)*.

В CR1000 есть пять команд CRBasic для мостового измерения. В таблице *Контуры моста сопротивления с напряжением возбуждения (стр. 338)* представлены идеальные контуры и связанные с ними уравнения. На приведенных схемах резисторы с меткой R_s являются, как правило, датчиками, а с меткой R_f — постоянными (статичными) прецизионными резисторами. В примере CRBasic *Измерение четырехжильного подключения с полным мостом (стр. 340)* приведен код CRBasic, позволяющий измерять и обрабатывать показания четырехжильных контуров с полным мостом.

При измерении моста применяется компенсация напряжения смещения. Помимо параметров ***RevDiff*** и ***MeasOff***, описанных в разделе *Компенсация напряжения смещения (стр. 323)*, команды CRBasic для измерения моста включают параметр ***RevEx***, позволяющий программировать второй набор измерений с инвертированной полярностью тока возбуждения. Большая часть погрешностей смещения, присутствующих в измерении моста, нейтрализуется за счет установки для параметров ***RevDiff, MeasOff*** и ***RevEx*** значения ***True***.

При использовании параметров ***RevDiff, MeasOff*** и ***RevEx*** может снизиться скорость измерений. Если требуется более одного измерения на каждый датчик, как это происходит с командами **BrHalf3W(), BrHalf4W()** и **BrFull6W**, инверсия входящего сигнала и тока возбуждения применяется по отдельности к каждому измерению. Например, при инверсии тока возбуждения в четырехжильном полумостовом подключении (**BrHalf4W()**) дифференциальное измерение перепада напряжения в датчике выполняется с учетом обоих вариантов полярности тока возбуждения, а затем ток возбуждения снова подается и инвертируется для измерения перепада напряжения на постоянном резисторе. Затем результаты выполнения команд измерения (X) должны пройти дальнейшую обработку, чтобы в итоге получилось значение сопротивления. Для этой обработки требуется дополнительное время выполнения программы.

Таблица 71. Контуры моста сопротивления с напряжением возбуждения

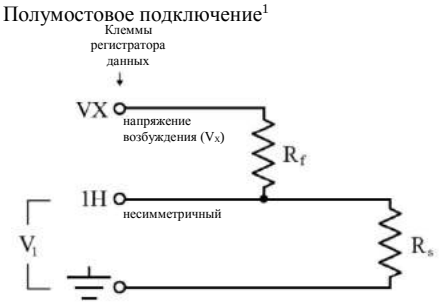
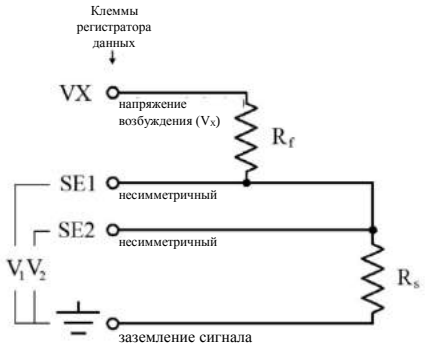
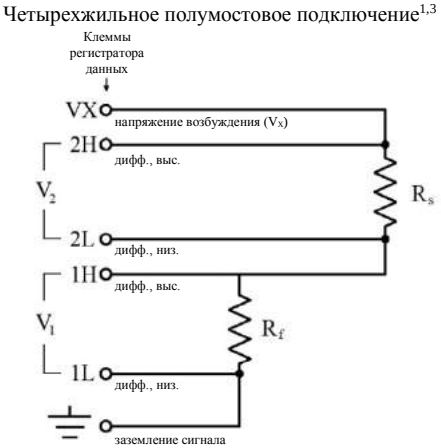
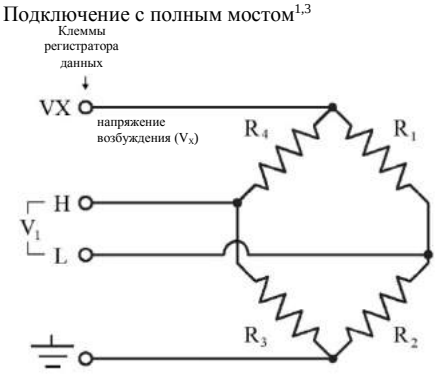
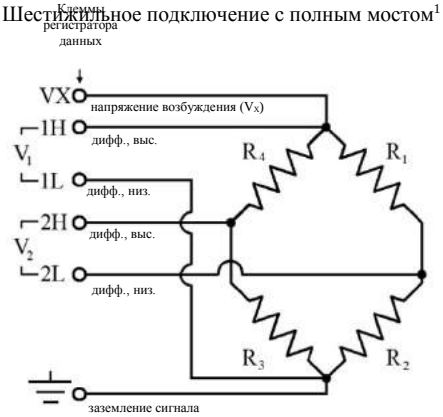
Тип моста сопротивления и схема контура	*Команда CRBasic и фундаментальное соотношение*	*Прочие соотношения*
Полумостовое подключение[1]  	Команда CRBasic: **BrHalf()** Фундаментальное соотношение[2]: $X = \frac{V_1}{V_X} = \frac{R_S}{R_S + R_f}$	$R_S = R_f \frac{X}{1 - X}$ $R_f = \frac{R_S(1 - X)}{X}$
Трехжильное полумостовое подключение[1,3]  	Команда CRBasic: **BrHalf3W()** Фундаментальное соотношение[2]: $X = \frac{2V_2 - V_1}{V_X - V_1} = \frac{R_S}{R_f}$	$R_f = R_S / X$ $R_S = R_f X$
Четырехжильное полумостовое подключение[1,3]  	Команда CRBasic: **BrHalf4W()** Фундаментальное соотношение[2]: $X = \frac{V_2}{V_1} = \frac{R_S}{R_f}$	$R_S = R_f X$ $R_f = R_S / X$

Таблица 71. Контуры моста сопротивления с напряжением возбуждения

Тип моста сопротивления и схема контура	*Команда CRBasic и фундаментальное соотношение*	*Прочие соотношения*
Подключение с полным мостом[1,3]  	Команда CRBasic: **BrFull()** Фундаментальное соотношение[2]: $X = 1000\frac{V_1}{V_X}$ $= 1000\left(\frac{R_3}{R_3+R_4} - \frac{R_2}{R_1+R_2}\right)$	Эти соотношения применимы для **BrFull()** и **BrFull6W()**. $X_1 = \frac{-X}{1000} + \frac{R_3}{R_3+R_4}$ $R_1 = \frac{R_2(1-X_1)}{X_1}$ $R_2 = \frac{R_1X_1}{1-X_1}$
Шестижильное подключение с полным мостом[1]  	Команда CRBasic: **BrFull6W()** Фундаментальное соотношение[2]: $X = 1000\frac{V_2}{V_1}$ $= 1000\left(\frac{R_3}{R_3+R_4} - \frac{R_2}{R_1+R_2}\right)$	$X_2 = \frac{X}{1000} + \frac{R_2}{R_1+R_2}$ $R_3 = \frac{R_4X_2}{1-X_2}$ $R_4 = \frac{R_3(1-X_2)}{X_2}$

[1] Ключ: V_X = напряжение возбуждения; V_1, V_2 = возвратное напряжение датчика; R_f = «постоянный», «мостовой» или «дополняющий» резистор; R_S = «регулируемый» или «сенсорный» резистор.

[2] Где X = результат выполнения команды CRBasic для измерения моста с множителем 1 и смещением 0.

[3] Список доступных входных модулей клеммы, упрощающих процедуру измерения, см. в приложении *Модули моста сопротивления* *(стр. 647)*.

CRBasic: пример 65. Измерение и обработка четырехжильного подключения с полным мостом

```
В этом примере программы демонстрируется измерение и обработка четырехжильного подключения
с полным мостом сопротивления. Измерение по умолчанию для данного примера, хранящееся в
переменной X, подробно анализируется с целью определения сопротивления резистора R1,
используемого в качестве регулируемого резистора в большинстве датчиков, активным элементом
которых является четырехжильное подключение с полным мостом.

Объявление переменных
Public X
Public X1
Public R1
Public R2 = 1000                                    Сопротивление постоянного резистора R2
Public R3 = 1000                                    Сопротивление постоянного резистора R2
Public R4 = 1000                                    Сопротивление постоянного резистора R4

Основная программа
BeginProg
   Scan(500,mSec,1,0)

    Измерение подключения с полным мостом:
    BrFull(X,1,mV2500,1,Vx1,1,2500,True,True,0,_60Hz,1.0,0.0)
    X1 = ((-1 * X) / 1000) + (R3 / (R3 + R4))
    R1 = (R2 * (1 - X1)) / X1

   NextScan
EndProg
```

8.1.2.4.1. Возбуждение переменным током

Для некоторых резистивных датчиков требуется возбуждение переменным током. Возбуждение переменным током определяется как возбуждение с одинаковой продолжительностью подачи и амплитудой положительного (+) и отрицательного (–) тока. Этот метод применяется для электролитических датчиков наклона, измерителей влажности почвы, датчиков водопроницаемости и систем определения влажности. При использовании возбуждения постоянным током с неизменной полярностью для этих датчиков может произойти поляризация материалов датчика и измеряемого вещества. Поляризация может стать причиной ошибочных измерений, изменений калибровки или быстрого выхода датчика из строя.

Другие датчики, как, например, линейно регулируемые дифференциальные трансформаторы (LVDT), требуют применения метода возбуждения переменным током, поскольку им необходим сигнал, поданный посредством индукционной связи. При возбуждении LVDT постоянным током измерение не даст результата.

Команды CRBasic по измерению моста обладают возможностью инверсии полярности и выполнения возбуждения переменным током; для этого параметру ***RevEx*** должно быть присвоено значение ***True***.

Примечание. Следует принять меры предосторожности и не допустить образования петель заземления при измерении датчиков, требующих возбуждения переменным током. См. раздел *Петли заземления при ионных измерениях* *(стр. 109)*.

8.1.2.4.2. Измерения сопротивления — погрешность

Читать далее. В следующих технических документах, доступных по адресу *www.campbellsci.com/app- notes* (*http://www.campbellsci.com/app-notes*), углубленно рассматриваются определенные темы, касающиеся качества измерения напряжения:

- *Предотвращение и устранение проблем, связанных с помехами при измерении*
- *Преимущества инверсии входящего сигнала и тока возбуждения для измерения напряжения*
- *Погрешность измерения напряжения, внутренняя калибровка, логометрические измерения*
- *Оценка погрешности измерения для команд логометрических измерений*

Примечание. Погрешность, описываемая в данном разделе, а также связанные технические характеристики CR1000 не включают погрешности, вызванные самим датчиком или передачей сигнала датчика в CR1000.

Технические характеристики погрешности для логометрических измерений сопротивления сгруппированы в таблице *Погрешность при логометрическом измерении сопротивления* *(стр. 342)*.

Таблица 72. Погрешность при логометрическом измерении сопротивления

от –25 до 50 °C
±(0,04 % от измерения напряжения + смещение)[1]

[1] Измерение напряжения представлено переменной V_1 или V_2 в таблице *Контуры моста сопротивления с напряжением возбуждения* *(стр. 338)*. Смещение аналогично значению при простых измерениях аналогового напряжения. См. таблицу *Смещения измерения аналогового напряжения* *(стр. 313)*.

В плане технических характеристик логометрической погрешности существуют следующие предположения:

- CR1000 соответствует заводским техническим характеристикам калибровки.
- Напряжение возбуждения менее 1000 мВ инвертируется в фазе возбуждения процедуры измерения.
- Воздействие перечисленных ниже факторов не учтено в технических характеристиках:
  - Погрешности мостовых резисторов
  - Помехи датчика
  - Помехи измерения

Более тщательно погрешность при логометрических измерениях рассматривается в техническом документе *Оценка погрешности измерений для команд по логометрическому измерению*, доступном по адресу *www.campbellsci.com/app-notes* (*http://www.campbellsci.com/app-notes*) с июня 2015 г.

8.1.2.5. Тензометрические измерения — подробная информация

Связанные темы:
• *Тензометрические измерения — общие сведения (стр. 68)*
• *Тензометрические измерения — подробная информация (стр. 342)*
• *Примеры FieldCalStrain () (стр. 223)*

Принципиальное использование четырехжильного подключения с полным мостом подразумевает измерение тензодатчиков в форме структурного анализа напряжения. **StrainCalc()** позволяет вычислить микродеформацию (με) с помощью формулы для конкретной используемой конфигурации тензометрического моста. Во всех тензодатчиках, поддерживаемых командой **StrainCalc()**, используется схема с полным мостом. В терминологии тензометрических датчиков понятия «четверть моста», «половина моста» и «полный мост» относятся к числу активных элементов в схеме подключения с полным мостом. Иными словами, «четвертьмостовой» тензодатчик имеет один активный элемент, «полумостовой» — два, «полномостовой» — четыре.

StrainCalc() требует использования кода конфигурации моста. В таблице **Команда StrainCalc():** *уравнения* *(стр. 343)* приведены уравнения, используемые каждым из кодов конфигурации. В начале каждого кода может стоять прочерк (-). Код без прочерка следует использовать, если согласно конфигурации моста выход уменьшается при одновременном увеличении деформации. Код с прочерком используется, если согласно конфигурации моста выход увеличивается при увеличении деформации. В уравнениях

таблицы **Команда StrainCalc():** *уравнения* *(стр. 343)* код с прочерком устанавливает для V_r отрицательную полярность.

Таблица 73. Команда StrainCalc():**уравнения**

StrainCalc() **Код BrConfig**	***Конфигурация***
1	Четвертьмостовой тензодатчик: $$\mu\varepsilon = \frac{-4 * 10^6 V_r}{GF(1 + 2V_r)}$$
2	Полумостовой тензодатчик. Один индикатор параллельно деформации, второй — под углом 90°. $$\mu\varepsilon = \frac{-4 * 10^6 V_r}{GF[(1 + \nu) - 2V_r(\nu - 1)]}$$
3	Полумостовой тензодатчик. Один индикатор параллельно +ε, второй — параллельно –ε $$\mu\varepsilon = \frac{-2 * 10^6 V_r}{GF}$$
4	Полномостовой тензодатчик. Два индикатора параллельно +ε, другие два — параллельно –ε: $$\mu\varepsilon = \frac{-10^6 V_r}{GF}$$
5	Полномостовой тензодатчик. На одной половине моста индикаторы расположены параллельно +ε и –ε, на другой — параллельно +νε и –νε: $$\mu\varepsilon = \frac{-2 * 10^6 V_r}{GF(\nu + 1)}$$
6	Полномостовой тензодатчик. На одной половине моста индикаторы расположены параллельно +ε и –νε, на другой — параллельно –νε и +ε: $$\mu\varepsilon = \frac{-2 * 10^6 V_r}{GF[(\nu + 1) - V_r(\nu - 1)]}$$

где:

- ν: Коэффициент Пуассона (0, если неприменимо)
- GF: Коэффициент чувствительности тензодатчика
- V_r: 0,001 (источник/ноль), если код BRConfig положителен (+)
- V_r: –0,001 (источник/ноль), если код BRConfig отрицателен (–)

где:

- «источник» — результат измерения полного моста ($X = 1000 • V_1 / V_X$) с множителем = 1 и смещением = 0.
- «ноль» — смещение датчика с целью достижения условного нуля (см. **FieldCalStrain()** в разделе *Примеры FieldCal()* *(стр. 213)*).

Пример StrainCalc. См. *Примеры FieldCalStrain()* (стр. 223)

8.1.2.6. Автоматическая калибровка — подробная информация

Связанные темы
- *Автоматическая калибровка — общие сведения* (стр. 92)
- *Автоматическая калибровка — подробная информация* (стр. 344)
- *Автоматическая калибровка — ошибки* (стр. 490)
- *Компенсация напряжения смещения* (стр. 323)
- *Заводская калибровка* (стр. 94)
- *Заводская калибровка или процедуры ремонта* (стр. 476)

В CR1000 используется функция автоматической калибровки для компенсации изменений, вызванных приработкой оборудования и изменением рабочей температуры. При отключении автоматической калибровки точность измерения вне рабочего диапазона ухудшается в 10 раз. То есть в расширенном температурном диапазоне от –40 °C до 85 °C погрешность составляет ±0,12 % от показаний прибора, а в случае отключения автоматической калибровки может снизиться до ±1 %. Но даже если температура CR1000 остается без изменений, при отключении автоматической калибровки присутствует дрейф относительно первоначально снятой градуировочной кривой. Автоматическую калибровку можно отключить, если интенсивность сканирования слишком мала. Ее можно отключить с помощью программы CRBasic командой **Calibrate()**.

Примечание. CR1000 оснащен внутренним эталоном напряжения для калибровки. Там, где требуется высокая точность измерения аналогового напряжения, эталон напряжения должен периодически проверяться и повторно калиброваться Campbell Scientific. Повторную калибровку рекомендуется проводить не реже, чем каждые два года.

Если команда **Calibrate()** не задана, CR1000 проводит автоматическую калибровку в свободное время в фоновом режиме, используя *медленную последовательность* (стр. 157), когда калибровка последующего сегмента выполняется через четыре секунды. Если не хватает времени на фоновую калибровку из-за сканирования пользовательской программы, при компиляции CR1000 отобразится следующее предупреждение: **Предупреждение! Фоновая калибровка отключена.**

8.1.2.6.1. Процесс автоматической калибровки

Функция составного переноса *PGIA* (стр. 306) и *А/Ц-преобразователя* (стр. 507) в CR1000 описывается следующим уравнением:

$$COUNTS = G \cdot Vin + B$$

где COUNTS — это результат А/Ц-преобразования, G — усиление напряжения для заданного входного диапазона, Vin — входное напряжение при подключении к V+ и V–, а B — измеренное изнутри напряжение смещения.

Автоматическая внутренняя калибровка затрагивает только значения G и B, необходимые для выполнения указанной программы CRBasic, в результате чего зависящее от программы число сегментов внутренней калибровки варьируется от 6 (минимум) до 91 (максимум). Как правило, при внутренней калибровке требуется 20 сегментов для аналоговых диапазонов и один сегмент для измерения температуры коммутационной панели, что в сумме дает 21 сегмент. Итого: (21 сегмент) • (4 с / сегмент) = 84 секунды на полную внутреннюю калибровку. Худший случай: (91 сегмент) • (4 с / сегмент) = 364 секунды на полную внутреннюю калибровку.

В процессе включения прибора CR1000 вычисляет коэффициенты калибровки путем расчета среднего значения для десяти полных наборов измерений внутренней калибровки. После включения вновь определенные значения G и B проходят фильтрацию низких частот:

Следующее значение = (1/5) • (новое значение) + (4/5) • (старое значение)

В результате получаются следующие проценты успокоения:

- 20 % для 1 нового значения;
- 49 % для 3 новых значений;
- 67 % для 5 новых значений;
- 89 % для 10 новых значений;
- 96 % для 14 новых значений;

Если обновление выполняется слишком медленно, можно использовать команду **Calibrate()**. Команда **Calibrate()** вычисляет требуемые значения G и B при каждом сканировании без фильтрации низких частот.

Для команды **VoltSe()** значение B определяется как часть внутренней калибровки, только если ***MeasOff = 0***. Исключением является значение B для команды **VoltSe()** с входным диапазоном ±2500 и интеграцией 250 мкс; в таком случае оно всегда определяется в рамках внутренней калибровки как значение для внутреннего использования. Для команды **VoltDiff()** значение B определяется как часть внутренней калибровки, только если параметр ***RevDiff = 0***.

Команды VoltSe() и **VoltDiff()** с одним и тем же входным диапазоном и длительностью интеграции используют одинаковые значения G, но различные значения B. Шесть диапазонов входного напряжения (±5000 мВ, ±2500 мВ, ±250 мВ и ±25 мВ) в сочетании с тремя наиболее частыми длительностями интеграции (250 мкс, половина цикла 50 Гц и половина цикла 60 Гц) дают в результате максимум 18 различных усилений (G), а 18 смещений для измерений **VoltSe()** (B) и 18 смещений для измерений **VoltDiff()** (B) определяются в процессе внутренней калибровки CR1000 (в итоге 54 значения максимум). Эти значения можно просмотреть в таблице **Состояние**, записи в которой соответствуют перечисленным в таблице *Записи калибровки в таблице состояния* *(стр. 346)*.

Автоматическая внутренняя калибровка может быть перекрыта командой **Calibrate()**, которая принудительно инициирует калибровку при каждом выполнении и не применяет фильтрацию низких частот для вновь определенных значений G и B. У команды **Calibrate()** есть два параметра: ***CalRange*** и ***Dest***. ***CalRange*** определяет, должна ли калибровка выполняться только для требуемых входных диапазонов заданной программы CRBasic (***CalRange*** = 0) или для всех диапазонов (***CalRange*** ≠ 0). Параметр ***Dest*** должен иметь достаточную величину для всех возвращенных значений G и B: минимально — 2 для автоматической внутренней калибровки **VoltSE()**, включая B (смещение) для входного диапазона ±2500 мВ и первой интеграций в 250 мкс; максимально — 54 для всех используемых входных диапазонов напряжения и возможных вариантов длительности интеграции.

Пример использования команды **Calibrate()**, запрограммированной на калибровку всех входных диапазонов, представлен в следующем фрагменте кода CRBasic:

```
Calibrate (Dest,Range)
Calibrate(cal(1),true)
```

где ***Dest*** — массив из 54 переменных, а ***Range ≠ 0***, что позволяет выполнить калибровку всех входных диапазонов. Результаты выполнения этой команды перечислены в таблице **Команда Calibrate():** *результаты* *(стр. 347)*.

Таблица 74. Усиления и смещения автокалибровки

Элемент таблицы состояния	*Описания элементов таблицы состояния*			
	Дифференциальное измерение (Diff) Несимметричное измерение (SE)	*Смещение или усиление*	*Входной диапазон ±мВ*	*Интеграция*
CalGain(1)		Усиление	5000	250 мс
CalGain(2)		Усиление	2500	250 мс
CalGain(3)		Усиление	250	250 мс
CalGain(4)		Усиление	25	250 мс
CalGain(5)		Усиление	7,5	250 мс
CalGain(6)		Усиление	2,5	250 мс
CalGain(7)		Усиление	5000	Подавление 60 Гц
CalGain(8)		Усиление	2500	Подавление 60 Гц
CalGain(9)		Усиление	250	Подавление 60 Гц
CalGain(10)		Усиление	25	Подавление 60 Гц
CalGain(11)		Усиление	7,5	Подавление 60 Гц
CalGain(12)		Усиление	2,5	Подавление 60 Гц
CalGain(13)		Усиление	5000	Подавление 50 Гц
CalGain(14)		Усиление	2500	Подавление 50 Гц
CalGain(15)		Усиление	250	Подавление 50 Гц
CalGain(16)		Усиление	25	Подавление 50 Гц
CalGain(17)		Усиление	7,5	Подавление 50 Гц
CalGain(18)		Усиление	2,5	Подавление 50 Гц
CalSeOffset(1)	SE	Смещение	5000	250 мс
CalSeOffset(2)	SE	Смещение	2500	250 мс
CalSeOffset(3)	SE	Смещение	250	250 мс
CalSeOffset(4)	SE	Смещение	25	250 мс
CalSeOffset(5)	SE	Смещение	7,5	250 мс
CalSeOffset(6)	SE	Смещение	2,5	250 мс
CalSeOffset(7)	SE	Смещение	5000	Подавление 60 Гц
CalSeOffset(8)	SE	Смещение	2500	Подавление 60 Гц
CalSeOffset(9)	SE	Смещение	250	Подавление 60 Гц
CalSeOffset(10)	SE	Смещение	25	Подавление 60 Гц
CalSeOffset(11)	SE	Смещение	7,5	Подавление 60 Гц
CalSeOffset(12)	SE	Смещение	2,5	Подавление 60 Гц
CalSeOffset(13)	SE	Смещение	5000	Подавление 50 Гц
CalSeOffset(14)	SE	Смещение	2500	Подавление 50 Гц
CalSeOffset(15)	SE	Смещение	250	Подавление 50 Гц

Таблица 74. Усиления и смещения автокалибровки

Таблица Status Элемент	*Описания элементов таблицы состояния*			
	Дифференциальное измерение (Diff) Несимметричное измерение (SE)	*Смещение или усиление*	*Входной диапазон ±мВ*	*Интеграция*
CalSeOffset(16)	SE	Смещение	25	Подавление 50 Гц
CalSeOffset(17)	SE	Смещение	7,5	Подавление 50 Гц
CalSeOffset(18)	SE	Смещение	2,5	Подавление 50 Гц
CalDiffOffset(1)	Diff	Смещение	5000	250 мс
CalDiffOffset(2)	Diff	Смещение	2500	250 мс
CalDiffOffset(3)	Diff	Смещение	250	250 мс
CalDiffOffset(4)	Diff	Смещение	25	250 мс
CalDiffOffset(5)	Diff	Смещение	7,5	250 мс
CalDiffOffset(6)	Diff	Смещение	2,5	250 мс
CalDiffOffset(7)	Diff	Смещение	5000	Подавление 60 Гц
CalDiffOffset(8)	Diff	Смещение	2500	Подавление 60 Гц
CalDiffOffset(9)	Diff	Смещение	250	Подавление 60 Гц
CalDiffOffset(10)	Diff	Смещение	25	Подавление 60 Гц
CalDiffOffset(11)	Diff	Смещение	7,5	Подавление 60 Гц
CalDiffOffset(12)	Diff	Смещение	2,5	Подавление 60 Гц
CalDiffOffset(13)	Diff	Смещение	5000	Подавление 50 Гц
CalDiffOffset(14)	Diff	Смещение	2500	Подавление 50 Гц
CalDiffOffset(15)	Diff	Смещение	250	Подавление 50 Гц
CalDiffOffset(16)	Diff	Смещение	25	Подавление 50 Гц
CalDiffOffset(17)	Diff	Смещение	7,5	Подавление 50 Гц
CalDiffOffset(18)	Diff	Смещение	2,5	Подавление 50 Гц

Таблица 75. Команда Calibrate(): результаты

Элемент массива Cal()	*Описание элементов массива*				*Типичное значение*
	Дифференциальное измерение (Diff) Несимметричное измерение (SE)	*Смещение или усиление*	*Входной диапазон ±мВ*	*Интеграция*	
1	SE	Смещение	5000	250 мс	±5 LSB
2	Diff	Смещение	5000	250 мс	±5 LSB
3		Усиление	5000	250 мс	–1,34 мВ/LSB
4	SE	Смещение	2500	250 мс	±5 LSB
5	Diff	Смещение	2500	250 мс	±5 LSB
6		Усиление	2500	250 мс	–0,67 мВ/LSB
7	SE	Смещение	250	250 мс	±5 LSB
8	Diff	Смещение	250	250 мс	±5 LSB
9		Усиление	250	250 мс	–0,067 мВ/LSB

Таблица 75. Команда Calibrate(): результаты

Элемент массива Cal()	Описание элементов массива				Типичное значение
	Дифференциальное измерение (Diff) Несимметричное измерение (SE)	Смещение или усиление	Входной диапазон ±мВ	Интеграция	
10	SE	Смещение	25	250 мс	±5 LSB
11	Diff	Смещение	25	250 мс	±5 LSB
12		Усиление	25	250 мс	–0,0067 мВ/LSB
13	SE	Смещение	7,5	250 мс	±10 LSB
14	Diff	Смещение	7,5	250 мс	±10 LSB
15		Усиление	7,5	250 мс	–0,002 мВ/LSB
16	SE	Смещение	2,5	250 мс	±20 LSB
17	Diff	Смещение	2,5	250 мс	±20 LSB
18		Усиление	2,5	250 мс	–0,00067 мВ/LSB
19	SE	Смещение	5000	Подавление 60 Гц	±5 LSB
20	Diff	Смещение	5000	Подавление 60 Гц	±5 LSB
21		Усиление	5000	Подавление 60 Гц	–0,67 мВ/LSB
22	SE	Смещение	2500	Подавление 60 Гц	±5 LSB
23	Diff	Смещение	2500	Подавление 60 Гц	±5 LSB
24		Усиление	2500	Подавление 60 Гц	–0,34 мВ/LSB
25	SE	Смещение	250	Подавление 60 Гц	±5 LSB
26	Diff	Смещение	250	Подавление 60 Гц	±5 LSB
27		Усиление	250	Подавление 60 Гц	–0,067 мВ/LSB
28	SE	Смещение	25	Подавление 60 Гц	±5 LSB
29	Diff	Смещение	25	Подавление 60 Гц	±5 LSB
30		Усиление	25	Подавление 60 Гц	–0,0067 мВ/LSB
31	SE	Смещение	7,5	Подавление 60 Гц	±10 LSB
32	Diff	Смещение	7,5	Подавление 60 Гц	±10 LSB
33		Усиление	7,5	Подавление 60 Гц	–0,002 мВ/LSB
34	SE	Смещение	2,5	Подавление 60 Гц	±20 LSB
35	Diff	Смещение	2,5	Подавление 60 Гц	±20 LSB
36		Усиление	2,5	Подавление 60 Гц	–0,00067 мВ/LSB
37	SE	Смещение	5000	Подавление 50 Гц	±5 LSB
38	Diff	Смещение	5000	Подавление 50 Гц	±5 LSB
39		Усиление	5000	Подавление 50 Гц	–0,67 мВ/LSB
40	SE	Смещение	2500	Подавление 50 Гц	±5 LSB
41	Diff	Смещение	2500	Подавление 50 Гц	±5 LSB
42		Усиление	2500	Подавление 50 Гц	–0,34 мВ/LSB
43	SE	Смещение	250	Подавление 50 Гц	±5 LSB

Таблица 75. Команда Calibrate(): результаты

Элемент массива Cal()	Описание элементов массива				Типичное значение
	Дифференциальное измерение (Diff) Несимметричное измерение (SE)	Смещение или усиление	Входной диапазон ±мВ	Интеграция	
44	Diff	Смещение	250	Подавление 50 Гц	±5 LSB
45		Усиление	250	Подавление 50 Гц	–0,067 мВ/LSB
46	SE	Смещение	25	Подавление 50 Гц	±5 LSB
47	Diff	Смещение	25	Подавление 50 Гц	±5 LSB
48		Усиление	25	Подавление 50 Гц	–0,0067 мВ/LSB
49	SE	Смещение	7,5	Подавление 50 Гц	±10 LSB
50	Diff	Смещение	7,5	Подавление 50 Гц	±10 LSB
51		Усиление	7,5	Подавление 50 Гц	–0,002 мВ/LSB
52	SE	Смещение	2,5	Подавление 50 Гц	±20 LSB
53	Diff	Смещение	2,5	Подавление 50 Гц	±20 LSB
54		Усиление	2,5	Подавление 50 Гц	–0,00067 мВ/LSB

8.1.3. Импульсные измерения — подробная информация

Связанные темы:
• Импульсные измерения — технические характеристики
• *Импульсные измерения — общие сведения* *(стр. 68)*
• *Импульсные измерения — подробная информация* *(стр. 349)*
• *Импульсные измерения — команды* *(стр. 553)*

Читать далее. См. разделы *СЧЕТЧИКИ ИМПУЛЬСОВ* *(стр. 349)* и Импульсы на С-клеммах в *Технических характеристиках CR1000* *(стр. 97)*. Программирование импульсных измерений описано в *Справке по редактору CRBasic* (команды **PulseCount()** и **TimerIO()**).

Примечание. Доступны периферийные устройства Campbell Scientific, способные увеличить число входных каналов для импульсов, измеряемых с помощью CR1000. Более подробную информацию см. в приложении *Списки внешнего оборудования для измерения и контроля* *(стр. 366)*.

На рисунке *Типы выходных сигналов импульсного датчика* *(стр. 69)* представлены типы импульсных сигналов, измеряемых с помощью CR1000:

- низкоуровневый переменный ток
- высокочастотный
- переключение-замыкание

На рисунке *Схема переключений-замыканий* *(стр. 350)* показан базовый внутренний контур и внешние подключения импульсного датчика переключения-замыкания. В таблице *Импульсные измерения: клеммы и программирование* *(стр. 351)* сгруппированы доступные измерения и клеммы для этих измерений, а также используемые команды CRBasic. Количество клемм, настраиваемое для импульсного входа, определяется из таблицы *Определения клемм CR1000* *(стр. 76)*.

Рисунок 91. Типы выходных сигналов импульсного датчика

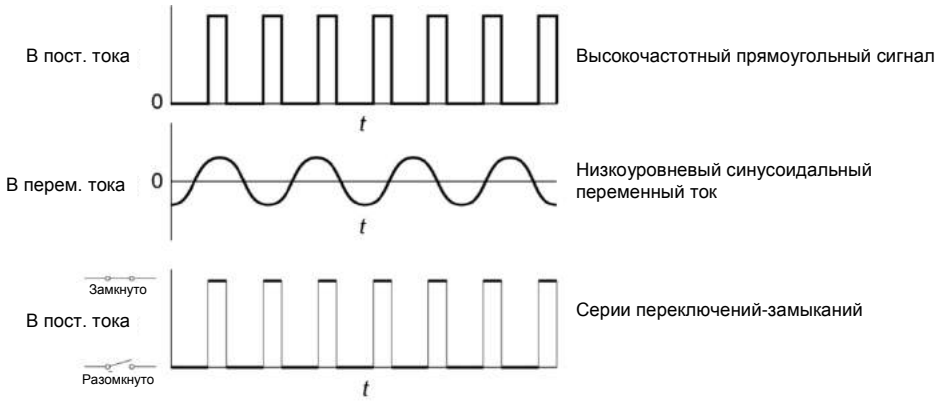


Рисунок 92. Импульсный датчик переключения-замыкания

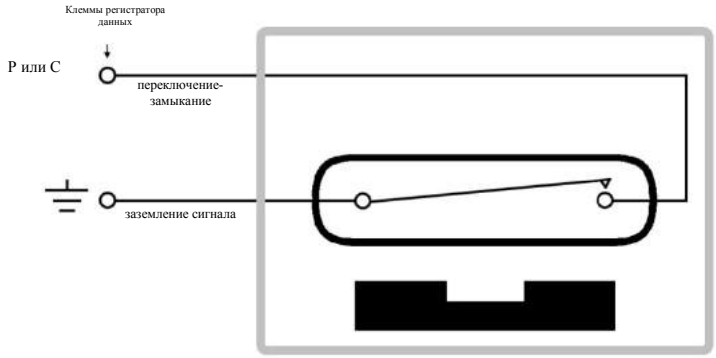


Рисунок 93. Настраиваемые клеммы для импульсного входа

Таблица 76. Импульсные измерения: клеммы и программирование			
Разрешение	***P-клеммы***	***C-клеммы***	***Команда CRBasic***
Низкоуровневый переменный ток, счетчик	✓		**PulseCount()**
Низкоуровневый переменный ток, Гц	✓		
Низкоуровневый переменный ток, скользящее среднее	✓		
Высокочастотный сигнал, счетчик	✓	✓	
Высокочастотный сигнал, Гц	✓	✓	
Высокочастотный сигнал, скользящее среднее	✓	✓	
Переключение-замыкание, счетчик	✓	✓	
Переключение-замыкание, Гц	✓	✓	
Переключение-замыкание, скользящее среднее	✓	✓	
Расчетный период		✓	**TimerIO()**
Расчетная частота		✓	
Время, прошедшее с крайнего значения на предыдущем порте		✓	
Время, прошедшее с крайнего значения на порте 1		✓	
Количество крайних значений		✓	
Количество импульсов, период		✓	
Количество импульсов, частота		✓	

8.1.3.1. Клеммы для импульсного измерения

Клеммы P

- Диапазон входного напряжения = от –20 до 20 В

Если входное импульсное напряжение превышает ±20 В, необходимо задействовать сторонние внешние преобразователи сигнала. Если потребуется помощь, свяжитесь со специалистом по применению Campbell Scientific.
Ни при каких условиях не допускается измерять напряжение выше 50 В.

Клеммы C

- Диапазон входного напряжения = от –8 до 16 В пост. тока

C-клеммы, настроенные для импульсного входа, отличаются небольшой постоянной времени (25 нс) для резистивно-емкостного фильтра между контактной колодкой и входным буфером CMOS, что позволяет выполнять импульсные измерения вплоть до частоты 250 кГц, а также подсчитывать крайние значения вплоть до 400 кГц. Входной буфер CMOS распознает входы ≥ 3,8 В как высокочастотные и входы ≤ 1,2 В как низкочастотные.

Датчики с открытым коллектором (биполярные транзисторы) и датчики со свободным стоком (MOFSET), как правило, измеряются как датчики частоты. **C**-клеммы могут быть отрегулированы под использование открытого коллектора или свободного стока с внешним резистором, повышающим логический уровень, как показано на рисунке «Использование резистора, повышающего логический уровень, для C-клемм». Резистор, повышающий логический уровень, взаимодействует с внутренним резистором на 100 кОм, понижающим логический уровень, что позволяет нагрузить входы до уровня > 3,8 В для обеспечения надежного измерения.

8.1.3.2. Низкоуровневые измерения с переменным током — подробная информация

Связанные темы:
• *Модули ввода по низкоуровневому переменному току — общие сведения* *(стр. 367)*
• *Низкоуровневые измерения с переменным током — подробная информация* *(стр. 352)*
• *Импульсные модули ввода — списки* *(стр. 646)*

Низкоуровневые сигналы переменного тока (синусоидальные) могут измеряться с помощью **P**-клемм. К датчикам, обычно выводящим низкоуровневый переменный ток, относятся:

- Анемометры генератора переменного тока

Возможны следующие измерения:

- Значение
- Частота (Гц)
- Скользящее среднее

Вращающиеся магнитные датчики, как правило, генерируют напряжение переменного тока в диапазоне от тысячных долей вольта при низкой скорости вращения до нескольких вольт при высокой скорости вращения. Для клемм, настроенных под низкоуровневый вход переменного тока, характерно линейное преобразование сигнала, что позволяет измерять сигналы в диапазоне от 20 мВ среднекв. (полный размах ±28 мВ) до 14 В среднекв. (полный размах ±20 В).

Клеммы P

- Максимальная входная частота зависит от входного напряжения:
  - от 1,0 до 20 Гц при 20 мВ среднекв.
  - от 0,5 до 200 Гц при 200 мВ среднекв.
  - от 0,3 до 10 кГц при 2000 мВ среднекв.
  - от 0,3 до 20 кГц при 5000 мВ среднекв.
- Команда CRBasic: **PulseCount()**

Внутренняя связь по переменному току используется для устранения напряжений смещения постоянного тока вплоть до ±0,5 В пост. тока.

Клеммы C

Низкоуровневые сигналы переменного тока не могут измеряться напрямую **C**-клеммами. В приложении *Список модулей импульсного ввода* *(стр. 646)* приведена информация о внешних модулях расширения клемм, доступных для преобразования низкоуровневых сигналов переменного тока в прямоугольные сигналы.

8.1.3.3. Высокочастотные измерения

Высокочастотные (прямоугольные) сигналы могут измеряться с помощью **P**- или **C**-клемм. К обычным датчикам, выводящим высокочастотный сигнал, относятся:

- Анемометры фотомодулятора
- Расходомеры

Измерения включают показания счетчиков, частоту (Гц) и скользящее среднее. В разделе *Разрешение по частоте* *(стр. 353)* приведена информация о различном разрешении по измерению частоты в зависимости от того, выполняется ли измерение с помощью команды **PulseCount()** или **TimerIO()**.

Клеммы P

- Максимальная входная частота = 250 кГц
- Команды CRBasic: **PulseCount()**

Высокочастотные импульсные входы связаны с инвертирующим входным буфером CMOS посредством гистерезиса на входе. Входной буфер CMOS находится на уровне выхода **0** при входном напряжении ≥ 2,2 В и на уровне выхода **1** при входящем напряжении ≤ 0,9 В. К клемме автоматически подключается внутренний резистор на 100 кОм, повышающий логический уровень до 5 В пост. тока. Этот повышающий логический уровень резистор обслуживает устройства выхода с открытым коллектором (свободным стоком).

Клеммы C

- Максимальная входная частота = 1 кГц
- Команды CRBasic: **PulseCount(), TimerIO()**

8.1.3.3.1. Разрешение по частоте

Разрешение по измерению частоты, выполненному с помощью команды **PulseCount()**, рассчитывается как

$$FR = \frac{1}{S}$$

где

FR = разрешение по измерению частоты (Гц)
S = интервал сканирования программы CRBasic

Разрешение по измерению частоты, выполненному с помощью команды **TimerIO()**:

$$FR = \frac{R/E}{P * (P + (R/E))}$$

где

FR = разрешение по частоте в рамках измерения (Гц)
R = разрешение по времени в рамках измерения **TimerIO()** = 540 нс
P = период входящего сигнала (в секундах). Например, P = 1/1000 Гц = 0,001 с
E = число нарастающих фронтов на прогон сканирования или 1, в зависимости от того, что выше

Таблица 77. Пример. E для входящего сигнала частотой 10 Гц

Scan (Сканирование)	*Нарастающий фронт / сканирование*	*E*
5,0	50	50
0,5	5	5
0,05	0.5	1

Команда TimerIO() измеряет частоты ≤ 1 кГц с высокочастотным разрешением на основе коротких (менее секунды) интервалов. Наоборот, в результате измерений частоты менее секунды с помощью **PulseCount()** получаются измерения с более низким разрешением. Допустим, частота на входе — 1 кГц. В таблице *Сравнение разрешений по частоте* *(стр. 354)* перечислены варианты разрешения по частоте, которые ожидаются для сигнала частотой 1 кГц, измеряемого с помощью команд **TimerIO()** и **PulseCount()** по интервалам сканирования 0,5 с и 5,0 с.

При увеличении интервала измерения с 1 с до 10 с посредством увеличения интервала сканирования (если используется **PulseCount()**) или усреднения (при использовании **PulseCount()** или **TimerIO()**) итоговое разрешение по частоте положительно изменяется с 1 Гц на 0,1 Гц. Усреднение может быть выполнено с помощью команд **Average(), AvgRun()** и **AvgSpa()**. Кроме того, в команде **PulseCount()** есть опция ввода числа, превышающего ***1,*** в качестве значения параметра ***POption***. При этом вводится интервал усреднения в миллисекундах, позволяющий напрямую вычислить скользящее среднее. Тем не менее процесс усреднения следует проводить осторожно. При усреднении любого измерения снижается уверенность в том, что результат действительно отражает реальный аспект измеряемого явления.

Таблица 78. Сравнение разрешений по частоте

	Сканирование: 0,5 с	*Сканирование: 5,0 с*
PulseCount(), POption=1	FR = 2 Гц	FR = 0,2 Гц
TimerIO(), *Function=2*	FR = 0,0011 Гц	FR = 0,00011 Гц

8.1.3.3.2. Измерения частоты: вопросы и ответы

В: Если на интервал сканирования приходится более одного импульса, какое значение возвращает команда **TimerIO()**, настроенная под измерения частоты? Находит ли она среднее значение по измеренным периодам, рассчитывая частоту на его основе (f = 1/T)? Например,

```
Scan(50,mSec,10,0)
   TimerI0(WindSpd(),11111111,00022000,60,Sec)
```

A: В фоновом режиме происходит сохранение показаний 32-битного счетчика таймирования при каждом переходе сигнала в соответствии с программой (нарастание или убывание). Счетчик работает с постоянной высокой частотой. Количество показаний также увеличивается при каждом переходе. При выполнении команды **TimerIO()** используется разница во времени между крайним значением перед последним выполнением и крайним значением перед текущим прогоном. Количество переходов между этими двумя моментами времени делится на разницу во времени, что в результате дает расчетную частоту. В случае наличия нескольких крайних значений между интервалами выполнения для процесса расчета предполагается, что частота не изменяется в интервале выполнения. При расчете возвращается среднее значение независимо от изменений сигнала.

8.1.3.4. Измерения для переключения-замыкания и открытого коллектора

Сигналы переключения-замыкания, а также сигналы с открытым коллектором могут измеряться с помощью **P-** или **C**-клемм. Механические переключения-замыкания имеют тенденцию к колебаниям перед тем, как произойдет полное замыкание. Если не применен фильтр, колебания могут привести к появлению нескольких показаний счетчика, относящихся к одному и тому же событию. CR1000 автоматически фильтрует колебания. Благодаря фильтрации максимальная частота переключения-замыкания ниже, чем максимальная частота высокочастотного измерения. К датчикам, обычно выводящим сигнал переключения-замыкания или сигнал с открытым коллектором, относятся:

- Дождемеры с опрокидывающимся сосудом
- Анемометры переключения-замыкания
- Расходомеры

Опции вывода данных включают показания счетчиков, частоту (Гц) и скользящее среднее.

Клеммы P

Внутренний резистор на 100 кОм, повышающий логический уровень, повышает входящий сигнал до 5 В пост. тока с переключением-размыканием, в то время как переключение-замыкание на землю доводит сигнал на входе до 0 В. Постоянная времени для внутреннего аппаратного фильтра колебаний составляет 3,3 мс. Конфигурации подключений представлены в таблице *Переключения-замыкания и открытые коллекторы для P-клемм* *(стр. 356)***.**

- Максимальная входная частота = 90 Гц
- Команда CRBasic: **PulseCount()**

Клеммы C

Режим переключения-замыкания — это особая функция подсчета крайних значений, измеряющая сигналы переключения-замыкания или сигналы с открытыми коллекторами. Операционная система фильтрует колебания. Конфигурации подключений представлены в таблице *Переключения-замыкания и открытые коллекторы для C-клемм* *(стр. 357)***.**

- Максимальная входная частота = 150 Гц
- Команда CRBasic: **PulseCount()**

8.1.3.5. Время крайних значений

Время и период крайних значений могут быть измерены с помощью **P-** или **C**-клемм. К прикладным областям применения расчета времени крайних значений относятся:

- Измерения для контроля обратной связи с использованием модуляции ширины или длительности импульсов(PWM/PDM).

Измерения содержат время между крайними значениями, выраженное частотой (Гц) или периодом (мкс).

Клеммы C

- Максимальная входная частота < 1 кГц
- Команда CRBasic: **TimerIO()**
- Обнаружены фронты нарастания или убывания прямоугольного сигнала:
  - Фронт нарастания — переход от < 1,5 В пост. тока к > 3,5 В пост. тока.
  - Фронт убывания — переход от > 3,5 В пост. тока к < 1,5 В пост. тока.
- Разрешение расчета времени для крайнего значения составляет примерно 540 нс.

8.1.3.6. Подсчет крайних значений

○ Показания счетчика для крайних значений могут быть измерены с помощью **C**-клемм.

Клеммы C

- Максимальная входная частота 400 кГц
- Команда CRBasic: **TimerIO()**
- Обнаружены фронты нарастания или убывания прямоугольного сигнала:
  - Фронт нарастания — переход от < 1,5 В пост. тока к > 3,5 В пост. тока.
  - Фронт убывания — переход от > 3,5 В пост. тока к < 1,5 В пост. тока.

8.1.3.7. Импульсные измерения: советы

Базовое подключение выходных импульсных датчиков представлено в таблице *Переключения-замыкания и открытые коллекторы* *(стр. 356, стр. 357)*

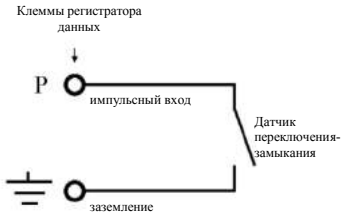
В рамках команды **PulseCount()** при измерении импульсных входов на клеммах **P** или **C** используются выделенные 24-битные счетчики, накапливающие все показания за установленный на программном уровне интервал сканирования. Разрешение по импульсным счетчикам составляет 1 показание или 1 Гц. Показания счетчиков считываются в начале каждого прогона сканирования, а затем сбрасываются. Счетчик будет переполнен, если накопленное количество показаний превысит 16 777 216, что приведет к погрешностям измерений.

- Показания счетчика являются предпочтительной опцией вывода **PulseCount()** при измерении числа переворотов дождемера с опрокидывающимся сосудом или открытий двери. Многие импульсные датчики выхода, такие как анемометры и расходомеры, откалиброваны по частоте (Гц (стр. 517)) и поэтому обычно измеряются с помощью опции частотного выхода **PulseCount()**.
- Точность **PulseCount()** ограничена небольшой погрешностью интервала сканирования, составляющей ±(3 миллионные доли от интервала сканирования + 10 мкс), и погрешностью разрешения по измерению ±1 / (интервал сканирования). Сумма фактически составляет ±1 / (интервал сканирования).
- Модуль *LLAC4* *(стр. 646)* позволяет преобразовать сигналы, не относящиеся к уровню TTL, включая низкоуровневые сигналы переменного тока, в сигналы уровня TTL для их дальнейшего ввода с помощью клемм **C**.
- Как показано в таблице *Переключения-замыкания и открытые коллекторы для C-клемм* *(стр. 357)*, **C**-клеммы имеют, благодаря зенеровскому диоду на 6,2 В, входное сопротивление в 100 кОм с входным напряжением < 6,2 В пост. тока. В случае входного

напряжения ≥ 6,2 В пост. тока входное сопротивление **C**-клемм составляет только 220 Ом.

Таблица 79. Переключения-замыкания и открытые коллекторы для P-клемм

Переключение-замыкание для ***P****-клеммы*



Открытый коллектор для ***P****-клеммы*

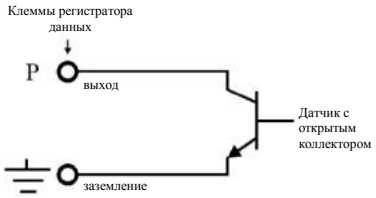
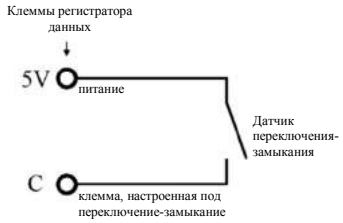
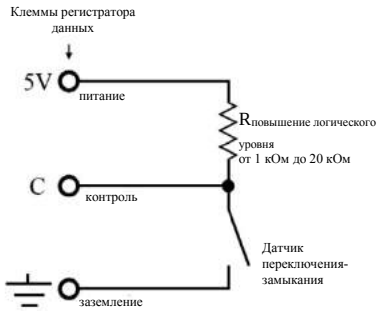


Таблица 80. Переключения-замыкания и открытые коллекторы для C-клемм

Переключение-замыкание для ***C****-клеммы:*
Без повышения логического уровня

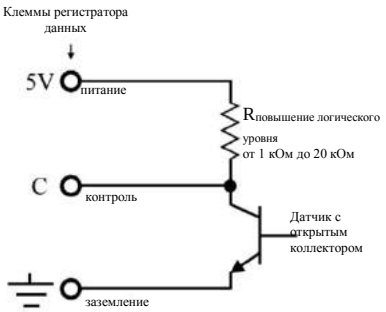


Переключение-замыкание для ***C****-клеммы:*
повышение логического уровня до
5 В пост. тока



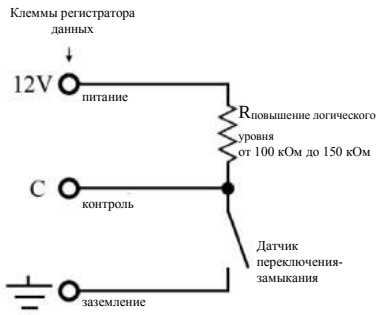
Утечка тока покоя < 50 мкА

Открытый коллектор для ***C****-клеммы:*
повышение логического уровня
до 5 В пост. тока



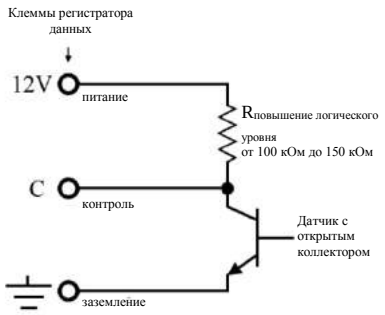
Утечка тока покоя < 50 мкА

*Переключение-замыкание для **C**-клеммы: повышение логического уровня до 12 В пост. тока*



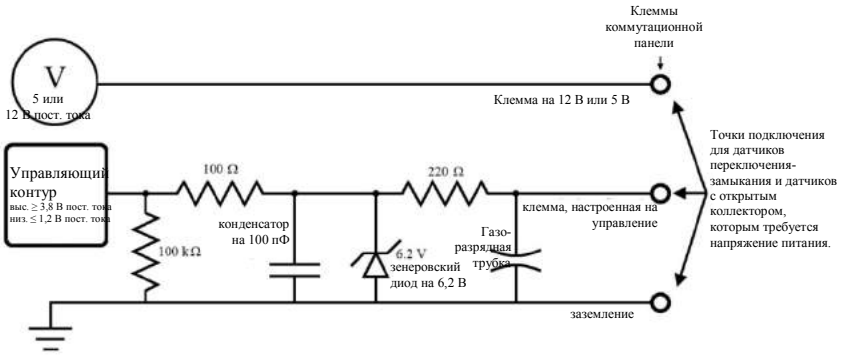
Утечка тока покоя < 60 мкА

*Открытый коллектор для **C**-клеммы: повышение логического уровня до 12 В пост. тока*



Утечка тока покоя < 60 мкА

Внутренняя электрическая схема CR1000, поддерживающая измерения с открытым коллектором и с переключением-замыканием (для информации)



- Необходимо внимательно относиться к техническим характеристикам. Рекомендуем потратить время на ознакомление с измеряемым сигналом и совместимыми входными клеммами, а также с командами CRBasic. В таблице *Три различия в технических характеристиках между клеммами P и C* *(стр. 358)* сравниваются технические характеристики клемм импульсного ввода с целью выявления необходимости сопоставления подходящего устройства для конкретного случая.

Таблица 81. Три различия в технических характеристиках между клеммами P и C

	Клемма P	***Клемма C***
Максимальный высокочастотный сигнал	250 кГц	400 кГц
Максимальное входное напряжение	20 В пост. тока	16 В пост. тока
Пороговые значения переходов состояния	Подсчет при переходе от <0,9 В пост. тока к >2,2 В пост. тока	Подсчет при переходе от < 1,2 В пост. тока к > 3,8 В пост. тока

8.1.3.7.1. Условия появления значения NAN при TimerIO()

- **NAN** будет результатом измерения **TimerIO()**, если выполняется одно из двух условий:
  - Истечение времени ожидания
  - Слишком высокая частота сигнала (> 3 кГц). Если на клемме **C** наблюдается слишком высокая частота, операционная система CR1000 отключает сигнал прерывания, регистрирующий точное время, пока не начнется обработка следующего прогона сканирования. Это делается для того, чтобы процессор CR1000 не был перегружен лишними сигналами прерывания. Небольшой резистивно-емкостный фильтр, модернизированный под переключатель датчика, должен решить эту проблему.

8.1.3.7.2. Входные фильтры и затухание сигнала

P- и **C**-клеммы оснащены фильтрами импульсного ввода с целью снижения электронных помех, которые могут негативно повлиять на корректность подсчета. Чем выше постоянная времени (τ) фильтра, тем строже фильтр. В таблице *Постоянные времени (стр. 359)* перечислены значения τ. Таким образом, в то время как клемма **C**, измеряемая в соответствии с командой измерения частоты **TimerIO()**, может прекрасно подходить для чистых сигналов, для измерения сигналов с электронными помехами может потребоваться фильтр клеммы **P** (значение τ существенно выше).

Входные фильтры гасят амплитуду (напряжение) сигнала. Величина затухания является функцией от частоты, проходящей фильтр. Сигналы с более высокой частотой затухают намного заметнее. При достаточно сильном затухании сигнал может перестать соответствовать пороговому условию перехода состояния, установленному устройством обнаружения в соответствии с таблицей *Клеммы импульсного входа и измерения (стр. 69)*). Во избежание чрезмерного затухания напряжение выходного датчика должно быть увеличено при повышенных частотах. Например, в таблице *Затухание низкоуровневого фильтра переменного тока (стр. 360)* показано, что для низкоуровневых входов переменного тока требуется повышенное напряжение, чтобы преодолеть затухание фильтра на клеммах **P**, настроенных под низкоуровневый сигнал переменного тока: фильтр постоянной времени 8,5 мс (частота 19 Гц, 3 дБ) для сигналов с низкой амплитудой; постоянная времени 1 мс (частота 159 Гц, 3 дБ) для сигналов с более высокой (> 0,7 В) амплитудой.

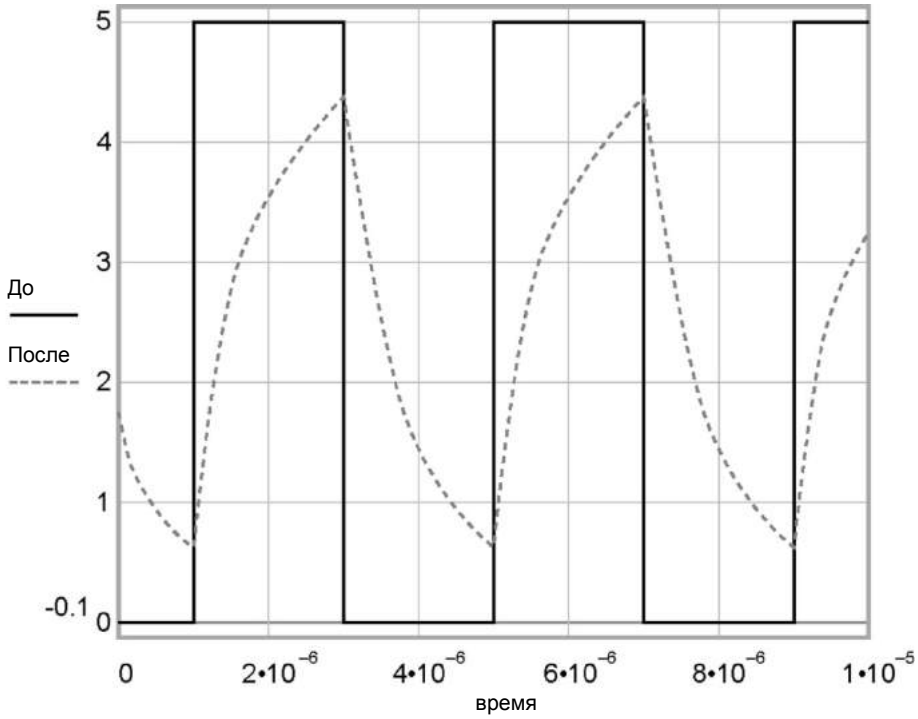
В случае клемм **P** входной резистивно-емкостный фильтр с постоянной времени приблизительно 1 мкс предшествует инвертирующему входному буферу CMOS. Итоговое сокращение амплитуды представлено на рисунке *Сокращение амплитуды волны подсчета импульсов (стр. 360)***.** Для входящего прямоугольного сигнала от 0 до 5 В пост. тока на импульсной клемме максимальная частота, которая может быть подсчитана в высокочастотном режиме, составляет приблизительно 250 кГц.

Таблица 82. Постоянные времени (τ)

Разрешение	*τ*
P-клемма: режим низкоуровневого переменного тока	См. сноску к таблице *Затухание фильтра для частотных сигналов (стр. 360)*
P-клемма: высокочастотный режим	1,2
P-клемма: режим переключения-замыкания	3300
C-клемма: высокочастотный режим	0,025
C-клемма: режим переключения-замыкания	0,025

Таблица 83. Амплитуда низкоуровневого переменного тока и максимальная измеряемая частота	
мВ перем. тока (среднекв.)	***Максимальная частота***
20	20
200	200
2000	10 000
5000	20 000

Рисунок 94. Сокращение амплитуды волны подсчета импульсов (до и после применения фильтра постоянной времени 1 мкс)



8.1.4. Среднее значение за период — подробная информация

Связанные темы:
• Среднее значение за период — технические характеристики
• *Среднее значение за период — общие сведения* *(стр. 70)*
• *Среднее значение за период — подробная информация* *(стр. 360)*

CR1000 может измерять период сигнала на **несимметричной** клемме. Указанное число циклов рассчитывается по времени с разрешением 136 нс, соответственно, при измерении периода разрешение 136 нс делится на выбранное число циклов.

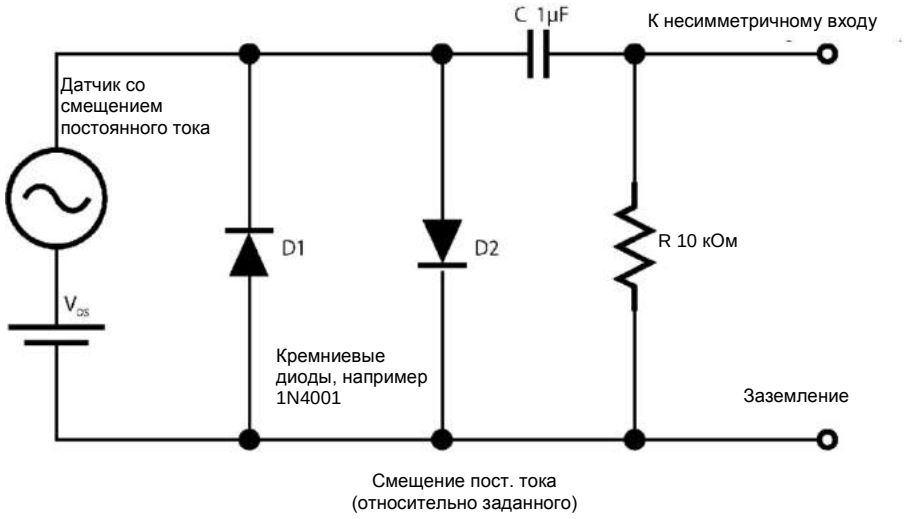
Низкоуровневые сигналы подвергаются усилению до поступления в компаратор напряжений. Внутренний компаратор напряжений обращается к запрограммированному пороговому значению. Пороговый параметр позволяет связывать внутренний компаратор напряжений с напряжениями, отличными от 0 В. Например, пороговое значение 2500 мВ позволяет внутреннему компаратору обнаруживать цифровые сигналы от 0 до 5 В пост. тока без необходимости дополнительной электрической схемы преобразования входящего сигнала. Пороговое значение позволяет напрямую подключать стандартные цифровые сигналы, но не рекомендуется для сигналов датчика с низкой амплитудой. Для датчиков с полной амплитудой менее 20 мВ рекомендуется использование подстроечного конденсатора постоянного тока с

целью центрирования сигнала на заземлении CR1000 (пороговое значение = 0) ввиду ухода напряжения смещения и ограниченной погрешности (±10 мВ) и разрешения (1,2 мВ) порогового значения, отличного от нуля. На рисунке *Входной контур преобразования для поиска среднего значения за период* *(стр. 361)* показан пример контура.

Минимальные требования к ширине импульса повышаются (максимальная частота снижается) по мере увеличения усиления. Сигналы, превышающие указанный максимум для диапазона, насыщают усилительные каскады и не предотвращают функционирование вплоть до максимальной заданной частоты. Как показано на рисунке, рекомендуется использовать встречно-включенные диоды для ограничения сигналов с высокой амплитудой до пределов диапазона входящего сигнала.

Внимание! Сигналы с помехами и медленным переходом через пороговое значение напряжения обладают потенциалом дополнительного подсчета показаний на уровне точки переключения компаратора. Компаратор напряжения с гистерезисом в 20 мВ проходит через усилительные каскады напряжения. Фактический гистерезис с учетом входящего сигнала равен результату деления 20 мВ на выбранное усиление напряжения. Фактический гистерезис с учетом входящего сигнала в диапазоне ±25 мВ равен 2 мВ; следовательно, помехи входящего сигнала, равные 2 мВ, могут стать причиной лишнего подсчета показаний. Для достижения наилучшего результата следует выбрать наибольший входной диапазон (наименьшее усиление), соответствующий минимальным требованиям к входящему сигналу.

Рисунок 95. Входной контур преобразования для поиска среднего значения за период



8.1.5. Вибрационные струнные измерения — подробная информация

Связанные темы:
- Вибрационные струнные измерения — технические характеристики
- *Вибрационные струнные измерения — общие сведения* *(стр. 71)*
- *Вибрационные струнные измерения — подробная информация* *(стр. 361)*

CR1000 может обеспечить измерение струнных счетчиков или вибрационных сенсорных полосок, включая тензометрические датчики, преобразователи давления, пьезодатчики, уклономеры, датчики трещин и датчики напряжения. Эти датчики применяются в строительстве, геологии и гидрологии, демонстрируя высокую точность, стабильность и надежность. CR1000 может считывать показания струнных счетчиков с помощью специальных модулей интерфейса. Подключение дополнительных датчиков производится через мультиплексор (см. раздел *Аналоговые мультиплексоры* *(стр. 646)*).

На рисунке *Струнный счетчик (стр. 362)* представлена базовая конструкция датчика. Для выполнения измерения на катушки датчика подается ток возбуждения с *качающейся частотой (стр. 530)*. В идеальном случае все нерезонансные частоты быстро затухают, а резонансная частота сохраняется. Т. к. резонансная частота отсекает линии магнитной индукции в принимающей катушке, та же самая частота возникает в сигнальных проводах, соединяющих датчик с CR1000 или интерфейсом.

Измерение резонансной частоты с помощью среднего значения за период является классическим методом, однако Campbell Scientific разработала статические и динамические методы спектрального анализа (*VSPECT (стр. 532)*$_t^m$), которые дают превосходное подавление шумов, более высокое разрешение, возможность получать данные диагностики и проводить динамические измерения VSPECT с частотой до 333,3 Гц.

Резистивный термометр (терморезистор или резистивный датчик температуры), включенный в большинство конструкций струнных счетчиков, может использоваться для компенсации погрешностей температуры при измерении.

Рисунок 96. Струнный счетчик

8.1.5.1. Измерения временной области

Хотя струнные измерения с усреднением значения за период в рамках временной области считаются устаревшими для многих сфер применения, они все же могут выполняться для клемм **H L.** Для измерения используется команда **VibratingWire()**. Измерения могут производиться непосредственно с помощью этих клемм, однако обычно выполняются через струнный интерфейс, усиливающий и преобразующий сигнал колебания струн с целью предоставления входящих данных для встроенных терморезисторов или РДТ. Campbell Scientific больше не поставляет интерфейсы подобного типа.

В большинстве случаев предпочтительным выбором являются продвинутые методы статических и динамических измерений VSPECT™.

8.1.6. Снятие показаний интеллектуальных датчиков — подробная информация

Связанные темы:
• *Снятие показаний интеллектуальных датчиков — общие сведения (стр. 71)*
• *Снятие показаний интеллектуальных датчиков — подробная информация (стр. 362)*

8.1.6.1. RS-232 и TTL

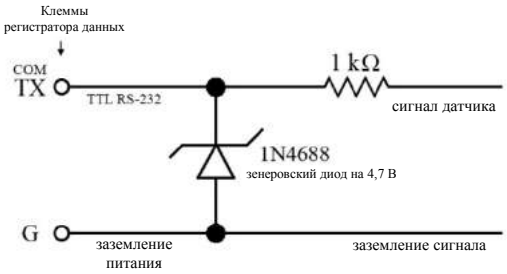
Читать далее. *Команды последовательного ввода/вывода (стр. 583)* и *Последовательный ввод/вывод (стр. 245)*.

CR1000 может принимать и записывать большинство показаний TTL (от 0 до 5 В пост. тока) и истинных данных RS-232, поступающих от устройств, таких как интеллектуальные датчики. В таблице *Определения клемм CR1000 (стр. 76)* представлена информация о клеммах и последовательных портах, настраиваемых для коммуникаций TTL или RS-232 (истинных). Для использования порта **CS I/O** для истинных коммуникаций RS-232 требуется применение устройства интерфейса. См. приложение *Последовательные*

интерфейсы CS I/O (стр. 652). Если необходимы дополнительные последовательные входы, можно подключить модули расширения последовательного ввода. См. приложение *Список модулей последовательного ввода/вывода (стр. 646)*. Последовательные данные обычно регистрируются в виде текстовых строк, которые затем разбираются (разбиваются) в соответствии с программой CRBasic.

Примечание. **C**-клеммы, настроенные как Tx, передают только логику от 0 до 5 В пост. тока. В то же время **C**-клеммы, настроенные как Rx, считывают большинство истинных сигналов RS-232. При подключении последовательных датчиков к клемме **C**, настроенной как Rx, энергопотребление датчика может увеличиться на несколько мА из-за фиксации напряжения в CR1000. Может понадобиться внешний резистор, подключенный последовательно к линии Rx, помогающий ограничить расход тока, однако это не рекомендуется в случае крайне высоких скоростей передачи (в бодах). На рисунке «Контур ограничения входящего сигнала RS-232 на клемме **C** до 5 В» *(стр. 363)* показан контур, ограничивающий напряжение до 5 В пост. тока.

Рисунок 97. Контур ограничения входящего сигнала RS-232 на клемме C до 5 В пост. Тока



8.1.6.2. Поддержка датчика SDI-12 — подробная информация

Связанные темы:
- *Поддержка датчика SDI-12 — общие сведения (стр. 72)*
- *Поддержка датчика SDI-12 — подробная информация (стр. 363)*
- *Последовательный интерфейс ввода/вывода: поддержка датчика SDI-12 — программные ресурсы (стр. 267)*
- *Поддержка датчика SDI-12 — команды (стр. 555)*

SDI-12 — это протокол связи, разработанный для передачи цифровых данных от интеллектуальных датчиков в устройства сбора данных. Это простой протокол, требующий наличия единственного провода связи. Как правило, устройство сбора данных также подает питание (12 В пост. тока и заземление) на датчик SDI-12. Команда **SDI12Recorder()** обеспечивает связь с датчиками SDI-12 через клеммы, настроенные под входящий сигнал SDI-12. С помощью таблицы *Определения клемм CR1000 (стр. 76)* можно определить клеммы, настраиваемые для коммуникации SDI-12.

8.1.7. Калибровка на месте — общие сведения

Связанные темы:
- *Калибровка на месте — общие сведения (стр. 73)*
- *Калибровка на месте — подробная информация (стр. 210)*

Калибровка увеличивает точность измерительного устройства, регулируя его выходные параметры или измерение выходных параметров таким образом, чтобы они соответствовали верифицированным эталонным величинам. Непосредственная регулировка выходного сигнала датчика является предпочтительным вариантом, но он не всегда доступен.

Добавляя команды **FieldCal()** или **FieldCalStrain()** к программе CR1000

CRBasic, можно регулировать линейный датчик, меняя применяемые к измерению запрограммированный множитель и сдвиг.

8.1.8. Эффекты подключения

Связанные темы:
• *Эффекты подключения — общие сведения* (стр. 74)
• *Эффекты подключения — подробная информация* (стр. 364)

Проводка кабелей может оказывать значительное влияние на реакцию и точность датчика. Это, как правило, не касается датчиков Campbell Scientific. Датчики Campbell Scientific разработаны для оптимальной работы с установленными на заводе кабелями.

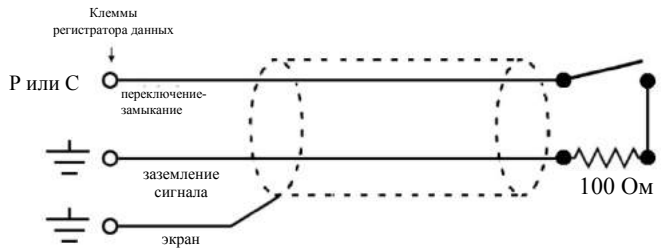
8.1.8.1. Кабели аналогового датчика

Длина кабеля в аналоговых датчиках с большой долей вероятности влияет на время успокоения сигнала. Более подробная информация доступна в разделе *Время успокоения сигнала* (стр. 317).

8.1.8.2. Импульсные датчики

Вследствие большого интервала между переключениями замыканий в дождемерах с опрокидывающимся сосудом между проводами в длинных кабелях может возникнуть ощутимое емкостное сопротивление. Накопленный заряд может вызвать дугообразование при замыкании переключателя и, как следствие, сократить срок службы переключателя. Как показано на рисунке *Резистор для ограничения тока в контуре дождемера* (стр. 364), резистор на 100 Ом последовательно подсоединяется к переключателю для предотвращения дугообразования. Подобный резистор устанавливается на все дождемеры, в настоящее время поставляемые Campbell Scientific.

Рисунок 98. Резистор для ограничения тока в контуре дождемера



8.1.8.3. Датчики RS-232

Длина кабеля датчика RS-232 должна быть ограничена 50 футами.

8.1.8.4. Датчики SDI-12

Согласно стандарту SDI-12, допускается длина кабеля до 200 футов. Campbell Scientific не рекомендует использовать для датчиков SDI-12 кабели длиной выше 200 футов, однако более длинные кабели иногда могут быть адаптированы за счет увеличения калибра провода или питания датчика с помощью второго источника на 12 В пост. тока, размещенного возле датчика.

8.1.9. Синхронизация измерений

Связанные темы:
• *Синхронизация измерений — общие сведения* (стр. 74)
• *Синхронизация измерений — подробная информация* (стр. 365)

Выбор времени измерения обычно контролируется на основании часов CR1000. Когда датчики в сети эксплуатируются с применением одного CR1000, время измерения синхронизируется, часто в течение нескольких миллисекунд, в зависимости от количества датчиков и типа измерений. Из-за большого количества датчиков, ограничений длины кабеля или больших расстояний между участками измерений может потребоваться установить несколько CR1000. Описанные ниже методы позволяют администраторам сети синхронизировать часы CR1000 и измерения в рамках сети CR1000.

При планировании операции изменения времени следует соблюдать осторожность. При любом изменении времени CR1000 отклонение нового значения времени от старого может оказаться достаточным, чтобы в таблицах данных произошел пропуск записи. Любая команда, используемая для синхронизации времени, должна выполняться только после команд **CallTable()** и рассчитываться по времени так, чтобы не пересекаться с интервалами вывода данных.

К методам синхронизации измерений в рамках сети относятся:

1. *LoggerNet* (стр. 95) — если надежность телекоммуникаций характерна для всех CR1000 в сети, автоматизированная проверка часов *LoggerNet* обеспечивает простой функционал синхронизации времени. Погрешность ограничивается системными часами на ПК, где запущен сервер *LoggerNet*. Точность ограничивается задержками сетевой передачи. *LoggerNet* компенсирует задержки в большинстве телекоммуникационных систем и может достигать синхронизации с отклонением < 100 мс. В сильно нагруженных RF-соединениях или подключениях к сети Интернет с большой протяженностью могут наблюдаться погрешности в 2–3 секунды.

Примечание. Стандартные часы на ПК известны своей неточностью. По ссылке **http://www.nist.gov/pml/div688/grp40/its.cfm** можно ознакомиться с полезными рекомендациями по настройке точности часов на ПК.

2. Цифровой триггер — в отличие от часов, цифровой триггер может подавать сигнал синхронизации. Если возможно организовать кабельное соединение от CR1000 к CR1000, каждое устройство CR1000 может принимать информацию о фронте нарастания цифрового импульса от основного CR1000 и синхронизировать измерения или другие функции с помощью команд **WaitDigTrig()**, независимо от часов CR1000 или меток времени. Когда программы выполняются в поточном режиме, измерения могут синхронизироваться с точностью до нескольких микросекунд (см. раздел *Сканирование WaitDigTrig* (стр. 157)).

3. Команды *PakBus* (стр. 88) — CR1000 является устройством с поддержкой PakBus, следовательно, оно может быть узлом в сети PakBus. Тактовые генераторы узлов в сети PakBus синхронизируются с помощью команд **SendGetVariable(), ClockReport()** или **PakBusClock()**. Разрешение часов CR1000 составляет 10 мс и соответствует разрешению, используемому функциями синхронизации часов PakBus. В сетях без использования маршрутизаторов, повторителей сигналов или повторных передач время коммуникации становится причиной дополнительной погрешности (как правило, несколько десятых долей миллисекунды). Команды часов PakBus устанавливают время по окончании сканирования с целью минимизации риска пропуска записи в таблице данных. Этот процесс отличается от процедуры проверки часов *LoggerNet*, поскольку здесь не используется расчет среднего времени прохождения сигнала туда и обратно с целью определения задержки сетевого соединения.

4. Радиосвязь — доступная в PakBus сеть радиосвязи имеет преимущество перед связью по Ethernet в том плане, что команда **ClockReport()** может передаваться на все регистраторы данных в сети одновременно. На каждом устройстве будет настроено время с помощью единственного сообщения PakBus от основного устройства. Для этого все регистраторы данных в сети должны быть запрограммированы посредством команды **PakBusClock()**.

Примечание. При использовании функций часов PakBus повторно выполняется синхронизация команды **Scan()**. Не допускается использование чаще одного раза в минуту. Сдвиг часов в CR1000 происходит с достаточно низкой скоростью, чтобы выполнения команды **ClockReport()** один раз в минуту хватало для сохранения максимально возможного расхождения между часами в 30 мс.
При любом из методов синхронизации необходимо внимательно контролировать момент и принцип выполнения операций. Изменение времени может привести к пропуску сканирования или записи, если было выполнено в некорректный момент или оказалось слишком существенным.

5. GPS — часы в CR1000 могут синхронизироваться друг с другом с точностью приблизительно до 10 мс посредством команды **GPS()**. Устройства CR1000, выпущенные в октябре 2008 года и позже (серийные номера ≥ [20409]) могут синхронизироваться с точностью до нескольких микросекунд и примерно до 200 мкс от UTC. Пока доступен сигнал GPS, CR1000 в основном использует именно его в качестве постоянного источника информации о времени, что минимизирует риск скачков системного времени и пропуска записей.

6. Ethernet — любое устройство CR1000 с сетевым подключением (Интернет, GPRS, частная сеть) может синхронизировать часы с единым всемирным временем (UTC) посредством команды **NetworkTimeProtocol()**. Погрешность обычно находится в пределах 10 мс. NTP-сервер может быть представлен другим регистратором данных или любым NTP-сервером (например, сервером электронной почты или nist.gov). Рекомендуется использовать локальный сервер, обладающий низкой или, как минимум, соответствующей задержкой связи. Наоборот, не рекомендуется выполнять команду **NetworkTimeProtocol()** на пике сканирования; старайтесь запрашивать серверное время в промежутке между четными секундами.

8.2. Внешнее оборудование для измерения и контроля — подробная информация

Связанные темы:
• *Внешнее оборудование для измерения и контроля — общие сведения (стр. 85)*
• *Внешнее оборудование для измерения и контроля — подробная информация (стр. 366)*
• *Внешнее оборудование для измерения и контроля — списки (стр. 645)*

Внешние устройства расширяют возможности CR1000 в плане ввода и вывода. Некоторые внешние устройства разработаны как SDM (синхронные измерительные устройства) или CDM (CPI-устройства для измерения). Устройства SDM и CDM относятся к классу интеллектуальных внешних устройств, получающих команды от CR1000 и возвращающих данные обратно с помощью специальных протоколов связи через клеммы SDM и интерфейсы CPI. В следующих разделах описываются внешние устройства, классифицированные по типам измерений.

8.2.1. Модули аналогового ввода

Читать далее. Для получения дополнительной информации см. приложение *Список модулей аналогового ввода (стр. 646)*.

Количество входов аналоговых датчиков можно расширить при помощи

механических или полупроводниковых мультиплексоров. Мультиплексоры предназначены для несимметричных и дифференциальных входов, а также входов для мостов сопротивления или входов термопар.

8.2.2. Модули импульсного ввода

Читать далее. Для получения дополнительной информации см. приложение *Список модулей импульсного ввода (стр. 646)*.

Расширительные модули импульсного ввода предназначены для измерения скорости замыкания контактов датчика, состояния, частоты, счета импульсов и определения интервалов времени.

8.2.2.1. Модули ввода по низкоуровневому переменному току — общие сведения

Связанные темы:
• *Модули ввода по низкоуровневому переменному току — общие сведения (стр. 367)*
• *Низкоуровневые измерения с переменным током — подробная информация (стр. 352)*
• *Импульсные модули ввода — списки (стр. 646)*

Модули ввода по низкоуровневому переменному току увеличивают число низкоуровневых сигналов по переменному току, которое может контролировать CR1000 путем преобразования низкоуровневых сигналов по переменному току в высокочастотный импульс.

8.2.3. Модули последовательного ввода/вывода — подробная информация

Читать далее. Для получения дополнительной информации см. приложение *Список модулей последовательного ввода/вывода (стр. 646)*.

Определенные трудности могут возникать при фиксации ввода с интеллектуальных устройств последовательного вывода. Несколько модулей последовательного ввода-вывода Campbell Scientific предназначены для улучшения считывания и анализа последовательных данных. Campbell Scientific рекомендует проконсультироваться с техническим специалистом при подборе модуля последовательного ввода для определенной сферы применения.

8.2.4. Терминальные модули ввода

Читать далее. Для получения дополнительной информации см. приложение *Список пассивных преобразователей сигнала (стр. 647)*.

Терминальные модули ввода (TIM) представляют собой устройства с контурами поддержки измерений в удобной комплектации. Модули TIM включают в себя делители напряжения для отсечки выходного напряжения датчиков до уровней, совместимых с CR1000, модулями для мостов сопротивления и шунтирующими модулями для измерения показаний аналоговых датчиков по току.

8.2.5. Вибрационные струнные модули

Читать далее. Для получения дополнительной информации см. приложение *Список вибрационных струнных модулей (стр. 647)*.

Вибрационные струнные модули служат для сопряжения вибрационных струнных преобразователей с CR1000.

8.2.6. Модули аналогового вывода

Читать далее. Для получения дополнительной информации см. приложение *Список модулей непрерывного аналогового вывода (CAO) (стр. 649)*.

CR1000 может масштабировать измеренные или обработанные значения и передавать их в цифровом виде в устройство аналогового вывода. Устройство аналогового вывода производит цифро-аналоговое преобразование для вывода аналогового сигнала по напряжению или по току. Уровень выходного сигнала поддерживается до обновления по команде CR1000.

8.2.7. Модули управления ПЛК — общие сведения

Связанные темы:
- *Управление ПЛК — общие сведения (стр. 74)*
- *Управление ПЛК — подробная информация (стр. 244)*
- *Модули управления ПЛК — общие сведения (стр. 368)*
- *Модули управления ПЛК — списки (стр. 648)*
- *Контроль ПЛК — команды (стр. 562)*
- Переключаемое выходное напряжение — технические характеристики
- Переключаемое выходное напряжение — общие сведения
- *Переключаемое выходное напряжение — подробная информация (стр. 103)*

Общей функцией CR1000 является контроль электропитания, подаваемого на внешнее устройство. Для двоичного (вкл/выкл) или аналогового (переменного) управления предусматриваются бортовые управляющие клеммы и периферийные устройства. Также предусматривается коммутируемая клемма 12 В пост. тока (**SW12V**). См. раздел *Переключаемое нерегулируемое напряжение (номинал 12 В) (стр. 105)*.

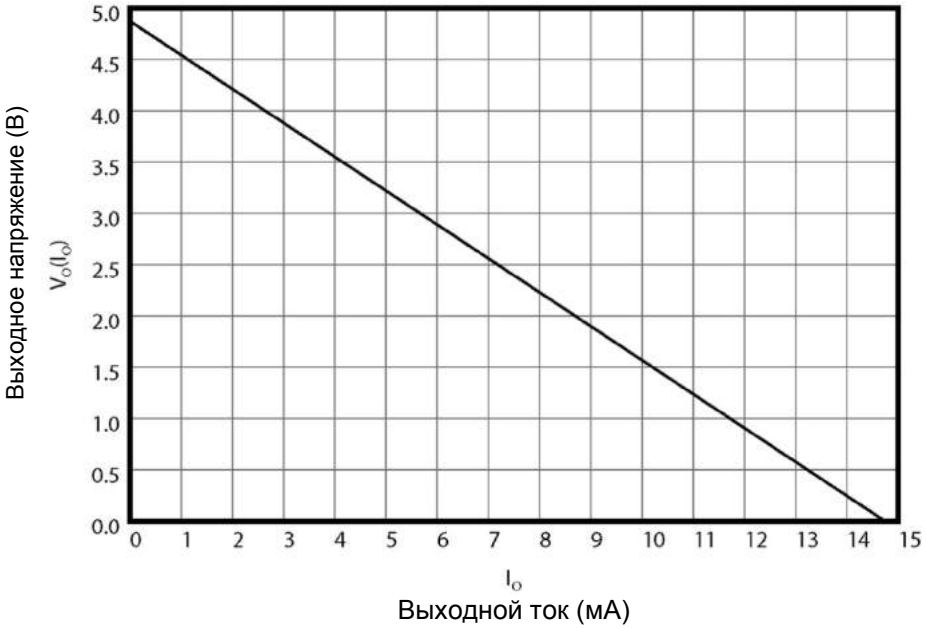
8.2.7.1. Клеммы, настроенные на управление

Клеммы C могут быть настроены в качестве выходных портов низкого (0 В пост. тока) или высокого (5 В пост. тока) напряжения с помощью команд **PortSet()** или **WriteIO()**. Порты **C4, C5** и **C7** могут быть настроены на широтно-импульсную модуляцию с максимальными периодами 36,4 с, 9,1 с и 2,27 с, соответственно. Клемма, настроенная на цифровой ввод-вывод, обычно используется для управления внешней управляющей цепью реле, поскольку сам по себе порт обладает ограниченной управляющей способностью. Управляющая способность определяется по наличию электропитания 5 В пост. тока и выходного сопротивления 330 Ом. Она вычисляется по формуле:

$$V_o = 4{,}9\ В - (330\ Ом) \cdot I_o,$$

где V_o — предел управления, а I_o — ток, необходимый для внешнего устройства. На рис. *Подача тока с порта управления (стр. 369)* представлена взаимозависимость.

*Рисунок 99. Подача тока с клемм **C**, настроенных на управление*



8.2.7.2. Реле и драйверы реле

Читать далее. Для получения дополнительной информации см. приложение *Список модулей драйверов реле (стр. 649)*.

Некоторые драйверы реле производятся компанией Campbell Scientific. Совместимые, недорогие и надежные одноканальные драйверы реле предусматриваются для широкого диапазона нагрузок различных поставщиков электронного оборудования, таких как *Crydom, Newark и Mouser (стр. 534)*.

8.2.7.3. Компонентные реле

На рисунке *Цепь драйвера реле с реле (стр. 370)* показана типовая цепь драйвера реле в сочетании с реле с возбуждением от катушки, которое может использоваться для переключения внешнего электропитания на устройство. В данном примере, если для клеммы, настроенной на управление, устанавливается высокое значение, напряжение 12 В пост. тока от регистратора данных подается через катушку реле, в результате чего замыкается реле, что приводит к замыканию цепи электропитания и включению вентилятора.

В других случаях применения может потребоваться простое переключение электропитания на устройство без прохождения через реле. На рисунке *Переключение электропитания без реле (стр. 370)* показан этот случай. Если запитываемое устройство потребляет ток свыше 75 мА при комнатной температуре (предел транзистора средней мощности 2N2907A), использование реле является обязательным.

Рисунок 100. Цепь драйвера реле с реле

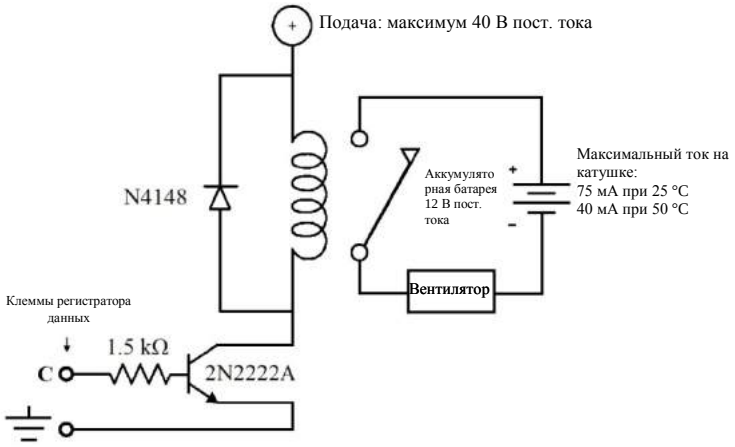
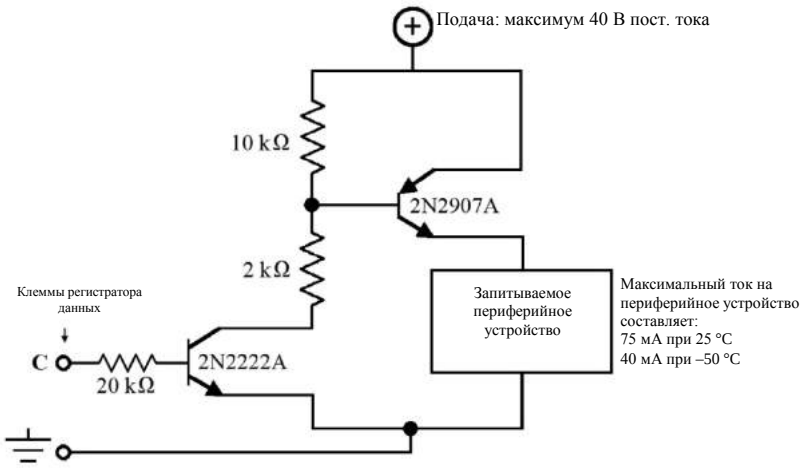


Рисунок 101. Переключение электропитания без реле



8.3. Память

Связанные темы:
• *Память — общие сведения* *(стр. 87)*
• *Память — подробная информация* *(стр. 370)*
• *Устройства хранения данных — список* *(стр. 653)*

8.3.1. Накопитель

Модуль памяти CR1000 состоит из четырех энергонезависимых накопителей:

- SRAM с внутренним аварийным питанием от батареи;
- внутренняя флеш-память;
- внутренняя последовательная флеш-память;
- внешняя флеш-память (опциональный диск USB:);
- внешняя память CompactFlash® *опциональная CF-карта и модуль (диск CRD:)* *(стр. 653).*

В таблице *Выделение памяти CR1000 (стр. 371)* и таблице *Память SRAM CR1000 (стр. 372)* показана структура памяти CR1000 по данным носителям. Большинство функций памяти CR1000 используются и поддерживаются в автоматическом режиме. Тем не менее пользователи должны периодически проверять области памяти, в которых хранятся файлы данных, программные файлы CRBasic и файлы изображений. См. раздел *Управление файлами в памяти CR1000 (стр. 382)* для получения более подробной информации.

По умолчанию хранение данных (пространство для сохранения данных) организовано в циклическом режиме. После завершения цикла, когда память заполнена, старые данные перезаписываются новыми. Однако команда **DataTable()** позволяет установить таблицу данных ***заполнения и остановки записи***.

Таблица 84. Выделение памяти CR1000

Сектор памяти	Комментарии
Основной модуль SRAM с аварийным питанием от батареи[1] 4 МБ*	• переменные ОС; • программа, скомпилированная в CRBASIC, с двоичной структурой; • переменные CRBASIC; • память окончательных данных; • коммуникационная память; • USR: диск RAM FAT32; • память KEEP; • динамическое распределение памяти при выполнении процедур; • см. таблицу *Память SRAM CR1000 (стр. 372)* для получения более подробной информации.
Операционная система Флеш-память[2] 2 МБ	• операционная система (ОС); • серийный номер; • код платы; • код загрузки; • стирается при загрузке новой операционной системы (код загрузки стирается только в случае его изменения).

Таблица 84. Выделение памяти CR1000

Внутренняя последовательная флеш-память [3] 512 кБ	• Настройки устройства (12 кБ) — адрес и настройки PakBus, наименование станции. Восстанавливаются при изменении настройки. • Диск CPU: (500 кБ) — программные файлы, файлы калибровки в полевых условиях, прочие редко перезаписываемые файлы. При компиляции и выполнении программа записывается на данный накопитель автоматически для загрузки при последующих процедурах включения питания. Файлы накапливаются, пока не будут удалены при помощи функции **File Control (Управление файлами)** или команды **FilesManage()**. Диск USR: используется для хранения других типов файлов. Доступная память на диске CPU: указывается в таблице **Status (Состояние)** в поле **CPUDriveFree (Свободные байты на накопителе ЦПУ).** • Файловая система FAT32. • Ограниченное число циклов записи (100 000). • Невысокая скорость при последовательном доступе
Внешняя флеш-память (Опционально) 2 ГБ: Диск USB:	*Диск USB: (стр. 653)* — хранит программные файлы. Содержит копию запрошенных данных из таблиц окончательных данных в виде файлов при использовании команды **TableFile()**. Данные диска USB: можно выгрузить из накопительного устройства с помощью проводника *Windows*. Диск USB: может упростить использование файла Powerup.ini.
Внешний CompactFlash (Опционально) ≤ 16 ГБ: Диск CRD:	*Диск CRD: (стр. 653)* — хранит программные файлы. Содержит копию данных из таблиц окончательных данных в виде файлов при использовании команды **TableFile()** с функцией ***Option 64*** (заменяет собой **CardOut()**). См. *Запись высокочастотных данных на карты памяти (стр. 205)* для получения более подробной информации. Если ПК запрашивает данные, то они в первую очередь предоставляются с SRAM. Если запрошенные записи были перезаписаны в SRAM, данные отправляются с диска CRD:. С другой стороны данные диска CRD: могут быть выгружены в двоичном формате с помощью *ПО поддержки регистратора данных (стр. 95)* **File Control (Управление файлами)**. Преобразование двоичных файлов осуществляется при помощи программного обеспечения *CardConvert*. 10 % или 80 кБ памяти CF (в зависимости от того, что меньше) резервируется для хранения программ. Карты CF могут упростить использование файла Powerup.ini.

[1] SRAM

• В CR1000 емкость SRAM изменена с 2 до 4 МБ в сентябре 2007. SN ≥ 11832 имеют объем 4 МБ.

[2] Флеш-память рассчитана на более чем 1 миллион циклов перезаписи.

[3] Последовательная флеш-память рассчитана на 100 000 циклов перезаписи (50 000 циклов перезаписи на устройствах 128 кБ). Особое внимание необходимо уделять программам, которые перезаписывают память на использование дисков CRD: или USR:, чтобы это не привело к преждевременному выходу из строя диска CPU:.

• В CR1000 увеличен объем последовательной флеш-памяти со 128 до 512 кБ в мае 2007. SN ≥ 9452 имеют объем 512 кБ.

Таблица 85. Основная память CR1000

Использование	*Комментарии*
Статическая память	Оперативная память, используемая операционной системой. Восстанавливается при включении электропитания, перекомпиляции программы и событиях сторожевой программы.
Рабочие настройки и свойства	*Память «Keep» (стр. 519)*. Хранит настройки, такие как адрес PakBus, наименование станции, сигнальные интервалы, списки соседних узлов и т. д. Кроме того, хранит динамические свойства, такие как таблица маршрутизации, коммуникационные таймауты и т. д.
Программа CRBasic Оперативная память	Хранит на текущий момент скомпилированную и выполняющуюся пользовательскую программу. Данный сектор восстанавливается при включении электропитания, перекомпиляции и событиях сторожевой программы.
Переменные и константы	Хранит переменные, используемые программой CRBasic. Данные значения могут сохраняться после включения электропитания, перекомпиляции и событий сторожевой программы, если в выполняемой программе предусматривается команда **PreserveVariables.**
Память окончательных данных	Хранит данные. Заполняет память, оставшуюся после удовлетворения всех остальных запросов. Может настраиваться как кольцевая память или память с ограниченным заполнением. Если для таблиц данных, отведенных пользователю, недостаточно памяти, возникает ошибка компиляции. Присваивается наименьший приоритет в выделении памяти SRAM.
Коммуникационная память 1	Создание и временное хранение пакетов PakBus.
Коммуникационная память 2	Созданная таблица маршрутизации: список известных узлов и маршрутов к узлам. Маршрутизаторам требуется больше пространства по сравнению с конечными узлами из-за необходимости сохранения маршрутов к соседним узлам. Увеличение значения в поле PakBusNodes в таблице **Status (Состояние)** приводит к увеличению данного выделения памяти.
Диск USR: <= 3,6 МБ (4 МБ память) <= 1,5 МБ (2 МБ память) Меньше на устройствах более ранних версий с более ограниченной памятью.	Выделяется дополнительно. Содержит файлы изображений. Содержит копию памяти окончательных данных, если используется команда **TableFile().** Предоставляет память для операций **FileRead()** и **FileWrite()**. Управление осуществляется в модуле File Control (Управление файлами). Состояние отображается в таблице **Status (Состояние)** в полях «USRDriveSize» и «USRDriveFree».

Таблица 86. Диски памяти

Диск	*Рекомендуемые типы файлов*
CPU:[1]	cr1, .CAL
USR:[2]	cr1, .CAL
USB:	.DAT
CRD	Основное использование — расширение *памяти окончательных данных* *(стр. 515)*, однако используется также для хранения файлов .JPG, cr1 и .DAT.

[1] Диск CPU: использует файловую систему FAT16, поэтому его емкость ограничена до 128 файлов. Если имена файлов превышают длину в 8,3 символа (например, 12345678.123), храниться может меньшее количество файлов.

[2] Диск USR: использует файловую систему FAT32, поэтому, помимо целесообразности и доступной памяти, ограничения по количеству возможных для хранения файлов отсутствуют. Поскольку файловая система FAT подвержена фрагментации, ухудшение производительности незаметно, поскольку диск обладает очень высокой скоростью доступа благодаря относительно небольшому числу твердотельных ОЗУ.

[3] Диск CRD: представляет собой карту CompactFlash, подключаемую к CR1000 посредством использования модуля хранения карт *CF* *(стр. 653)*. Для достижения наилучшей производительности карты необходимо отформатировать в файловой системе FAT32. Функция форматирования карт, предусмотренная в системе CR1000, форматирует карту в соответствии с ранее использованным на карте форматом.

8.3.1.1. Диски памяти — на плате

Диски хранения данных перечислены в таблице *Диски памяти CR1000* *(стр. 373)*. Выделение разделов в SRAM таблицы данных и диске CPU: осуществляется автоматически для использования в системе CR1000. Выделение разделов на диске USR: осуществляется по мере необходимости. Выделение разделов на диске USB: осуществляется автоматически при подключении *запоминающего устройства* Campbell Scientific *(стр. 653)*. Выделение разделов на диске CRD: осуществляется автоматически при установке карты памяти.

8.3.1.1.1. SRAM таблицы данных

Первичным хранилищем для данных измерения являются области в SRAM, выделенные под таблицы данных, как описано в таблице *Память SRAM CR1000* *(стр. 372)*. Данные измерений могут также сохраняться в виде дискретных файлов на диске USR: или USB: посредством использования команды **TableFile()**.

Систему CR1000 можно запрограммировать на хранение каждого измерения или, обобщая, на хранение обработанных значений, таких как средние, максимальные и минимальные значения, гистограммы, FFT и т. д. Данные сохраняются периодически или при определенных условиях в таблицах данных в SRAM, как требуется по условиям программы CRBasic (см. *Структура* *(стр. 123)*). Команда **DataTable()** позволяет запрограммировать размер таблицы данных. Дискретные файлы данных обычно создаются только на ПК, когда данные выгружаются с помощью *ПО поддержки регистратора данных* *(стр. 95)*.

Данные обычно стираются из этой области, когда программа передается в CR1000. Однако при использовании в меню **File Control (Управление файлами)** команды **Send (Отправить)** *(стр. 515)* или в приложении *CRBasic Editor* **команды Compile, Save and Send (Компилировать, сохранить и отправить)** *(стр. 511)* в ПО поддержки предусматриваются опции для сохранения данных при загрузке данных.

8.3.1.1.2. Диск CPU:

CPU: является диском по умолчанию, на котором хранятся программы и файлы калибровки. Он отформатирован в файловой системе FAT16, поэтому его емкость ограничена 128 файлами. Не следует хранить данные на диске

CPU:, поскольку это с большой долей вероятности повлечет за собой преждевременный выход из строя памяти.

8.3.1.1.3. Диск USR:

SRAM может быть разделена для создания диска USR: в файловой системе FAT32, аналогичного выделению разделов второго привода на жестком диске ПК. Определенные типы файлов сохраняются в USR: для резервирования ограниченной памяти CPU: для программ регистратора данных и файлов калибровки. Выделение разделов также помогает предотвратить вмешательство от таблицы данных SRAM. USR: конфигурируется с помощью параметров *DevConfig* или команды **SetStatus()** в программе CRBasic. Выделите раздел диска USR: объемом не менее 11264 байт и увеличивайте с шагом 512 байт. Если введенное значение не является кратным 512 байт, объем округляется. Максимальный размер USR: — это общий объем RAM менее 400 кБ; т. е. для CR1000 с объемом памяти 4 МБ максимальный размер USR: составляет около 3,6 МБ.

На USR: не влияет перекомпиляция программы или форматирование других дисков. Он будет сброшен до исходного состояния только в случае форматирования диска USR:, загрузки новой операционной системы или изменения размера диска USR:. Размер диска USR: изменяется вручную посредством доступа к таблице **Status (Состояние)** или загрузки программы CRBasic с другим размером диска USR:, введенным в команде **SetStatus()** или **SetSetting()**. См. раздел *Конфигурирование с помощью программы CRBasic (стр. 115)*.

Данные измерений могут также сохраняться в виде дискретных файлов на диске USR: посредством использования команды **TableFile()**. В таблице *Форматы файлов данных для команды TableFile() (стр. 378)* описываются доступные форматы файлов данных.

Примечание. Дополнительный параметр размера диска USR: в программе CRBasic замещает ручные изменения размера диска USR:. Если размер диска USR: изменяется вручную, программа CRBasic перезапускается, и запрограммированный размер диска USR: вступает в силу немедленно.

Диск USR: хранит любые типы файлов в пределах ограничений размера диска и ограничений на имена файлов. Типовые хранимые файлы включают в себя файлы изображений с камер (см. приложение Камеры), определенные конфигурационные файлы, файлы, записанные для FTP-поиска, HTML-файлы для просмотра с использованием веб-доступа, а также файлы, созданные с помощью команды **TableFile ()**. Файлы на диске USR: могут быть собраны с помощью *ПО поддержки регистратора данных (стр. 95), а именно команды* **Retrieve** *(стр. 515)* или автоматически с помощью функций вкладки **Setup File Retrieval** ПО поддержки регистратора данных.

Следите за использованием доступной памяти USR:, чтобы обеспечить достаточное пространство для хранения новых файлов. Команда **FileManage()** может использоваться в программе CRBasic для удаления файлов. Файлы также можно удалить с помощью команды **Delete (Удалить)** *(стр. 515)* в ПО поддержки регистратора данных.

Два регистра таблицы **Status (Состояние)** отслеживают использование и размер диска USR:. Оставшийся размер в байтах указывается в регистре **USRDriveFree**. Общий размер указывается в регистре **USRDriveSize**. Память, выделенная для диска USR: и менее затратная для каталожного использования, показана в окне ПО поддержки регистратора данных **File Control (Управление файлами)** *(стр. 515)*.

8.3.1.1.4. Диск USB:

Диск USB: использует *Флеш-память (стр. 516)* на запоминающем устройстве Campbell Scientific (см. приложение *Запоминающие устройства (стр. 653)*). Его основным назначением является хранение файлов данных ASCII. Данные измерений могут также сохраняться в виде дискретных файлов на диске USB:

посредством использования команды **TableFile()**. В таблице *Форматы файлов данных для команды TableFile() (стр. 378) описываются доступные форматы файлов данных.* @@@ *(стр. 378)* описываются доступные форматы файлов данных.

Внимание! При извлечении запоминающего устройства убедитесь, что светодиод на нем не горит и не мигает.

При использовании запоминающих устройств производства Campbell Scientific необходимо учитывать следующие аспекты:

- формат FAT32;
- подключение к порту **CS I/O** системы CR1000;
- извлечение только в неактивном состоянии, поскольку в противном случае это может привести к повреждению данных.

8.3.1.2. Карта памяти (диск CRD: диск) — подробная информация

Связанные темы:
• *Карта памяти (CRD: диск) — общие сведения (стр. 89)*
• *Карта памяти (CRD: диск) — подробная информация (стр. 376)*
• *Карты памяти и номера записей (стр. 466)*
• *Выходные данные: запись высокочастотные данные на карты памяти (стр. 205)*
• *Ошибки файловой системы (стр. 389)*
• *Устройства хранения данных — список (стр. 653)*
• *Примеры форматов файлов данных (стр. 379)*
• *Таблица устройств для хранения данных (стр. 373)*

В дисках CRD: используются только карты памяти CompactFlash® (CF).
Его основным назначением является хранение файлов двоичных данных.
В системе CR1000 требуется добавление слота для периферийных карт.
См. приложение *Список устройств для хранения данных (стр. 653)*.
Рекомендуется приобретать карты памяти промышленного класса производства Campbell Scientific. Использование карт потребительского класса существенно увеличивает риск потери данных.

Внимание! При вставке или извлечении карт памяти соблюдайте осторожность. Перед подсоединением или отсоединением модулей карты обязательно отключите питание системы CR1000. Извлечение карты из модуля в процессе записи на нее данных может повлечь за собой искажение данных или повреждение карты. Перед извлечением карты нажмите кнопку извлечения или выгрузки и дождитесь, пока об отключении карты не будет сообщено посредством соответствующей индикации светодиода.
Во избежание потери данных обеспечьте сбор данных с карты памяти до отправки данных в регистратор данных. Когда программа отправляется в регистратор данных, все данные на карте памяти могут быть удалены.

Модули карт памяти CF производства Campbell Scientific подключаются к периферийному порту CR1000. Каждый модуль имеет слот для карт CF типа I или типа II. На карте памяти может быть создано не более 30 таблиц данных.

Примечание. Программное обеспечение *CardConvert*, входящее в пакет *ПО поддержки регистратора данных (стр. 654) среднего и высшего уровня*, преобразует двоичные данные карты в стандартный формат данных Campbell Scientific.

Если таблица данных отправляется на карту памяти, таблица данных с идентичным именем в SRAM используется в качестве буфера для передачи данных на карту. Если карта вставлена, в таблице **Status (Состояние)** отображается размер таблицы на карте. Если карта извлечена, отображается размер таблицы в SRAM.

Если компилируется новая программа, которая отправляет данные на карту памяти, система CR1000 проверяет наличие карты и доступного места на ней для таблиц данных. Если карта отсутствует или на ней недостаточно места,

система CR1000 выдаст предупреждение о том, что карта не используется. Тем не менее программа CRBasic выполняется в любом случае, а данные сохраняются в SRAM. Если карта вставляется позднее, данные, собранные в таблице SRAM, копируются на карту.

Форматирование карт памяти

Система CR1000 может распознавать карты памяти, отформатированные в файловых системах FAT или FAT32; однако рекомендуется использовать FAT32. В противном случае могут быть недоступны некоторые функции, такие как возможность управления большим количеством файлов (> 254). Ранние версии операционной системы CR1000 предусматривали возможность форматирования карт в файловых системах FAT или FAT32. Новые операционные системы форматируют карты только в файловой системе FAT32.

Во избежание длительной компиляции на вновь отформатированной карте отформатируйте карту на ПК, затем скопируйте небольшой файл на карту, а затем удалите его (не извлекая карту из ПК). При копировании файла на вновь отформатированную карту, ПК обновляет информационный сектор. ПК обновляет информационный сектор гораздо быстрее, чем регистратор данных.

В системе FAT32 используется «информационный сектор» для хранения информации о свободных кластерах. Данный информационный сектор исключает необходимость циклического обхода системы FAT для получения информации по свободным байтам. После форматирования карты компьютером информационный сектор не обновляется автоматически. По этой причине регистратор данных при загрузке должен обнаружить доступные байты на карте перед загрузкой таблицы **Status** (Состояние). Для обхода всей файловой системы FAT карты объемом 16 ГБ может занять до 30 и более минут. Однако при последующей компиляции длительность будет гораздо меньше, поскольку информационный сектор обновил информацию о свободных байтах.

Таблица 87. Состояния карты памяти

CardStatus	*CardBytesFree*	*CompileResults*	*Светодиод*	*Ситуации*
Card OK (Карта в порядке)	> 0			Вставлена отформатированная карта, включение электропитания
	> 0		Горит постоянно в течение 20 с	Карта вставлена, но была нажата кнопка извлечения
	-1			CFM100/NL115 извлечен во время работы регистратора (выполнять данное действие запрещается)
	> 0			Программа содержит команду CardOut(). Карта вставлена до включения электропитания.
No Card Present (Карта не вставлена)	-1			Включение электропитания, карта не вставлена
	-1			Карта выгружена / физически извлечена
	-1			Регистратор запущен без CFM100 / NL115
No Card Present. Card Not Being Used (Карта не вставлена. Карта не используется)	-1	Модуль Compact Flash не обнаружен: Команда CardOut не используется.		Программа содержит команду CardOut(). CFM100/NL115 не подключен при включении электропитания.
	-1		Постоянно горит оранжевым светом	Программа содержит команду CardOut(). Карта не вставлена при включении электропитания.
Initializing Table Files! (Инициализация файлов таблицы!)	0, также регистрировалось значение –1, что не является согласованным значением		Уменьшение яркости / быстрое мигание оранжевым светом	Программа содержит команду CardOut(). Карта не вставлена при включении электропитания. Карта вставлена после включения электропитания. Если неисправности отсутствуют, значение параметра CardStatus изменится на «Card OK (Карта в порядке)», а значение параметра CardBytesFree станет >0.

8.3.2. Форматы файлов данных

Опции формата файлов данных доступны при использовании команды **TableFile()**. Данные временного ряда обладают возможностью включения

заголовка, временной метки и номера записи. См. таблицу *Команда TableFile() — форматы файлов данных (стр. 378)*. Чтобы формат был совместим со средствами построения графиков и отчетов *ПО поддержки регистратора данных (стр. 95),* обычно требуются заголовок, временные метки и номера записей. Полностью совместимые форматы обозначаются звездочкой. Более подробно форматы файлов данных рассматриваются в публикации Campbell Scientific *LoggerNet Instruction Manual (Руководство по LoggerNet),* которое доступно по адресу *www.campbellsci.com.*

Таблица 88. Команда TableFile() — форматы файлов данных

TableFile(): ***вариант формата***	***Базовый формат файла***	***Содержащиеся элементы***		
		Информация заголовка	***Временная метка***	***Номер записи***
0[1]	TOB1	✓	✓	✓
1	TOB1	✓	✓	
2	TOB1	✓		✓
3	TOB1	✓		
4	TOB1		✓	✓
5	TOB1		✓	
6	TOB1			✓
7	TOB1			
8[1]	TOA5	✓	✓	✓
9	TOA5	✓	✓	
10	TOA5	✓		✓
11	TOA5	✓		
12	TOA5		✓	✓
13	TOA5		✓	
14	TOA5			✓
15	TOA5			
16[1]	CSIXML	✓	✓	✓
17	CSIXML	✓	✓	
18	CSIXML	✓		✓
19	CSIXML	✓		
32[1]	CSIJSON	✓	✓	✓
33	CSIJSON	✓	✓	
34	CSIJSON	✓		✓
35	CSIJSON	✓		
64[2]	TOB3			

[1] Форматы, совместимые с инструментами просмотра данных и построения графиков *ПО поддержки регистратора данных (стр. 95)*.

[2] См. *Запись высокочастотных данных на карты памяти (стр. 205)* для получения более подробной информации по использованию опции ***64***.

Примеры форматов файлов данных

TOB1

Файлы TOB1 могут содержать заголовок ASCII и двоичные данные. Последняя строка в примере содержит криптографический текст, который представляет двоичные данные.

Пример:

```
"TOB1","11467","CR1000","11467","CR1000.Std.20”,"CPUifile format.CR1”,"61449","Test"
"SECONDS","NANOSECONDS","RECORD","battfivoltfiMin","PTemp"
"SECONDS","NANOSECONDS","RN","",""
"","","","Min","Smp"
"ULONG","ULONG","ULONG","FP2","FP2"
}Ÿp'   E1H¢Ÿp'    E1H>Ÿp'     E1Hªŷp'     E1H¹Ÿp'      E1H
```

TOA5

Файлы TOA5 содержат заголовок *ASCII* *(стр. 507)* и разделенные запятой данные.

Пример:

```
"TOA5","11467","CR1000","11467","CR1000.Std.20”,"CPUifile format.CR1”,"26243","Test"
"TIMESTAMP","RECORD","battfivoltfiMin","PTemp"
"TS","RN","",""
"","","Min","Smp"
"2010-12-20 11:31:30",7,13.29,20.77
"2010-12-20 11:31:45",8,13.26,20.77
"2010-12-20 11:32:00",9,13.29,20.8
```

CSIXML

Файлы CSIXML содержат информацию заголовка и данные в формате *XML* *(стр. 533)*.

Пример:

```
<?xml version="1.0" standalone="yes"?>
<csixml version="1.0">
<head>
<environment>
<station-name>11467</station-name>
<table-name>Test</table-name>
<model>CR1000</model>
<serial-no>11467</serial-no>
<os-version>CR1000.Std.20</os-version>
<dld-name>CPU:file format.CR1</dld-name>
</environment>
<fields>
<field name="battfivoltfiMin" type="xsd:float" process="Min"/>
<field name="PTemp" type="xsd:float" process="Smp"/>
</fields>
</head>
<data>
<r time="2010-12-20T11:37:45" no="10"><v1>13.29</v1><v2>21.04</v2></r>
<r time="2010-12-20T11:38:00" no="11"><v1>13.29</v1><v2>21.04</v2></r>
<r time="2010-12-20T11:38:15" no="12"><v1>13.29</v1><v2>21.04</v2></r>
</data>
</csixml>
```

CSIJSON

Файлы CSIJSON содержат информацию заголовка и данные в формате *JSON* *(стр. 519)*.

Пример:

```
"signature": 38611,"environment": {"stationfiname": "11467","tab"lefiname":
"Test','model": "CR1000","serialfino": "11467",
"osfiversion": "CR1000.Std.21.03","progfiname": "CPU:file format.CR1"},"fields":
[{"name": "battfivoltfiMin","type": "xsd:float",
"process": "Min"},{"name": "PTemp","type": "xsd:float","process": "Smp"}]},
"data": [{"time": "2011-01-06T15:04:15","no": 0,"vals": [13.28,21.29]},
{"time":  "2011-01-06T15: 04:30","no":    1,"vals":   [13.28,21.29]},
{"time":  "2011-01-06T15:04:45","no":     2, "vals":  [13.28,21.29]},
{"time":  "2011-01-06T15: 05 : 00","no":  3, "vals" : [13.28,21.29]}]}
```

Элементы формата файлов данных

Заголовок

В заголовке файла содержатся метаданные, которые описывают данные в файле. Заголовок TOA5 содержит метаданные, описанные ниже. Другие форматы данных содержат подобную информацию, если в команде **TableFile()** в программе CRBasic системы CR1000 не будет выбрана опция формата не для заголовков.

Строка 1 — источники данных

Включает следующие последовательности метаданных: тип файла, наименование станции, название модели CR1000, серийный номер CR1000, версия ОС, название программы CRBasic, программная подпись, название таблицы данных.

Строка 2 — имена полей данных

Содержит список имен отдельных полей данных. Если поле является элементом массива, к имени добавляется список разделенных запятой индексов в скобках, которые идентифицируют индекс массива. Например, переменная с именем «values» (значения) назначается массивом два-на-два, т. е.

```
Public Values(2,2)
```

будет представлена четырьмя именами полей: «values(1,1)», «values(1,2)», «values(2,1)» и «values(2,2)». Скалярные переменные (не относящиеся к массиву) не будут иметь индекса.

Строка 3 — блоки данных

Включает блоки, связанные с каждым полем в записи. Если ни один блок не запрограммирован в программе CRBasic системы CR1000, в данное поле вводится пустая строка.

Строка 4 — дескрипторы обработки данных

Записи описывают тип обработки, выполненный в системе CR1000, для формирования соответствующих данных, например, Smp обозначает образцы, Min обозначает минимальные значения. Если для поля отсутствует распознанный вид обработки, ему присваивается пустая строка. Для каждого имени поля, данного в строке 2 заголовка, предусматривается один дескриптор.

Элемент записи 1 — временная метка

Данные без временных меток обычно не имеют значения. Несмотря на это, команда **TableFile()** дополнительно включает временные метки в некоторых форматах.

Элемент записи 2 — номер записи

Номера записей дополнительно предусматриваются в некоторых форматах, чтобы обеспечить целостность данных и поле данных счета вверх для операций построения графиков. Максимальным номером записи является &hffffffff (32-битный номер), после которого последовательность номеров записей сбрасывается на нуль. Система CR1000 передает значение в ПО поддержки регистратора данных длиной 31 бит или максимум &h7fffffff, затем сбрасывается на 0. Например, если номер записи увеличивается один раз в секунду, сброс на нуль будет происходить один раз приблизительно один раз в 68 лет (да, лет).

8.3.3. Сброс CR1000

Под сбросом понимается «сброс памяти». Перед сбросом обязательно создайте резервную копию текущей конфигурации CR1000 на случай, если потребуется восстановить прежние настройки.

Следующие функциональные возможности предусматриваются для полного или выборочного сброса памяти CR1000:

- полный сброс памяти;
- сброс отправки программы;
- ручной сброс таблицы данных;
- форматирование флеш-накопителей.

8.3.3.1. Полный сброс памяти

Полный сброс памяти осуществляется, когда операционная система отправляется в CR1000 с помощью *DevConfig* или при вводе значения **98765** в таблицу **Status (Состояние)** (поле **FullMemReset**). При полном сбросе памяти выполняются следующие действия:

- стирание и форматирование диска CPU: (стирание всех программных файлов);
- стирание таблиц данных SRAM;
- стирание элементов таблицы **Status (Состояние)**;
- восстановление настроек по умолчанию;
- инициализация системных переменных;
- стирание коммуникационной памяти;
- перекомпиляция текущей программы.

Полный сброс памяти не затрагивает диск CRD: напрямую. Последовательные выгрузки программы могут, однако, стереть диск CRD:.

Операционные системы могут быть также отправлены с помощью программной функции **Send (Отправить)** в *ПО поддержки регистратора данных (стр. 95)*. Полный сброс памяти не осуществляется в этом случае. Начиная с версии 16 операционной системы CR1000, параметры и регистры в таблице **Status (Состояние)** сохраняются при отправке последующей операционной системы при помощи данного способа; таблицы данных стираются. Полагаться на данную функцию следует только с соблюдением максимальной осторожности при отправке ОС в CR1000 в удаленных или труднодоступных местах.

8.3.3.2. Сброс отправки программы

Окончательные данные в памяти (стр. 515) стираются при выгрузке пользовательских программ, если не используются опции сохранения/стирания данных. Опции сохранения/стирания данных представляются при отправке программ посредством команды **File Control Send (Управление файлами — отправить)** *(стр. 515)* и команды Compile, Save and Send (Компилировать, сохранить и отправить) в программе *CRBasic Editor (стр. 511)*. См. *Сохранение данных при отправке программы (стр. 127)*

для получения более подробной информации по сохранению/стиранию данных при отправке программы.

8.3.3.3. Ручной сброс таблицы данных

Память таблицы данных выборочно сбрасывается с помощью

- ПО поддержки — команда **Station Status (Состояние станции)** *(стр. 529)*;
- экранной клавиатуры CR1000KD: Data | Reset Data Tables (Данные | Сбросить таблицы данных).

8.3.3.4. Форматирование дисков

Диски CPU:, USR:, USB: и CRD: можно отформатировать по отдельности. При форматировании диска стираются все файлы на этом диске. Если выполняемая на текущий момент пользовательская программа находится на диске, подлежащем форматированию, то программа прекращает выполнение, и стираются все данные SRAM, связанные с программой. Форматирование диска выполняется в *ПО поддержки регистратора данных (стр. 654) — команда Format (Форматировать) (стр. 515)*.

8.3.4. Управление файлами

Как показано в таблице *Функции управления файлами (стр. 382)*, управление и контроль файлов в памяти CR1000 (программы, данные, CAL, изображения) осуществляется с помощью *ПО поддержки регистратора данных (стр. 95)*, *Web API (стр. 423)* или *CoraScript (стр. 510) системы CR1000.* Использование приложения *CoraScript* описывается в руководстве по программному обеспечению *LoggerNet*, которое можно скачать на сайте *www.campbellsci.com*. Более подробная информация по атрибутам файла, повышающих безопасность регистратора данных приводится в разделе *Безопасность (стр. 92)* .

Таблица 89. Функции управления файлами

Функции управления файлами	*Доступ обеспечивается посредством*
Отправка программ в CR1000	**Program Send (Отправка программы)**[1], **File Control Send (Управление файлами — отправить)**[2], *DevConfig*[3], клавиатура или powerup.ini, хранящийся на запоминающем устройстве или карте памяти Campbell Scientific [4,5], *web API (стр. 423)* **HTTPPut** (Sending a File to a Datalogger (Отправка файла в регистратор данных))
Настройка атрибутов программных файлов. См. Атрибуты файлов (стр. 383)	**File Control (Управление файлами)**[2]; включение электропитания с запоминающим устройством или картой памяти Campbell Scientific[5], команда **FileManage()**[6], web API **FileControl**
Отправка ОС в CR1000. Сброс настроек CR1000.	*DevConfig*[3] — вкладка **Send OS (Отправить ОС)**; *DevConfig*[3] — **вкладка File Control (Управление файлами)**; запоминающее устройство или карта памяти Campbell Scientific[5]
Отправка ОС в CR1000. Сохранить настройки CR1000.	**Send (Отправить)**[1]; *DevConfig*[3] — **вкладка File Control (Управление файлами)**; включение электропитания с запоминающим устройством или картой памяти Campbell Scientific при помощи файла default.crl[5], web API **HTTPPut** (Sending a File to a Datalogger (Отправка файла в регистратор данных))
Форматирование флеш-накопителей CR1000	**File Control (Управление файлами)**[2], включение электропитания с запоминающим устройством или картой памяти Campbell Scientific[5], web API **FileControl**
Выгрузка программ из CR1000	**Retrieve (Выгрузить)**[7], **File Control (Управление файлами)**[2], клавиатура с запоминающим устройством или картой памяти Campbell Scientific[4], web API **NewestFile**
Предписывает диспозицию (сохранить или удалить) старые файлы данных на запоминающем устройство или карте памяти Campbell Scientific	**File Control (Управление файлами)**[2], включение электропитания с запоминающим устройством или картой памяти Campbell Scientific[5], *web API (стр. 423)* **FileControl**

Таблица 89. Функции управления файлами

Функции управления файлами	*Доступ обеспечивается посредством*
Удаление файлов с флеш-накопителей	**File Control (Управление файлами)**[2], включение электропитания с запоминающим устройством или картой памяти Campbell Scientific[5], web API **FileControl**
Остановка выполнения программы	**File Control (Управление файлами)**[2], web API **FileControl**
Переименование файла	**FileRename()**[6]
Присвоение временной метки файлу	**FileTime()**[6]
Создать список файлов	**File Control (Управление файлами)**[2], **FileList()**[6], web API **ListFiles**
Создать файл данных из таблицы данных	**TableFile()**[6]
Диспетчер файлов JPEG	Экранная клавиатура CR1000KD, *LoggerNet* | *PakBusGraph*, web API **NewestFile**
Скрыть файлы	Web API **FileControl**
Кодирование файлов	Web API **FileControl**
Прекратить выполнение программы при включении электропитания	Нажать клавишу DEL на клавиатуре регистратора данных

[1] *ПО поддержки регистратора данных (стр. 95) — команда* **Program Send (Отправка программы)** *(стр. 524)*

[2] ПО поддержки регистратора данных — служебная программа **File Control (Управление файлами)** *(стр. 515)*

[3] *Служебная программа конфигурации устройства (DevConfig) (стр. 111)*

[4] Ручное управление с запоминающим устройством или картой памяти Campbell Scientific. См. *Хранилище данных (стр. 374)*

[5] Автоматическое управление с запоминающим устройством или картой памяти Campbell Scientific и файлом Powerup.ini. См. *Включение электропитания (стр. 386)*

[6] Команды CRBasic. См. *Объявления таблиц данных (стр. 540)*, *Управление файлами (стр. 382)* и *Справка для редактора CRBasic*

[7] ПО поддержки регистратора данных — команда **Retrieve (Выгрузить)** *(стр. 515)*

8.3.4.1. Атрибуты файлов

Атрибуты файлов являются функциональной возможностью программных файлов. В таблице *Атрибуты файлов CR1000 (стр. 383)* приводится список доступных атрибутов файлов, их функций, а также случаи, при которых атрибуты обычно используются. Например, программный файл, отправленный с помощью ПО поддержки — команда **Program Send (Отправка программы)** *(стр. 524)*, выполняется a) немедленно («run now» (выполнить сейчас)) и b) при выключении и включении электропитания на CR1000 («run on power-up» (выполнить при включении электропитания)). Данная функциональная возможность инициируется, поскольку команда **Program Send (Отправка программы)** *(стр. 524)* задает два атрибута файла CR1000 на программном файле, а именно: **Run Now (Выполнить сейчас)** и **Run on Power-up (Выполнить при включении электропитания)**. При совместном использовании атрибутам **Run Now (Выполнить сейчас)** и Run on Power-up (Выполнить при включении электропитания) присваивается тег **Run Always (Выполнять всегда)**.

Примечание. Активацию файла с атрибутом «выполнить при включении электропитания» можно предотвратить путем удерживания клавиши **Del** нажатой на экранной клавиатуре CR1000KD во время включения электропитания CR1000.

Таблица 90. Атрибуты файлов CR1000

Атрибуты	*Функция*	*Атрибуты для программ, отправленных в CR1000 с помощью:*
Run Always (Выполнять всегда) (выполнить при включении электропитания + выполнить сейчас)	Выполняется сейчас и при включении электропитания.	a) **Send (Отправить)** *(стр. 515)*[1] b) **File Control (Управление файлами)**[2] с отмеченными атрибутами **Run Now (Выполнить сейчас)** и Run on Power-up (Выполнить при включении электропитания). c) включение электропитания запоминающего

		устройства или карты памяти Campbell Scientific[3] с помощью команд 1 и 13(см. таблицу *Команды Powerup.ini (стр. 388)*).

Таблица 90. Атрибуты файлов CR1000

Атрибуты	*Функция*	*Атрибуты для программ, отправленных в CR1000 с помощью:*
Run on Power-up (Выполнить при включении электропитания)	Выполняется только при включении электропитания	a) **File Control (Управление файлами)**[2] с отмеченным атрибутом **Run on Power-up (Выполнить при включении электропитания).** b) включение электропитания запоминающего устройства или карты памяти Campbell Scientific[3] с помощью команды 2 (см. таблицу *Команды Powerup.ini (стр. 388)*).
Run Now (Выполнить сейчас)	Выполняется только при отправке файла в CR1000	a) **File Control (Управление файлами)**[2] с отмеченным атрибутом **Run Now (Выполнить сейчас).** b) включение электропитания запоминающего устройства или карты памяти Campbell Scientific[3] с помощью команд 6 и 14 (см. таблицу *Команды Powerup.ini (стр. 388)*). Однако, если внешнее запоминающее устройство не отключается, программа загружается еще раз с внешнего запоминающего устройства.

[1] ПО поддержки — команда **Send** *(стр. 515)*. См. справку по программному обеспечению.

[2] ПО поддержки — *File Control (Управление файлами) (стр. 515)*. См. справку по программному обеспечению и *Сохранение данных при отправке программы (стр. 127)*.

[3] Автоматическое управление при включении электропитания CR1000 с запоминающим устройством или картой памяти Campbell Scientific и файлом Powerup.ini. См. *Включение электропитания (стр. 386)*.

8.3.4.2. Диспетчер файлов

```
FilesManager := { "(" pakbus-address name-prefix number-
files ")" }.
pakbus-address := number. ; 0 < number < 4095 name-prefix :=
string.
number_files := number. ; 0 <= number < 10000000
```

Данная настройка задает количество файлов назначенного типа, которые сохраняются при получении из определенного узла. Предусматривается до четырех данных настроек. Файлы переименовываются посредством применения определенного имени файла, дополнительно измененного путем добавления серийного номера перед типом файла. Данный серийный номер используется регистратором данных для определения того, какой файл подлежит удалению, после того как серийный номер превышает заданное число файлов, подлежащих сохранению. Если число файлов равняется 0, серийный номер не добавляется. Можно использовать специальный PakBus-адрес 3210 узла, если файлы отправляются по протоколу FTP, или адрес 3211, если файлы записываются при помощи приложения CRBasic.

Примечание. Данная настройка будет работать только с файлом, имя которого не является пустой строкой.

Пример:

```
(129,CPU:NorthWest.JPG,2)
(130,CRD:SouthEast.JPG,20)
(130,CPU:Message.TXT,0)
```

В примере выше файлы *.JPG из узла 129 называются CPU:NorthWestnnn.JPG и сохраняется два файла, а файлы *.JPG из узла 130 называются CRD:SouthEastnnn.JPG, тогда как сохраняется 20 файлов. Серийный номер **nnn** начинается на **1** и содержится в последующих девяти символах. В данном примере, все файлы *.TXT из узла 130 хранятся под именем CPU:Message.Txt без добавления серийного номера.

Второй экземпляр настройки можно сконфигурировать с помощью того же

PakBus-адреса узла и того же типа файлов, при этом два файла будут записаны в соответствии с каждой из двух настроек. Например,

```
(55,USR:photo.JPG,100)
(55:USR:NewestPhoto.JPG,0)
```

сохранит два файла при каждом получении JPG-файла от узла 55. Они будут иметь имена USR:photonnn.JPG и USR:NewestPhoto.JPG. Данная функция используется, когда требуется сохранить ряд файлов, но требуется сохранить также копию одного файла, имя которого никогда не изменяется. Второму экземпляру файла также можно присвоить серийный номер, и его можно использовать, когда требуется сохранить ряд файлов на различных дисках.

Если для PakBus-адреса вводится значение 3212, это активирует сохранение информации IP-трассировки в файле. Параметр «количество файлов» задает размер файла. Файл представляет собой кольцевой файл, поэтому обеспечивается сохранение последних записей трассировки. Границу между последними и первыми записями трассировки определяется по временным меткам трассировки. Зарегистрированная информация может быть непоследовательной.

Пример:

```
(3212, USR:IPTrace.txt, 5000)
```

Синтаксическая структура создает файл на диске USR: с именем IPTrace.txt, размер которого увеличивается приблизительно до 5 кБ, а затем новые данные записываются поверх старых.

8.3.4.3. Сохранение данных

Опция сохранения данных в памяти CR1000 при отправке программы, связанная с атрибутами файлов. Данная опция применяется к таблице данных SRAM, CompactFlash® (CF) и *ПО поддержки регистратора данных (стр. 512) — кэшированным данным (стр. 511)*. В зависимости от применения может потребоваться сохранение файлов данных при загрузке программы. Если программа отправляется в CR1000 с помощью команды **Send (Отправить)** ПО поддержки регистратора данных, данные всегда удаляются перед выполнение программы. Если программа отправляется с помощью команды ПО поддержки **File Control Send (Управление файлами — Отправить)** *(стр. 515)* или *CRBasic Editor — команды* **Compile, Save and Send (Компилировать, сохранить и отправить)** *(стр. 511)*, предусматриваются опции для сохранения (не стирать) или несохранения (стирать) данных. В алгоритмах в таблице *Опции сохранения данных (стр. 386)* сводится диспозиция данных CR1000 в зависимости от выбранной опции сохранения данных.

Таблица 91. Опции сохранения данных

```
if "Preserve data if no table changed"
    keep CF data from overwritten program
    if current program = overwritten program
      keep CPU data
      keep cache data
    else
                                  erase CPU data
                                  erase cache data
    end if
end if


if "erase CF data"
    erase CF data from overwritten program
    erase CPU data
    erase cache data
end if
```

8.3.4.4. Файл Powerup.ini — подробная информация

Загрузка файла *OS* *(стр. 522)* или файла пользовательской программы в CR1000 в полевых условиях может быть весьма проблематичной задачей, особенно в экстремальных погодных условиях. Жара, холод, снег, дождь, высота, песчаная буря и расстояние до места производства работ — все эти факторы влияют на простоту процесса программирования при помощи ноутбука или планшета. Альтернативным способом является принести файл на место на легком внешнем запоминающем устройстве, например, диске *USB:* *(стр. 653)* или *CRD:* *(стр. 653)*. Шаги для загрузки новой ОС или программы CRBasic с внешнего флеш-накопителя:

1. Записать текстовый файл с именем **powerup.ini**, содержащий соответствующие команды, на внешний флеш-накопитель вместе с новой ОС или программным файлом CRBasic.
2. Подключить внешнее устройство к CR1000, а затем выключить и включить электропитание регистратора данных.

В результате такого простого процесса файл загружается в CR1000 с дополнительными атрибутами выполнения, такими как **Run Now (Выполнить сейчас), Run on Power Up (Выполнить при включении электропитания)** или **Run Always (Выполнять всегда)**, заданных для отдельных файлов. Также возможно простое копирование файла на определенный диск без атрибутов выполнения или форматирование флеш-накопителя. Опции команды для файла **powerup.ini** также обеспечивают возможность управления памятью окончательных данных на картах CF, сопоставимую с *ПО поддержки регистратора данных* *(стр. 95)* *— функция* **File Control (Управление файлами).**

Опции для **powerup.ini** также обеспечивают возможность управления памятью окончательных данных, сопоставимую с **File Control (Управление файлами)** *(стр. 515)*. Следует учитывать, что диск CRD: имеет приоритет над диском USB:.

Внимание! Перед применением файла **powerup.ini** в полевых условиях необходимо протестировать его и выполнение соответствующих процедура в лаборатории. В качестве резерва всегда имейте наготове ноутбук или мобильное устройство (с установленным ПО поддержки регистратора данных) при работе в труднодоступных или дорогостоящих для доступа местах.

Команды Powerup.ini включают следующие функции:

- Отправка программ в CR1000
- Дополнительно настраиваемые атрибуты выполнения программных файлов CR1000
- Отправка ОС в CR1000

- Форматирование флеш-накопителей
- Удаление файлов данных, связанных с ранее выполнявшейся программой

При подаче электропитания на CR1000 в системе осуществляется поиск файла powerup.ini и выполнение команды (команд) перед компиляцией программы. Powerup.ini выполняет три операции:

1. Копирует программный файл на флеш-накопитель.
2. Дополнительно задает атрибут выполнения (**Run Now (Выполнить сейчас), Run on Power-up (Выполнить при включении электропитания)** или **Run Always (Выполнять всегда)**) для программного файла.
3. Дополнительно удаляет файлы данных, сохраненные от перезаписанной (предыдущей) программы.
4. Форматирование заданного диска.

Выполнение файла **powerup.ini** имеет приоритет в ходе процесса включения электропитания CR1000. Несмотря на то, что **powerup.ini** задает атрибуты файла для загруженных программ, его присутствие на диске не позволяет данным атрибутам файла управлять процессом включения электропитания. Во избежание путаницы необходимо либо удалить внешний диск, на котором присутствует файл **powerup.ini**, либо удалить файл после завершения операции включения электропитания.

8.3.4.4.1. Создание и редактирование Powerup.ini

Powerup.ini создается в текстовом редакторе на ПК, затем сохраняется на флеш-накопителе системы CR1000. Файл можно сохранить на флеш-накопителе, наряду с операционной системой или пользовательским программным файлом, с помощью *ПО поддержки регистратора данных (стр. 654) — команда* **File Control | Send (Управление файлами | Отправить)** *(стр. 515)* .

Примечание. Некоторые текстовые редакторы (такие как MicroSoft® WordPad®) добавляют информацию заголовка к файлу powerup.ini, что приводит к его прерыванию его выполнения. Проверьте текст файла powerup.ini в CR1000 с помощью экранной клавиатуры CR1000KD, чтобы проверить, какая конкретно информация будет доступна в системе CR1000.

К файлу можно добавить комментарии, поставив перед ними символ одиночной кавычки ('). Все текстовые записи, следующие за знаком комментария на данной строке, игнорируются.

Синтаксис

Синтаксическая конструкция файла **powerup.ini** имеет следующий вид:

```
Command,File,Device
```

где

- ***Command (Команда)*** — это одна из числовых команд в таблице *Команды Powerup.ini (стр. 388).*
- ***File (Файл)*** — сопутствующая операционная система или пользовательский программный файл. Длина имени может составлять до 22 символов.
- ***Device (Устройство)*** — это флеш-накопитель системы CR1000, на который копируется сопутствующая операционная система или пользовательский программный файл (обычно CPU:). Если устройство не указано или выбрана недействительная опция, устройством по умолчанию будет CPU:. Используйте то же обозначение диска, что и транспортировочное внешнее устройство, если предпочтительно, чтобы файл не копировался.

Таблица 92. Команды и приложения файла Powerup.ini

Команда	*Описание*	*Приложения*
1[1]	Run Always, preserve data (Выполнять всегда, сохранить данные)	Копирует заданную программу на указанный диск и задает атрибут выполнения программы **Run Always (Выполнять всегда).** Данные карте CF от ранее выполнявшейся программы будут сохранены.
2	Run on Power-up (Выполнить при включении электропитания)	Копирует заданную программу на указанный диск. Программе, заданная в команде 2, будет присвоен атрибут **Run Always (Выполнять всегда)**, если не будет использована команда **6** или **14** для настройки отдельной программы с атрибутом **Run Now (Выполнить сейчас).**
5	Формат	Форматирует указанный диск.
6[1]	Run now, preserve data (Выполнить сейчас, сохранить данные)	Копирует заданную программу на указанный диск и задает атрибут выполнения программы **Run Now (Выполнить сейчас).** Данные карте CF от ранее выполнявшейся программы будут сохранены.
7	Копирует файлы на указанный диск без атрибутов выполнения. Используется для копирования *Include (Включить)* *(стр. 518)* или программных файлов поддержки на диск CPU: перед копированием программного файла для выполнения.	Копирует заданный файл на указанный диск без атрибутов выполнения.
9	Load OS (File = .obj) (Загрузить ОС (Файл = .obj))	
13	Run Always, erase data (Выполнять всегда, стереть данные)	Копирует заданную программу на указанный диск и задает атрибут выполнения программы **Run Always (Выполнять всегда)**. Данные карте CF от ранее выполнявшейся программы будут удалены.
14	Run now, erase files (Выполнить сейчас, стереть файлы)	Копирует заданную программу на указанный диск и задает атрибут выполнения **Run Now (Выполнить сейчас)**. Данные карте CF от ранее выполнявшейся программы будут удалены.

[1] Данные и переменные можно сохранить при использовании команды **PreserveVariables()** в программе CRBasic в сочетании с командами ***1*** и ***6***.

Пример файлов Powerup.ini

Таблица 93. Пример файла Powerup.ini. Code Format and Syntax (Формат кода и синтаксические правила)

```
Формат кода и синтаксические правила

Команда = числовая команда включения электропитания
Файл = файл, связанный с действием
Устройство = устройство, на которое копируется файл. Значения по умолчанию для CPU:

Command,File,Device
13,Write2CRD_2.cr1,cpu:
```

Таблица 94. Пример файла Powerup.ini. Run Program on Power-up (Выполнение программы при включении электропитания)

```
Копировать программный файл pwrup.cr1 с внешнего диска в CPU:
Файл будет выполнен только при последующем включении электропитания системы CR1000.
2,pwrup.cr1,cpu:
```

Таблица 95. Пример файла Powerup.ini. Форматировать диск USR Drive (Форматировать диск USR:)

```
Форматировать диск USR:
5,,usr:
```

Таблица 96. Пример файла Powerup.ini. Send OS on Power-up (Отправить ОС при включении электропитания)

```
Загрузить файл операционной системы (.obj) во флеш-память как новую ОС.
9,CR1000.Std.28.obj
```

Таблица 97. Пример файла Powerup.ini. Выполнить программу с диска USB Drive (Выполнить программу с диска USB:)

```
Программный файл находится на внешнем диске USB:.
Не копировать программный файл с диска USB:
Выполнять программу всегда, стереть данные
13,toobigforcpu.cr1,usb:
```

Таблица 98. Пример файла Powerup.ini. Run Program Always, Erase Data (Выполнять программу всегда, стереть данные)

```
Выполнять программный файл всегда, стереть данные.
13,pwrup_1.cr1,cpu:
```

Таблица 99. Пример файла Powerup.ini. Run Program Now, Erase Data (Выполнить программу сейчас, стереть данные)

```
Выполнить программный файл сейчас, стереть данные сейчас.
14,run.cr1,cpu:
```

Выполнение файла Power-up.ini

После обработки файла **powerup.ini** выбор программы CR1000, подлежащей выполнению, осуществляется в соответствии со следующими правилами:

- Если программа для немедленного выполнения изменяется, то это программа, которая выполняется.
- Если программа для немедленного выполнения не изменяется, однако меняется программа для выполнения при включении электропитания, выполняется новая программа для выполнения при включении электропитания.
- Если ни программа для немедленного выполнения, ни программа для выполнения при включении электропитания не изменяются, то выполняется предыдущая программа для выполнения при включении электропитания.

8.3.4.5. Управление файлами — вопрос-ответ

В: Как скрыть программный файл на CR1000 без использования команды CRBasic **FileManage()**?

А: Воспользуйтесь *CoraScript* *(стр. 510)* *— команда* **File-Control (Управление файлами)** или *web API* *(стр. 423)* *— команда* **FileControl.**

8.3.5. Имена файлов

Максимальная длина имени файла, который можно хранить, выполнять как программу или передавать по протоколу FTP в систему CR1000, составляет 59 символов. Если имя файла превышает длину в 59 символов, отображается ошибка **Invalid Filename (Недействительное имя файла)**. Если хранится несколько файлов, каждый из которых имеет длинное имя файла, память, выделенная под корневой каталог, может быть превышена до того, как будет превышена фактическая память для хранения файлов. В этом случае отобразится ошибка «недостаточно ресурсов или память заполнена».

8.3.6. Ошибки файловой системы

Таблица *Коды ошибок файловой системы* *(стр. 390)* содержит список кодов ошибок, связанных с файловой системой CR1000. Ошибки могут происходить при попытке доступа к файлам на любых доступных дисках. Все эти случаи

достаточно редки, однако с большей степенью вероятности происходят при использовании диска CRD:.

Таблица 100. Коды ошибок файловой системы

Код ошибки	*Описание*
1	Недействительный формат
2	Ошибка функциональных возможностей устройства
3	Невозможно выделить память для работы файла
4	Превышено максимальное число доступных файлов
5	Записи о файле отсутствуют в каталоге
6	Произошло изменение диска
7	Часть пути (подкаталог) не найден
8	Файл достиг EOF
9	Обнаружен поврежденный кластер
10	Буфер файлов недоступен
11	Слишком длинное имя файла или имеет недопустимые символы
12	Путь к файлу не является каталогом
13	Разрешение доступа, открытие DIR или LABEL как файла или попытка открыть файла как DIR или mkdir существующего файла
14	Открытие для записи файла только для чтения
15	Недостаточно места на диске (невозможно выделить новый кластер)
16	Корневой каталог переполнен
17	Недействительный ptr (указатель) файла или устройство не инициализировано
18	Устройство не поддерживает данную операцию
19	Предоставлен недействительный аргумент функции
20	Поиск границ за пределами файла
21	Попытка выполнения команды mkdir для существующего каталога
22	Недействительная подпись сектора раздела
23	Непредвиденный идентификационный байт системы в записи раздела
24	Путь уже открыт
25	Доступ к неинициализированному логическому диску
26	Попытка переименования в устройствах
27	Подкаталог не пуст
31	Попытка записи на диск, защищенный от записи
32	Отсутствует ответ от диска (возможно, открыта дверца)
33	Метка адреса или сектор не найден
34	Обнаружен поврежденный сектор
35	Ошибка пересечения границы памяти DMA

Таблица 100. Коды ошибок файловой системы	
Код ошибки	***Описание***
36	Различные ошибки ввода-вывода
37	Запрошен диаметр трубы, равный 0
38	Ошибка освобождения памяти (relmem)
39	Нечитаемые секторы FAT (все копии)
40	Поврежденный сектор BPB
41	Таймаут ожидания доступа к файловой системе
42	Ошибка контроллера
43	Полный путь к файлу превышает значение MAX PATHNAME

8.3.7. Память — вопрос-ответ

B: Может ли пользователь создать программу, размер которой не позволяет разместить ее на диске CPU: (>100k), и запустить ее с диска CRD: (карта памяти)?

A: Программу невозможно запустить с карты памяти. Тем не менее, очень большая программа (размер которой не позволяет разместить ее на диске CPU:) может быть скомпилирована в основной памяти CR1000 с карты, если двоичная форма скомпилированной программы не превышает доступный объем *основной памяти* *(стр. 370)*.

8.4. Сбор и обмен данными — подробная информация

Связанные темы:
• *Сбор и обмен данными — краткое руководство (стр. 45)*
• *Сбор и обмен данными — общие сведения (стр. 88)*
• *Сбор и обмен данными — подробная информация (стр. 391)*
• *Периферийное оборудование для сбора и обмена данными — списки (стр. 651)*

Телекоммуникация, в контексте работы CR1000, представляет собой перемещение информации между CR1000 и другим вычислительным устройством, обычно ПК. Информацией могут быть данные, программы, файлы или команды управления.

Телекоммуникационным системам требуется три основных компонента: аппаратное обеспечение, несущий сигнал и протокол. Например, распространенным способом связи с CR1000 является программное обеспечение *PC200W* с применением COM-порта ПК. В данном примере аппаратным обеспечением выступает COM-порт ПК, порт **RS-232** системы CR1000 и кабель для последовательной передачи данных. Несущим сигналом является RS-232, а протокол — PakBus®. Из этих трех компонентов наиболее часто придется выбирать только аппаратное обеспечение, поскольку несущий сигнал и протокол являются очевидными в большинстве случаев применения.

Системам обычно требуется один тип аппаратного обеспечения и несущего сигнала. В некоторых применениях, однако, требуются гибридные системы, состоящие из двух и более типов аппаратного обеспечения и несущих сигналов.

Просим проконсультироваться с инженером по приложениям Campbell Scientific при конфигурировании телекоммуникационной системы.

Краткое описание программного обеспечения для поддержки телекоммуникационных устройств и протоколов приводится в приложении *ПО поддержки* *(стр. 654)*. Специально следует отметить приложение *Network Planner*, являющееся клиентом *LoggerNet*, предназначенным для упрощения процесса настройки конфигурации телекоммуникационных сетей PakBus.

8.4.1. Протоколы

CR1000 обменивается данными с *ПО поддержки регистратора данных (стр. 95)* и другими *регистраторами данных Campbell Scientific (стр. 645)* посредством протокола *PakBus (стр. 522)*. См. раздел *Альтернативная связь — подробная информация (стр. 407)* для получения информации по другим поддерживаемым протоколам, таким как TCP/IP, Modbus и др.

8.4.2. Сохранение полосы пропускания

Некоторые телекоммуникационные услуги, такие как спутниковые сети, могут быть очень дорогими для отправки и получения информации. Среди наилучших способов по сокращению затрат можно выделить следующие:

- Присваивать **Общий доступ** только тем переменным, которые должны быть общедоступными.
- Соблюдать устоявшиеся правила использования строковых переменных и размера строковых переменных. Размер строковых переменных должен соответствовать необходимому, но не превышать его; учитывать, что минимальным размером является 24 байта. Присваивать строковым переменным **Общий доступ** и сводить строковые переменные в таблицы данных только в случае необходимости.
- Если используются команды **GetVariables() / SendVariables()** для отправки значений между регистраторами данных, поместить данные в массив и использовать одну команду для получения множества значений. Использование одной команды для получения 10 значений из массива и полосы из 10 намного более эффективно (требуется всего 1 транзакция), чем использование 10 команд для получения 10 отдельных значений (требуется 10 транзакций).
- Устанавливать CR1000 в качестве маршрутизатора PakBus только при необходимости. Если система CR1000 является маршрутизатором и подключается к другому маршрутизатору, например LoggerNet, она обменивается информацией по маршрутизации с данным маршрутизатором и, вероятно (в зависимости настроек данной сети), с другими маршрутизаторами в сети.
- Установить маяки PakBus и точно определить интервалы. Например, не требуется определять маршруты каждые пять минут, если предполагаемые сеансы связи будут осуществляться только каждые 6 часов.

8.4.3. Инициация связи (обратный вызов)

Сеансы связи обычно инициируются ПК. После установления соединения ПК выдает команды на отправку программ, установку времени, сбор данных и т. д. Ввиду того, что сбор данных осуществляется под управлением ПК, несколько ПК может получить доступ к CR1000, не нарушая целостности данных. PakBus® обеспечивает возможность нескольким ПК обмениваться данными с CR1000 одновременно, если установлены соответствующие телекоммуникационные сети.

Обычно ПК инициирует связь с CR1000 при помощи *ПО поддержки регистратора данных (стр. 654)*. Однако некоторым приложениям требуется, чтобы система CR1000 выполнила обратный вызов ПК (инициировала связь). Данная функция называется «обратным вызовом». Специальные функции *LoggerNet (стр. 654)* позволяют ПК принимать вызовы от CR1000.

Например, если садовод желает получать аварийный сигнал в случае заморозков, CR1000 может связаться с ним посредством вызова ПК, отправки электронного письма, текстового сообщения, пейджингового сообщения или вызова с синтезированной речью по телефону. Обратный вызов использовался в таких областях применения, как Ethernet, телефонная связь по наземным линиям, мобильная связь и прямое соединение. Обратный вызов с помощью телефона подробно описан в *Справке по редактору CRBasic* (поисковый термин «обратный вызов»). Для получения более подробной информации по

другим доступным функциям Callback (Обратный вызов) см. руководства по различному телекоммуникационному оборудованию, в которых рассматриваются функции обратного вызова. Для получения последней информации по приложениям обратного вызова обратитесь к инженеру по приложениям компании Campbell Scientific.

Внимание! Если используется коммуникационный порт ComME с протоколами, отличными от PakBus, входящие символы могут быть искажены в результате использования CS I/O для обмена данными SDC. Для связи PakBus используется низкоуровневый протокол (последовательность «пауза/окончание/готовность») для остановки входящих данных при осуществлении SDC.
Протоколы обмена данными, отличные от PakBus, включают следующие: TCP/IP, ModBus, DNP3 и универсальный CS I/O на базе CRBasic.
И хотя обычно не замечается, но короткий всплеск связи SDC имеет место при включении электропитания и в других случаях, когда регистратор данных перезапускается, например при компилировании программы или изменении настроек, для вступления в силу которых требуется перекомпиляция. Данная активность осуществляется, когда регистратор данных осуществляет опрос, чтобы проверить доступна ли экранная клавиатура CR1000KD.
Когда функции *DevConfig* и *PakBus Graph* получают параметры, система CR1000 осуществляет опрос, чтобы определить, какие устройства SDC подключены. Результаты опроса можно просмотреть в таблицах параметров *DevConfig* и *PakBusGraph*. Запросы SDC отправляются вне зависимости от того, подключены устройства SDC или нет.

8.5. Связь PakBus® — подробная информация

Связанные темы:
• *Связь PakBus® — общие сведения* *(стр. 88)*
• *Связь PakBus® — подробная информация* *(стр. 393)*
• *Связь PakBus® — команды* *(стр. 584)*
• *Руководство по работе в сети PakBus* ((доступно на сайте *www.campbellsci.com/manuals* (*http://www.campbellsci.com/manuals*))

CR1000 обменивается данными с компьютерами или другими регистраторами данных Campbell Scientific по протоколу PakBus. Протокол PakBus является запатентованным телекоммуникационным протоколом и по своей структуре сходен с межсетевым протоколом Internet. PakBus обеспечивает возможность бесшовного объединения регистраторов данных Campbell Scientific и телекоммуникационного периферийного оборудования в сети PakBus.

Читать далее. Данный раздел приводится в виде введения в обмен данными по протоколу PakBus. Более подробная информация приводится в приложениях *Одноранговая связь PakBus* *(стр. 584)* и Состояние/Настройки/DTI: информация о PakBus и *Руководство по сетевому взаимодействию по протоколу PakBus,* которое можно скачать на сайте *www.campbellsci.com.*

8.5.1. PakBus-адреса

Регистраторам CR1000 присваиваются адреса PakBus® **1** в виде заводских настроек. Организация сетей с более чем несколькими станциями должна предусматривать схему адресации, которая гарантирует присвоение уникальных адресов всем узлам. Одним из способов, который продемонстрирован на рисунке *Сетевая адресация PakBus* *(стр. 394)*, является назначение однозначных адресов узлам первого уровня, двузначных — узлам второго уровня, трехзначных — узлам третьего уровня и т. д. Следует учитывать, что адрес каждого узла на ответвлении начинается с одной и той же цифры. Устройствам, таким как ПК, с адресами, превышающими 4000, предоставляется особый административный доступ к сети.

PakBus-адреса задаются с помощью *DevConfig, PakBusGraph*, таблицы **Status (Состояние)** CR1000 или с помощью экранной клавиатуры CR1000KD.

DevConfig (служебная программа конфигурации устройства) является основным редактором настроек. Ему требуется аппаратное последовательное соединение с ПК. Кроме того, он обеспечивает резервирование настроек на жестком диске ПК. *PakBusGraph* работает с использованием линии связи и предназначен для изменения настроек, но не предусматривает возможность резервирования.

Внимание! Особое внимание необходимо уделять при изменении адресов PakBus® посредством приложения *PakBusGraph* или таблицы **Status (Состояние)**. Если адрес меняется на неизвестное значение, возможно, потребуется выезд на место с ноутбуком с установленным приложением *DevConfig* для определения неизвестного адреса.

8.5.2. Узлы: оконечные узлы и маршрутизаторы

- Сеть PakBus® может состоять из двух до 4093 связанных узлов.
- В сети допускается наличие одного или более оконечных узлов и маршрутизаторов.
- Оконечные узлы представляют собой измерительные устройства в конце ответвления сети PakBus.
  - Оконечные узлы могут быть соединены с любым маршрутизатором.
  - Оконечный узел не может маршрутизировать пакеты, но может их формировать или получать.
- Маршрутизаторы представляют собой измерительные или телекоммуникационные устройства, которые маршрутизируют пакет на другие маршрутизаторы или оконечные узлы.
  - Маршрутизаторы могут быть маршрутизаторами ответвлений. Маршрутизаторам ответвлений известны только соседние центральные маршрутизаторы, маршрутизаторы в маршруте к центральным маршрутизаторам и маршрутизаторы на один уровень выше в сети.
  - Маршрутизаторы могут быть центральными маршрутизаторами. Центральным маршрутизаторам известна вся сеть. ПК, на котором выполняется *LoggerNet*, обычно является центральным маршрутизатором.
  - Маршрутизаторами могут быть регистраторы данных с поддержкой функций маршрутизатора или коммуникационные устройства.

По умолчанию регистратор CR1000 имеет настройку в качестве оконечного узла. Его можно настроить в качестве маршрутизатора, установив для параметра **IsRouter** в таблице **Status (Состояние)** значение **1** или **True**. Сеть, представленная на рисунке *Сетевая адресация PakBus (стр. 394)*, содержит шесть маршрутизаторов и восемь оконечных узлов.

8.5.2.1. Конфигурация маршрутизаторов и оконечных узлов

В приложении «Маршрутизаторы и оконечные узлы» содержится таблица доступных маршрутизаторов и оконечных узлов PakBus®. *LoggerNet* по умолчанию настроен как маршрутизатор и может обеспечивать связь между регистраторами данных.

Рисунок 102. Сетевая адресация PakBus

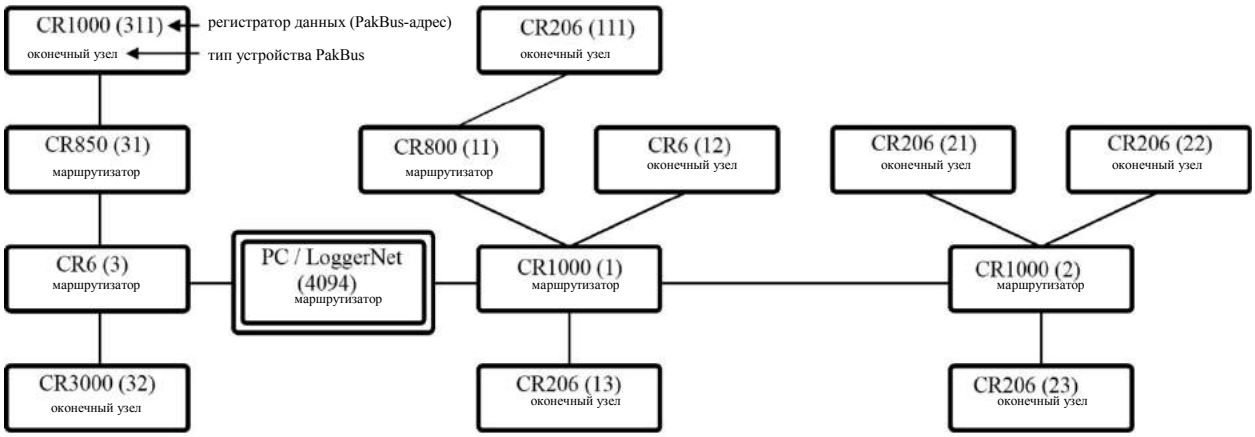


Таблица 101. Конфигурация маршрутизаторов и оконечных узлов PakBus

Сетевое устройство	*Описание*	*Оконечный узел PakBus*	*Маршрутизатор PakBus*	*Совместимость с PakBus*	*Видимость*
CR200X	Регистратор данных	•			
Порт CR6 CS I/O	Регистратор данных	•	•		
CR800	Регистратор данных	•	•		
CR1000	Регистратор данных	•	•		
CR3000	Регистратор данных	•	•		
CR5000	Регистратор данных	•	•		
LoggerNet	Программное обеспечение		•		
CR6 Ethernet-порт	Сетевое подключение				
NL100	Сетевое подключение к последовательному порту		•		•
NL115	Сетевое подключение к периферийному порту[1]				•
NL120	Сетевое подключение к периферийному порту[1]				•
NL200	Сетевое подключение к последовательному порту				
NL240	Подключение к беспроводной сети				
MD485	Многоточечная линия связи			•	•
RF401, RF430, RF450	Радио		•	•	•
CC640	Камера	•			
SC105	Последовательный интерфейс				•
SC32B	Последовательный интерфейс				•
SC932A	Последовательный интерфейс				•
COM220	Телефонный модем				•
COM310	Телефонный модем				•
SRM-5A	Модем ближней связи				•

8.5.3. Соединение узлов PakBus: обнаружение соседей

Новые термины (см. *Узлы: оконечные узлы и маршрутизаторы (стр. 394)*):

- узел
- соединение
- сосед
- фильтры соседей
- hello
- hello-обмен
- hello-сообщение
- hello-запрос
- CVI
- маяк

Чтобы образовать сеть, узлы должны установить соединения с соседями (соседи — это смежные узлы). Соединения устанавливаются в ходе процесса обнаружения. Обнаружение происходит, когда узлы обмениваются приветствиями. Hello-обмен происходит в процессе hello-сообщения между двумя узлами.

8.5.3.1. Hello-сообщение

Hello-сообщение является двусторонним обменом между узлами с целью установления соседского соединения. Hello-сообщение отправляется в ответ либо на сигнал маяка или на hello-запрос, либо на оба данных запроса.

8.5.3.2. Сигнал маяка

Сигнал маяка — это односторонняя передача от узла с заданным интервалом для уведомления всех узлов в пределах «слышимости» о том, что hello-сообщение может быть отправлено. Если узел предпринимает попытку определения себя в качестве соседа относительно узла-маяка, то он отправляет hello-сообщение узлу-маяку. Узлы, которые уже определены как соседи, не отвечают на сигнал маяка.

8.5.3.3. Hello-запрос

Hello-запрос — это односторонняя передача. Все узлы, получающие hello-запрос (существующие и потенциальные соседи), передают hello-сообщение для установления или повторного установления соседских отношений с узлом, передавшим запрос.

8.5.3.4. Список соседей

Устройства PakBus в сети можно настроить посредством списка соседей. CR1000 отправляет со случайным интервалом *1 hello-сообщение каждому узлу из списка*, CVI *(стр. 511)* которого закончился. Если узел отвечает, происходит обмен hello-сообщениями и узел становится соседом.

Фильтры соседей указывают, какие узлы являются соседями, и направляют пакеты по маршрутам, заданным администратором сети. *LoggerNet,* являющийся узлом PakBus, получает фильтр соседей из информации о соединении в карте устройств *LoggerNet Setup*.

[1] Интервал представляет собой случайное число секунд между интервалом и длительностью, в два раза превышающей интервал, где интервал равен CVI (если отличается от нуля) или 300 секундам, если для параметра CVI установлено нулевое значение.

8.5.3.5. Регулирование соединений

PakBusGraph, клиент программы *LoggerNet*, особенно полезен при тестировании и настройке маршрутов PakBus. Пути, установленные посредством сигналов маяка, могут быть резервными и могут различаться по надежности. Резервные пути могут предоставить резервные соединения в случае сбоя основных путей. Резервные и ненадежные пути можно исключить посредством активации фильтров соседей в различных узлах и деактивации

некоторых маяков.

8.5.3.6. Поддержка соединений

Поддержка соединений осуществляется посредством *CVI (стр. 511)*. CVI может быть задан в каждом узле с помощью параметра **Verify Interval (Интервал определения)** в программе *DevConfig* (**ComPorts Settings**). Применяются следующие правила:

Примечание. Во время hello-сообщения параметр CVI должен быть передан между двумя соседями. Передаваемый CVI меньше параметра CVI первого узла и составляет 6/5 от параметра CVI соседа.

- Если **Verify Interval (Интервал определения) = 0**, то CVI = 2,5 x **Beacon Interval (Интервал между сигналами маяка)**
- Если **Verify Interval (Интервал определения) = 60**, то CVI = 60 секунд
- Если **Beacon Interval (Интервал между сигналами маяка) = 0** и **Verify Interval (Интервал** определения) = 0, то CVI = 300 секунд
- Если маршрутизатор или ведущее устройство не получает от соседа одного CVI, то он снова начинает отправку hello-сообщения этому узлу со случайным интервалом.

При присваивании значения параметру **Verify Interval (Интервал определения)** пользователям необходимо исходить из временных рамок, применимых для нормальной связи, таких как запланированный сбор данных *LoggerNet* или обмен данными между регистраторами данных. Идея заключается в том, чтобы не позволить истечь CVI до истечения периода нормальной связи. Если CVI заканчивается, устройства начинают обмен hello-сообщениями для восстановления соседских взаимосвязей, что влечет за собой увеличение сетевого трафика.

8.5.4. Диагностика сетей PakBus

Для выполнения диагностики в сетях PakBus предусматриваются различные инструменты и методы.

8.5.4.1. Целостность соединения

Благодаря обнаружениям посредством сигналов маяка или фильтра соседей, соединения устанавливаются и проверяются с применением относительно небольших пакетов данных (hello-сообщений). Однако, если соединения используются для постоянной связи, используются сообщения большей длины. Следовательно, соединение может быть достаточно надежным для обнаружения посредством hello-сообщений, но ненадежным в случае более длинных сообщений или пакетов. Данное условие наиболее распространено в беспроводных сетях, особенно если максимальный размер пакета > 200.

Связь по протоколу PakBus в граничных соединениях можно улучшить путем уменьшения размера пакетов PakBus с помощью параметра **Max Packet Size (Макс. размер пакета)** в программе *DevConfig на вкладке* **Advanced**. Наилучший результат достигается, если максимальный размер пакета уменьшается в обоих узлах.

8.5.4.1.1. Автоматическая регулировка размера пакета

Транзакция получения файлов BMP5 позволяет клиенту BMP5 (*LoggerNet*) задавать размер следующего фрагмента файла, который отправляет регистратор CR1000.

Примечание. PakBus использует транзакцию получения файлов для получения определений таблицы из регистратора данных.

Ввиду того, что *LoggerNet* должен задавать размер для следующего фрагмента файла, он использует любые ограничения размера, применимые к данному соединению.

По этой причине размер ответов на команды получения файлов, которые отправляет регистратор CR1000, определяется параметром **Max Packet Size (Макс. размер пакета)** для регистратора данных, а также любым из его родительских параметров в карте сети *LoggerNet*. Следует учитывать, что данный расчет также принимает во внимание коэффициент ошибок для устройств в соединении.

Транзакция сбора данных BMP5 не предоставляет клиенту никаких способов для задания предела размера ответного сообщения. Это основная причина, по которой существует параметр **Max Packet Size (Макс. размер пакета).** Регистратор CR1000 может просмотреть данный параметр до точки, в которой он формирует ответное сообщение, и отсечь количество данных, которое он обычно отправляет, если параметр ограничивает размер сообщения.

8.5.4.2. Ping (PakBus)

Целостность соединения можно проверить посредством следующей процедуры с использованием приложения *PakBusGraph — команда* **Ping Node (Пинговать узел)**. Узлы можно пинговать пакетами размером 50, 100, 200 или 500 байт.

Примечание. Не используйте пакеты длиной свыше 90 байт при пинговании радиомодемов и беспроводных регистраторов данных с мощностью сигнала 100 мВт. См. приложение *Периферийное оборудование для сбора и обмена данными — списки* *(стр. 651)***.**

Пингование с десятью повторениями для каждого размера пакета характеризует соединение. Перед пингованием весь другой сетевой трафик (запланированный сбор данных, проверка часов и т. п.) должен быть временно отключен. Начните пингование с первого уровня соединений (соседей) от маршрутизатора ПК/*LoggerNet*, затем переходите к узлам, находящимся на удалении более одного перехода. В таблице *Шкала пропускной способности соединения в сети PakBus* *(стр. 398)* приводится шкала пропускной способности соединения.

Таблица 102. Шкала пропускной способности соединения в сети PakBus

Отправленный по команде Ping пакет размером 500 байт	*Количество успешных откликов*	*Состояние соединения*
10	10	отлично
10	9	хорошо
10	7–8	приемлемо
10	< 7	предельно низкое

8.5.4.3. Поток трафика

При увеличенном трафике (большое число узлов и/или частый сбор данных) интервалы между сигналами маяка должны быть максимально большими. Длительные интервалы между сигналами маяка сокращают вероятность столкновения с другими пакетами и результирующими повторными попытками. Минимальным рекомендуемым значением параметра **Beacon Interval (Интервал между сигналами маяка)** является **60** секунд. Если трафик на линии связи большой, рассмотрите возможность применения интервалов между сигналами маяка длительностью в несколько минут. Если требуется повышенная пропускная способность, увеличьте полосу пропускания за счет внедрения ответвительных маршрутизаторов или отмены всех маяков и настройки фильтров соседей.

8.5.5. Конфигурация карты сети LoggerNet

Как видно из рисунка *Схема одноуровневого подчинения* *(стр. 399)* и рисунка *Схема многоуровневого подчинения* *(стр. 399)*, существенным элементом карты устройств сети PakBus в *LoggerNet* является **PakBusPort**. После

добавления корневого порта (COM, IP или другого) добавьте PakBusPort и регистраторы данных.

Рисунок 103. Схема одноуровневого подчинения

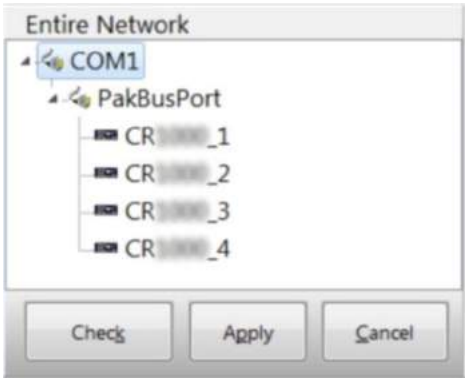
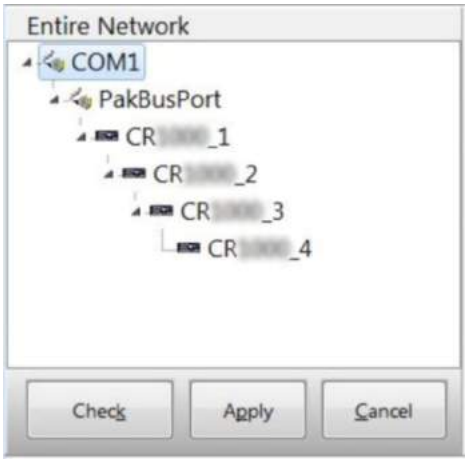


Рисунок 104. Схема многоуровневого подчинения



Различие между двумя конфигурациями заключается в том, что при одноуровневом подчинении маршрутизатору назначаются статические маршруты, при которых все регистраторы данных являются соседями относительно сервера. При многоуровневом подчинении настраиваются статические маршруты, при котором «CR1000» настраивается в качестве соседа, а «CR1000_2», «CR1000_3» и «CR1000_4» настраиваются на использование «CR1000» в качестве маршрутизатора. Создание более сложной иерархии, если и допускается, не имеет смысла в рамках сети PakBus, поскольку протокол PakBus не позволяет прописывать коммуникационный путь целиком. Можно задать адрес маршрутизатора только для первого перехода.

В рамках сервера динамически обнаруженные маршруты имеют приоритет над статическими маршрутами, поскольку после настройки сети связь будет работать бесперебойно. Несмотря на это, наличие правильного начального статического маршрута зачастую критически важно, поскольку попытка вызова ложного соседа может привести к таймауту до обнаружения маршрутизации от реального соседа.

Говоря другими словами, используйте многоуровневую схему, если для обеспечения связи требуется применение маршрутизаторов. Структура схемы служит для того, чтобы не допустить прямого соединения *LoggerNet* с CR1000_2 и CR1000_3, и подразумевает применение ограниченных маршрутов, которые, скорее всего, будут заданы в маршрутизаторах установленными пользователем фильтрами соседей. Это приводит к предположению, что маяки *LoggerNet* выключены. В противном случае, если для маяков *LoggerNet* установлен адрес по умолчанию 4094, не будет срабатывать фильтр соседей любого узла в зоне досягаемости.

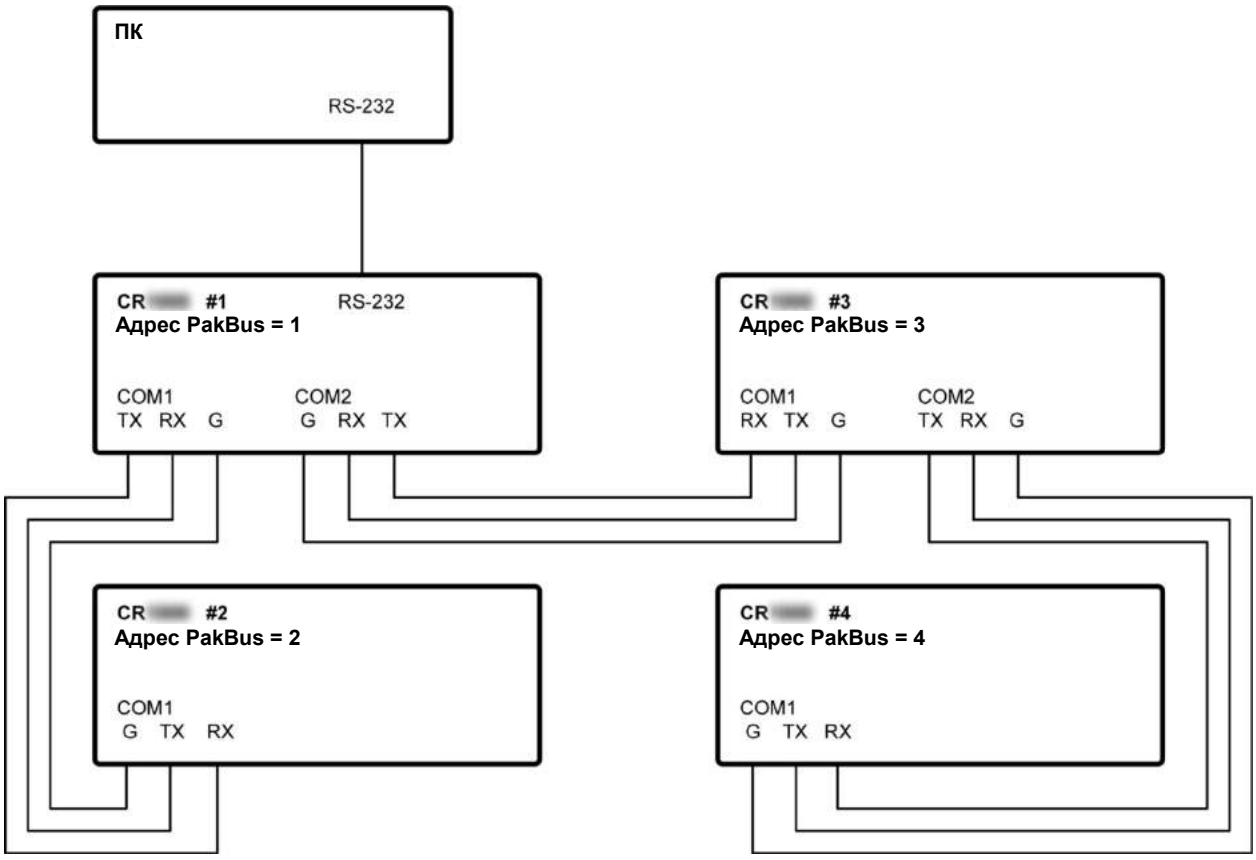
8.5.6. Пример ЛВС на основе протокола PakBus

Для демонстрации сетевого взаимодействия по протоколу PakBus небольшой участок ЛВС (локальная вычислительная сеть) регистраторов CR1000 можно настроить, как показано на рисунке *Конфигурация и кабельная система ЛВС PakBus* *(стр. 400)***.** ПК, на котором выполняется *LoggerNet*, использует порт **RS-232** первого регистратора CR1000 для обмена данными со всеми регистраторами CR1000. Все функции *LoggerNet*, такие как отправка программ, мониторинг измерений и сбор данных, доступны всем регистраторам CR1000. Регистраторы CR1000 могут быть также запрограммированы на обмен данными между собой (функция обмена данными не показана в данном примере).

8.5.6.1. Кабельная система ЛВС

Используйте трехжильный кабель для соединения регистраторов CR1000, как показано на рисунке *Конфигурация и кабельная система ЛВС PakBus* *(стр. 400)***.** Длина кабеля между любыми двумя модулями CR1000 не должна превышать 7,6 м. **COM1 Tx** (передача) и **Rx** (прием) являются клеммами **C1** и **C2** регистратора CR1000, соответственно; а **COM2 Tx** и **Rx** являются клеммами C3 и C4, соответственно. Tx от CR1000 подсоединяется к Rx близлежащего регистратора CR1000.

Рисунок 105. Конфигурация и кабельная система ЛВС PakBus



8.5.6.2. Настройка ЛВС

Конфигурацию регистраторов CR1000 необходимо настроить перед их подключением к ЛВС:

1. Запустите служебную программу конфигурации устройства *(DevConfig).* Нажмите **Device Type (Тип устройства)**: и выберите **CR1000**. Следуя командам на экране, подайте электропитание на модули CR1000 и подключите их к ПК. Закройте любые другие программы, которые могут использовать COM-порт ПК, такие как *LoggerNet, PC400, PC200W, HotSync* и др.
2. Нажмите кнопку **Connect (Подключить)** в левом нижнем углу.
3. Настройте параметры с помощью *DevConfig*, как описано в таблице *Пример ЛВС PakBus — настройки связи регистратора данных* *(стр. 402)*. Не меняйте значения по умолчанию для неуказанных параметров. Примеры со снимками диалоговых окон DevConfig показаны на рисунках, начиная с *DevConfig — Deployment (Развертывание)* | *Вкладка Datalogger (Регистратор данных)* *(стр. 401)* и заканчивая рисунком *DevConfig — Deployment (Развертывание) I Вкладка Advanced (Расширенные настройки)* *(стр. 402)***.** Если используются не новые регистраторы CR1000, рекомендуется обновить операционную систему выполнить сброс на заводские настройки по умолчанию перед работой с данным примером.

Рисунок 106. DevConfig — вкладка Deployment (Развертывание)

Рисунок 107. DevConfig — Deployment (Развертывание) | Вкладка ComPorts Settings (Настройки COM-портов)

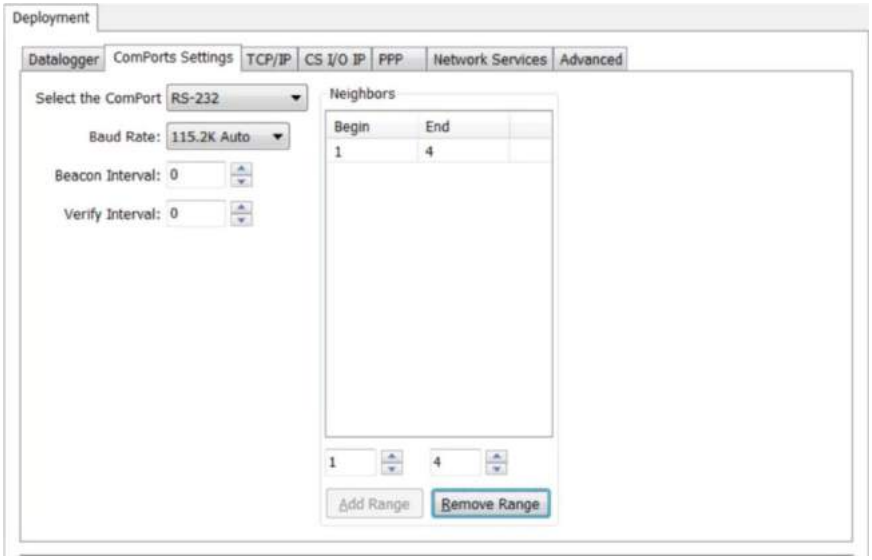


Рисунок 108. DevConfig — **Deployment (Развертывание) I Вкладка *Advanced*** *(Расширенные настройки)*

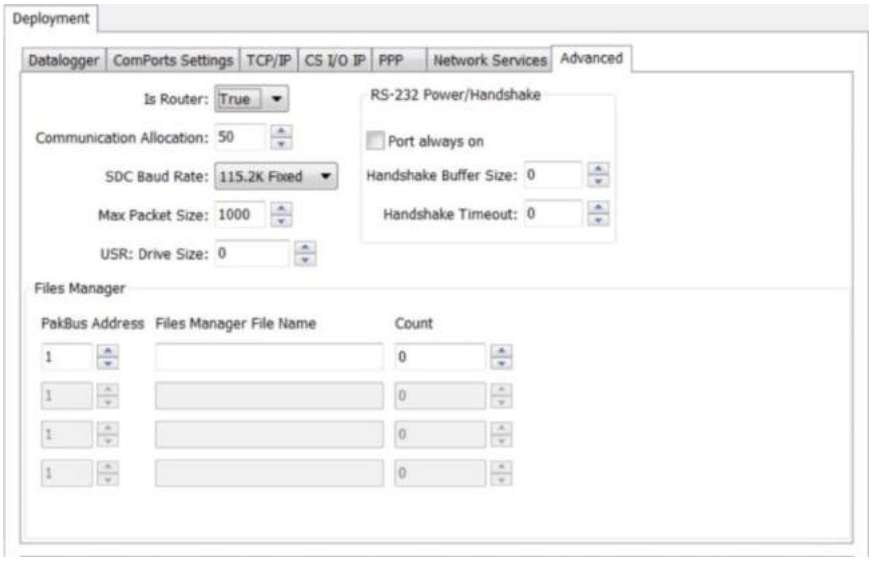


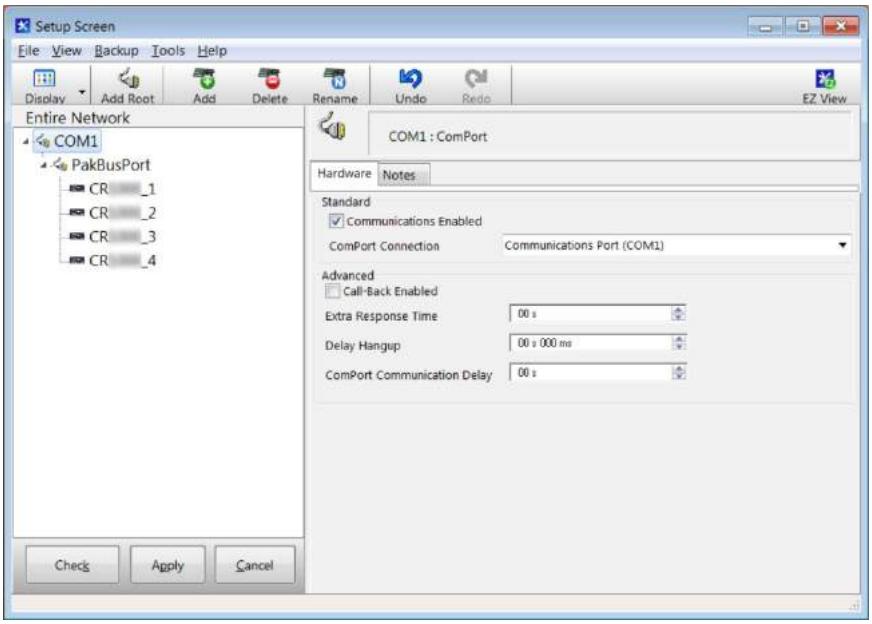
Таблица 103. Пример ЛВС PakBus — настройки связи регистратора данных

ПО →	*Служебная программа конфигурации устройства (DevConfig)*								
Вкладка →	**Deployment (Развертывание)**								
Подвкладка →	**Регистратор данных**	**ComPort Settings (Настройки COM-портов)**							Advanced (Расширенные настройки)
Параметр →	**PakBus Adr**	**COM1**			**COM2**			Is Router	
Подчиненный параметр →		**Baud Rate (Скорость передачи в бодах)**	Соседи[1]		**Baud Rate (Скорость передачи в бодах)**	Соседи[1]		
Регистратор данных ↓			Begin:	End:		Begin:	End:	
CR1000_1	**1**	**115.2K Fixed (115 200 фикс.)**	**2**	**2**	**115.2K Fixed (115 200 фикс.)**	**3**	**4**	Да
CR1000_2	**2**	**115.2K Fixed (115 200 фикс.)**	**1**	**1**	**Disabled (Отключено)**			Нет
CR1000_3	**3**	**115.2K Fixed (115 200 фикс.)**	**1**	**1**	**115.2K Fixed (115 200 фикс.)**	**4**	**4**	Да
CR1000_4	**4**	**115.2K Fixed (115 200 фикс.)**	**3**	**3**	**Disabled (Отключено)**			Нет

[1]Процесс настройки можно упростить путем присвоения всем спискам соседних узлов значений Begin: 1 End: 4.

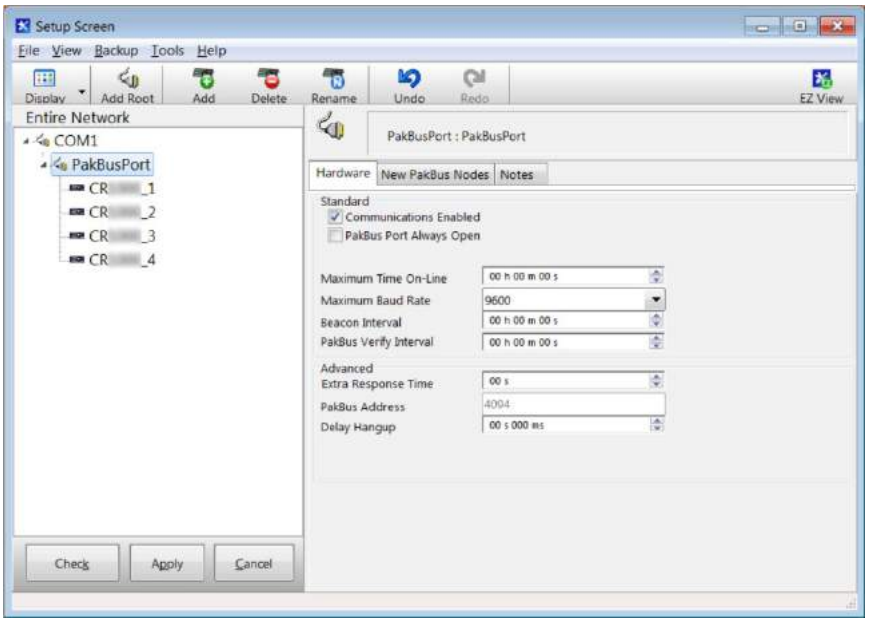
8.5.6.3. Настройка LoggerNet

*Рисунок 109. Настройка карты сети LoggerNet: **COM**-порт*



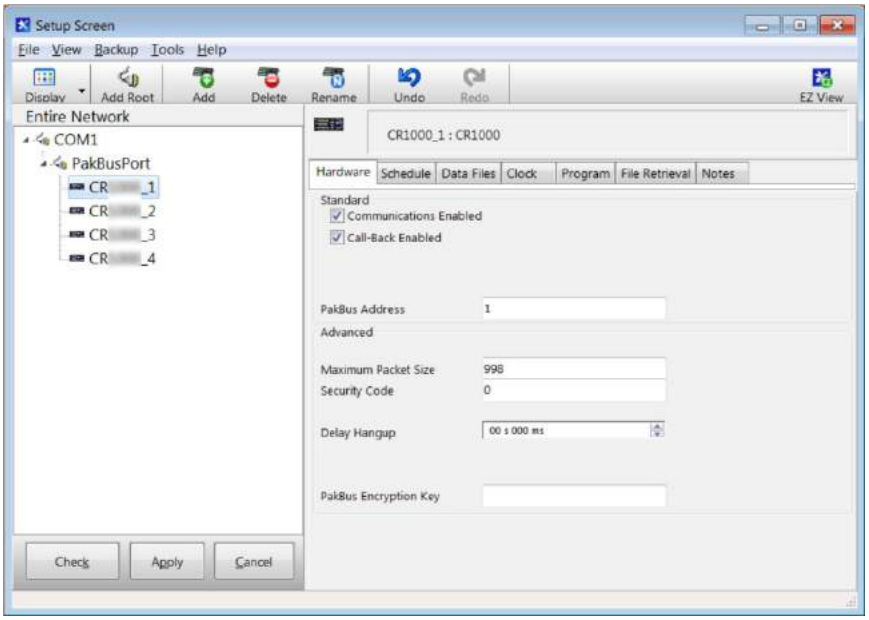
В окне *LoggerNet Setup* нажмите *Add Root (Добавить корневой элемент)* и добавьте **ComPort**. Затем нажмите **Add (Добавить)** и выберите **PakBusPort** и добавьте (4) регистраторы данных **CR1000** в карту устройств, как показано на рисунке *Настройка карты устройств LoggerNet* *(стр. 403)*.

Рисунок 110. Настройка карты сети LoggerNet: ***PakBusPort***



Как показано на рисунке *LoggerNet Device Map Setup: PakBusPort* *(стр. 404)*, задайте максимальную скорость передачи в бодах для порта PakBus, равную **115200**. Оставьте остальные параметры без изменений.

Рисунок 111. Настройка карты сети LoggerNet: регистраторы данных



Как показано на рисунке *Настройка карты устройств LoggerNet: регистраторы данных* *(стр. 404)*, настройте адрес PakBus® для каждого CR1000, как указано в таблице *Пример ЛВС PakBus — настройки связи регистратора данных* *(стр. 402)*.

8.5.7. Фильтры маршрутов

Настройка фильтров маршрутов ограничивает маршрутизацию или обработку некоторых типов сообщений PakBus таким образом, что сообщения «изменения состояния» могут обрабатываться или направляться только данным регистратором CR1000, если адрес источника данного сообщения находится в одном из диапазонов источников, а адрес получателя данного сообщения находится в соответствующем диапазоне получателей. Если ни один из диапазонов не задан (по умолчанию), регистратор CR1000 не применяет никаких ограничений к маршрутизации. Типы сообщений «изменения состояния» включают следующие: настройка переменной, сброс таблицы, управление файлами — отправка файла, настройка параметров и восстановление параметров.

Например, если данному параметру было присвоено значение (**4094, 4094, 1, 10**), регистратор CR1000 обработает или направит только те сообщения «изменения состояния», которые были отправлены с адреса 4094 и предназначены для адресов в диапазоне от одного до десяти.

Это отображается и оформляется с помощью следующих формальных синтаксических правил:

```
route-filters := { "(" source-begin source-end ","
dest-begin "," dest-end ")" }.
source-begin := uint2. ; 1 < source-begin <= 4094
source-end := uint2. ; source-begin <= source-end <= 4094
dest-begin := uint2. ; 1 < dest-begin <= 4094
dest-end := uint2. ; dest-begin <= dest-end <= 4094
```

8.5.8. PakBusRoutes

Команда PakBusRoutes() определяет маршруты (в случае маршрутизатора) или соседей маршрутизатора (в случае оконечного узла), которые известны регистратору CR1000 на момент считывания параметра. Каждый маршрут представлен четырьмя элементами, разделенными запятыми и заключенными в круглые скобки:

```
PakBusRoutes(port, via neighbor adr, pakbus adr, response time)
```

Описание параметров команды **PakBusRoutes()**:

port (порт)

Задает числовой код для порта, который будет использовать маршрутизатор:

Таблица 104. Номера портов маршрутизатора

Описание порта	*Числовой код*
ComRS232	1
ComME	2
ComSDC6 (Com310)	3
ComSDC7	4
ComSDC8	5
ComSDC9 (Com320)	6
ComSDC10	7
ComSDC11	8
Com1 (C1,C2)	9
Com2 (C3,C4)	10
Com3 (C5,C6)	11

Com4 (C7,C8)	12
IP[1]	101,102,...

[1] Если значение номера порта ≥ 101, соединение устанавливается по протоколу PakBus/TCP либо посредством выполнения регистратором CR1000 команды **TCPOpen()** либо путем создания соединения со службой PakBus/TCP регистратора CR1000.

via neighbor adr (через адрес соседа)

Задает адрес соседа/маршрутизатора, который будет использоваться для отправки сообщений по данному маршруту. Если маршрут ведет к соседу, данное значение идентично адресу.

pakbus adr (адрес PakBus)

В случае маршрутизатора задает адрес устройства, к которому ведет маршрут. В случае оконечного узла данный параметр имеет значение ***0***.

response time (время отклика)

В случае маршрутизатора задает время в миллисекундах, допустимое для данного маршрута. В случае оконечного узла данный параметр имеет значение ***0***.

8.5.9. Соседи

Имя в редакторе настроек: Neighbors Allowed xxx (Разрешенные соседние узлы xxx)

Массив целых чисел, указывающий соседние узлы PakBus для коммуникационных портов:

RS-232, ME, SDC7, SDC8, SDC10, SDC11 Com1 (C1,C2)
Com2 (C3,C4)
Com3 (C5,C6)
Com4 (C7,C8)

Данный параметр задает для данного порта явно определенный список адресов PakBus узлов, которые регистратор CR1000 будет принимать в качестве соседей. Если список пуст (состояние по умолчанию), любой узел принимается как сосед. Данный параметр не влияет на определение в качестве соседа, если адрес такого соседа больше чем 3999. Формальные синтаксические правила для данного параметра:

```
neighbor := { "(" range-begin range-end ")" }.
range-begin := pakbus-address. ;
range-end := pakbus-address.
pakbus-address := number. ; 0 < number < 4000
```

Если в списке допустимых узлов содержится более 10 соседей и интервал между сигналами маяка равен **0**, то интервалу между сигналами маяка присваивается значение **60** секунд, а отправка сигналов маяка используется для обнаружения соседей вместо отправки hello-запросов, занимающих коммуникационную память.

8.5.10. Шифрование PakBus

Два устройства PakBus могут обмениваться зашифрованными командами и данными. Шифрование осуществляется с применением алгоритма AES-128. Маршрутизаторы и другие оконечные узлы необязательно устанавливать на шифрование. Регистратор CR1000 предусматривает параметр, доступ к которому обеспечивается посредством *DevConfig* *(стр. 111)*, благодаря которому он может отправлять и получать только зашифрованные команды и данные. Аналогичным образом *LoggerNet* *(стр. 655)* имеет параметр, предусматриваемый для определенной станции, благодаря которому он может отправлять и получать только зашифрованные команды и данные.

Информация на уровне заголовков, необходимая для маршрутизации, не шифруется. Регистратор CR1000 с активированным параметром шифрования также может обмениваться информацией с регистратором данных с отключенным параметром шифрования. Используйте команду **EncryptExempt()** в программе CRBasic, чтобы определить один или несколько адресов PakBus, на которые зашифрованные сообщения не будут отправляться.

Изделия Campbell Scientific, которые поддерживают шифрование PakBus:

- LoggerNet 4.2;
- регистратор данных CR1000 (с версией ОС 26 и выше);
- регистратор данных CR3000 (с версией ОС 26 и выше);
- регистраторы данных серии CR800 (с версией ОС 26 и выше);
- регистраторы данных серии CR1000 (с версией ОС 1 и выше).

Служебная программа конфигурации устройства (DevConfig) версии 2.04 и выше

- *Network Planner* версии 1.6 и выше.

Области протокола, к которым применяется шифрование PakBus:

- все сообщения BMP5;
- все сообщения, связанные с параметрами.

Примечание. Базовые сообщения PakBus, такие как **Hello, Hello Request, Send Neighbors, Get Neighbors** и **Echo** НЕ шифруются.

Ключ шифрования PakBus можно установить в регистраторе данных CR1000 следующими способами:

- DevConfig — вкладка **Deployment** (Развертывание);
- *DevConfig — вкладка* **Settings Editor** (Редактор настроек);
- диалоговое окно редактора настроек *PakBusGraph*;
- экранная клавиатура.

Храните ключ шифрования в безопасном месте. Если ключ шифрования утерян, его необходимо сбросить. Сброс ключа на экранной клавиатуре осуществляется путем удаления знаков маркера, которые отображаются в поле, с последующим вводом нового ключа.

Примечание. Ключ шифрования невозможно задать с помощью программы CRBasic.

Настройка ключа шифрования с помощью *ПО поддержки регистратора данных* *(стр. 512)* *(LoggerNet версии 4.2* и выше):

- применяется к регистраторам данных, маршрутизаторам PakBus и типам устройств с портами PakBus CR1000, CR3000, серий CR800 и CR1000.
- может устанавливаться в *LoggerNet* **(на экране Set Up)**, *Network Planner* или *CoraScript* (настроить порт PakBus можно только в *CoraScript*).

Примечание. Задание ключа шифрования для устройства с портом PakBus приведет к тому, что все отправляемые сообщения будут шифроваться.

8.6. Альтернативная связь — подробная информация

Связанные темы:
• *Альтернативная связь — общие сведения (стр. 90)*
• *Альтернативная связь — подробная информация (стр. 407)*

CR1000 обменивается данными с *ПО поддержки регистратора данных (стр. 95)* и другими *регистраторами данных Campbell Scientific (стр. 645)* посредством протокола *PakBus (стр. 522)*. Также предусматривается поддержка протоколов Modbus, DNP3, TCP/IP и нескольких других протоколов промышленной связи. Поддержка связи CAN обеспечивается при использовании коммуникационного модуля *SDM-CAN (стр. 651)* производства Campbell Scientific.

8.6.1. DNP3 — подробная информация

Связанные темы:
• *DNP3 — общие сведения (стр. 91)*
• *DNP3 — подробная информация (стр. 408)*

В начале 2015 года предполагается значительное обновление данного раздела.

8.6.1.1. DNP3 — введение

Регистратор CR1000 обладает совместимостью DNP3 SCADA. DNP3 — это протокол SCADA, который, в основном, используется в отраслях инженерных коммуникаций, сетях выработки и распределения электроэнергии и водоочистной отрасли.

Протокол распределенной сети (Distributed Network Protocol, DNP) — открытый протокол, используемый в приложениях для обеспечения целостности с использованием минимальной полосы пропускания. DNP, реализованный в регистраторе CR1000, — DNP3 уровня 2 с совместимостью с ведомыми устройствами и с поддержкой некоторых операций, присущих уровню 3. Стандартная программа CR1000 с командами DNP будет принимать массивы данных в PMB или обработанных данных, а затем сопоставлять их с массивами DNP в целочисленном или двоичном формате. Регистратор CR1000 отправляет запрашиваемые данные в ответ на запрос любого ведущего устройства DNP или отправляет незапрашиваемые ответы заданному ведущему устройству DNP. Поддержка связи по протоколу DNP реализована в CR1000 посредством порта RS-232, **COM1, COM2, COM3** или **COM4** или через TCP, используя преимущество нескольких вариантов связи, совместимых с регистратором CR1000, таких как РЧ, мобильная или спутниковая связь. Состояние и история DNP3 сохраняются при выключении-включении электропитания и других сбросах в энергонезависимой памяти.

Программное обеспечение DNP SCADA обеспечивает возможность перемещения данных регистратора CR1000 непосредственно в базу данных или на экран дисплея. Области применения включают следующие: мониторинг погоды вблизи линий электропередачи для расширения круга оперативных решений, мониторинг и контроль орошения из установки очистки сточных вод, контроль удаленных насосов, измерение речного потока, а также мониторинг движения и качества воздуха на электростанции.

8.6.1.2. Программирование для DNP3

В примере CRBasic *Реализация DNP3 (стр. 410)* рассматривается случай, когда в коде CRBasic используются аналоговые данные ***Iarray()*** и двоичные данные ***Barray()*** (состояние контрольного порта 5) и сопоставляются с массивами DNP. Регистратор CR1000 отправляет запрашиваемые данные в ответ на запрос любого ведущего устройства DNP или отправляет незапрашиваемые ответы заданному ведущему устройству DNP.

8.6.1.2.1. Объявления (программирование DNP3)

В таблице *Реализация DNP3 — типы данных, подлежащих хранению в таблицах общего пользования для групп объектов (стр. 409)* представлены группы объектов, поддерживаемые реализацией DNP в регистраторе CR1000, и необходимые типы данных. Полный список групп и различных вариантов содержится в *Справке по редактору CRBasic* для команды **DNPVariable().**

Таблица 105. Реализация DNP3 — типы данных, подлежащих хранению в таблицах общего пользования для групп объектов		
Тип данных	***Группа***	***Описание***
Логический	1	Дискретный ввод
	2	Изменение дискретного ввода
	10	Дискретный вывод
	12	Блок управления
Длинный	30	Аналоговый ввод
	32	Событие аналогового изменения
	40	Состояние аналогового вывода
	41	Блок аналоговых выводов
	50	Время и дата
	51	Время и дата — СТО

8.6.1.2.2. Команды CRBasic (DNP3)

Полное описание и варианты команд приводятся в *Справке по редактору CRBasic*.

DNP()

Устанавливает регистратор CR1000 в качестве ведомого устройства DNP (удаленная станция/сервер) с назначением адреса и DNP3-выделенного COM-порта. Обычно находится между **BeginProg** и **Scan(),** поэтому выполняется однократно. Пример приводится в примере CRBasic — *Реализация DNP3 (стр. 410)*, строка 20.

Синтаксис

```
DNP(ComPort, BaudRate, DNPSlaveAddr)
```

DNPVariable()

Связывает определенный массив переменных с группой объектов DNP. Если ведущее устройство опрашивает регистратор CR1000, последний отправляет в ответ все заданные переменные с указанием их конкретных групп. Также используется для настройки данных события, которые отправляются в ведущее устройство при изменении значения переменной. Пример приводится в примере CRBasic *Реализация DNP3 (стр. 410)*, строка 24.

Синтаксис

```
DNPVariable(Source, Swath, DNPObject, DNPVariation, DNPClass,
   DNPFlag, DNPEvent, DNPNumEvents)
```

DNPUpdate()

Определяет момент, когда ведомое устройство DNP (удаленная станция/сервер) будет обновлять свои массивы элементов DNP. Задает адрес ведущего устройства DNP, которому отправляются незапрашиваемые ответы (данные события). Должен включаться однократно в процедуру **Scan() / NextScan** для ведомого устройства DNP для обновления своих массивов. Обычно размещается в программе следом за обновляемыми элементами массива. Регистратор CR1000 отвечает любому ведущему устройству DNP независимо от его адреса.

Синтаксис

```
DNPUpdate (DNPSlaveAddr,DNPMasterAddr)
```

8.6.1.2.3. Программирование для DNP3 — сбор данных

Как показано в примере CRBasic *Реализация DNP3* *(стр. 410)*, запрограммируйте регистратор CR1000 на отправку данных в ответ на запрос ведущего устройства DNP3, выполнив следующие три действия:

1. Введите команду **DNP()** в начале программы между параметрами **BeginProg** и **Scan().** Задайте COM-порт, скорость передачи данных в бодах и DNP3-адрес.
2. Настройте переменные, которые должны отправляться ведущему устройству с помощью команды **DNPVariable().** Сдвоенные команды охватывают статические данные (текущие значения) и данные события (предыдущие десять записей).
   - Для аналоговых измерений:

     ```
     DNPVariable(Variable_Name,Swath,30,2,0,&B00000000,0,0)
     DNPVariable(Variable_Name,Swath,32,2,3,&B00000000,0,10)
     ```

   - Для цифровых измерений (порты управления):

     ```
     DNPVariable(Variable_Name,Swath,1,2,0,&B00000000,0,0)
     DNPVariable(Variable_Name,Swath,32,2,3,&B00000000,0,10)
     ```

3. Введите команду **DNPUpdate()** следом за **Scan()** внутри основного сканирования. Ведущее устройство DNP3 уведомляется о любых изменениях данных при каждом выполнении команды **DNPUpdate()**. Например, в случае сканирования с интервалом в 10 секунд ведущее устройство уведомляется каждые 10 секунд.

CRBasic: пример 66. Реализация DNP3

```
В данном примере программы демонстрируется базовая реализация DNP3 в регистраторе CR1000.
Регистратор CR1000запрограммирован на отправку данных по протоколу IP в ответ на запрос ведущего
устройства DNP3. Ключевыми элементами программы являются следующие:
@@@of the program are as follows:

    1. Команда DNP() вводится в начале программы между параметрами BeginProg
       и Scan(). COM-порт, скорость передачи данных в бодах и DNP3-адрес заданы.

    2. Переменные настраиваются для отправки ведущему устройству с помощью DNPVariable().
       Сдвоенные команды охватывают статические данные (текущие значения) и данные события
       (предыдущие десять записей).
       @@@cover static data (current values) and event data (previous ten records). Далее
       приводятся наборы сдвоенных команд для аналоговых и цифровых измерений:
       @@@are the sets of dual instructions for analog and digital measurements:

            Для аналоговых измерений:
              DNPVariable(Variable_Name,Swath,30,2,0,&B00000000, 0,0)
              DNPVariable(Variable_Name,Swath,32,2,3,&B00000000, 0,10)

            Для цифровых измерений (портов управления):
            DNPVariable(Variable_Name,Swath,1,2,0,&B00000000,0,0)
            DNPVariable(Variable_Name,Swath,32,2,3,&B00000000,0,10)

    3. Команда DNPUpdate() вводится следом за Scan(), внутри основного сканирования. Ведущее
       устройство DNP3 уведомляется о любых изменениях данных при каждом выполнении команды
       DNPUpdate().Например, в случае сканирования с интервалом в 10 секунд ведущее устройство
       уведомляется каждые 10 секунд.
       @@@notified of any change in data each time DNPUpdate() runs. @@@For example, for a 10
       @@@second scan, the master is notified every 10 seconds.
```

```
Public IArray(4) As Long
Public BArray(2) As Boolean

Public WindSpd
Public WindDir
Public Batt_Volt
Public PTemp_C

Units WindSpd=meter/Sec
Units WindDir=Degrees
Units Batt_Volt=Volts
Units PTemp_C=Deg C

Основная программа
BeginProg

   Связь DNP через IP со скоростью 115,2 кбит/с. DNP3-адрес регистратора CR1000 равен 1.
   DNP(20000,115200,1)

   DNPVariable(Source,Swath,DNPObject,DNPVariation,DNPClass,DNPFlag,DNPEvent,DNPNumEvents)
   DNPVariable(IArray,4,30,2,0,&B00000000,0,0)

   Группа объектов 30, вариант 2 используется для отправки аналоговых данных в ответ на опрос
   регистратора CR1000.
   @@@is polled. Признак присваивается пустому 8-битному номеру (все нули), DNPEvent является
   зарезервированным параметром и на текущий момент всегда установлен на нуль. Номер события
   используется только для данных события.
   @@@only used for event data.
   DNPVariable(IArray,4,32,2,3,&B00000000,0,10)
   DNPVariable(BArray,2,1,1,0,&B00000000,0,0)
   DNPVariable(BArray,2,2,1,1,&B00000000,0,1)

   Scan(1,Sec,1,0)
     Измерения датчика направления и скорости ветра WS_ms and WindDir:
     PulseCount(WindSpd,1,1,1,3000,2,0)
     IArray(1) = WindSpd * 100
     BrHalf(WindDir,1,mV2500,1,Vx1,1,2500,True,0,_60Hz,355,0)
     If WindDir>=360 Then WindDir=0
     IArray(2) = WindDir * 100

     Измерения напряжения аккумулятора регистратора данных по умолчанию Batt_Volt:
     Battery(Batt_Volt)
     IArray(3) = Batt_Volt * 100

     Измерение температуры коммутационной панели PTemp_C:
     PanelTemp(PTemp_C,_60Hz)
     IArray(1) =PTemp_C
     PortGet(Barray(1),5)

     Обновить массивы DNP и отправить незапрашиваемые ответы ведущему устройству DNP с адресом 3
     DNPUpdate(2, 3)
   NextScan
EndProg
```

8.6.2. Modbus — подробная информация

Связанные темы:
- *Modbus — общие сведения* *(стр. 91)*
- *Modbus — подробная информация* *(стр. 411)*

CR1000 поддерживает протокол Modbus, в качества ведущего и ведомого устройства в сетях Modbus SCADA. Modbus — это широко распространенный протокол передачи данных SCADA, который облегчает обмен информацией и данными между компьютером/программным обеспечением HMI, устройствами (RTU) и датчиками, совместимыми с протоколом Modbus. CR1000 обменивается данными с Modbus через порты RS-232, RS-485 (с адаптером RS-232 в RS-485) и TCP.

Системы, основанные на протоколе Modbus, состоят из ведущего устройства (компьютера) и ведомого устройства (RTU/PLC), портативных устройств (датчиков) и оборудования сети передачи данных. Коммуникационный порт, скорость передачи данных, биты данных, стоп-биты и четность устанавливаются драйвером Modbus ведущего и/или ведомых устройства. Стандарт Modbus использует два режима связи: RTU и ASCII. Но CR1000s использует только режиме RTU.

CR1000 позволяет опрашивать портативные устройства. Поскольку Modbus имеет структуру набора команд, запрограммировать CR1000 на получение данных от датчиков для получения данных от портативных устройств гораздо проще, чем для получения данных от серийных датчиков. Поскольку Modbus использует общую шину и обращается отдельно к каждому узлу, портативные устройства эффективно передают данные в мультиплексном режиме в CR1000 без дополнительного оборудования.

Регистратор CR1000 переходит в спящий режим через 40 секунд коммуникационного бездействия. После перехода в спящий режим, требуется отправить два пакета, чтобы регистратор CR1000 ответил. Первый пакет выводит CR1000 из спящего режима; второй пакет принимается как данные. Регистраторы CR1000 можно настроить в *DevConfig* или таблице **Status (Состояние)** (см. приложение *Таблица состояния и настройки* *(стр. 603)*) на поддержание коммуникационных портов в открытом и активном состоянии, но с повышенным потреблением электроэнергии.

8.6.2.1. Терминология Modbus

В таблице *Эквивалентные термины в протоколах Modbus и Campbell Scientific* *(стр. 412)* приводится список терминологических эквивалентов, с помощью которых можно встраивать регистраторы CR1000 в систему SCADA.

Таблица 106. Эквивалентные термины в протоколах Modbus и Campbell Scientific

Домен Modbus	*Вид данных*	*Домен Campbell Scientific*
Катушки	Однобитовый	Порты, флаги, булевы логические переменные
Цифровые регистры	16-битное слово	Переменные с плавающей запятой
Входные регистры	16-битное слово	Переменные с плавающей запятой
Регистры временного хранения	16-битное слово	Переменные с плавающей запятой
RTU/ПЛК		CR1000
Ведущее устройство		Обычно ПК
Ведомое устройство		Обычно CR1000
Полевой КИП		Датчик

8.6.2.1.1. Глоссарий терминов Modbus

Термин: катушки (00001–09999)

Изначально «катушками» назывались катушки реле. В регистраторах CR1000 катушками являются исключительно клеммы, настроенные на управление, программные флаги или массивы булевых логических переменных. Клеммы, настроенные на управление, выводятся логически, если параметр 5 команды **ModbusSlave()** имеет значение 0. Катушки

присваиваются регистрам Modbus от **00001** до **09999.**

Термин: цифровые регистры 10001–19999

Параметры удержания, полученные в результате цифрового измерения. Цифровые регистры в домене Modbus имеют атрибут «только для чтения». В домене Campbell Scientific первая цифра в регистрах Modbus игнорируется, и поэтому они присваиваются одному массиву переменных **Dim** или **Public** (чтение/запись).

Термин: входные регистры 30001–39999

Параметры удержания, полученные в результате аналогового измерения. Входные регистры в домене Modbus имеют атрибут «только для чтения». В домене Campbell Scientific первая цифра в регистрах Modbus игнорируется, и поэтому они присваиваются одному массиву переменных **Dim** или **Public** (чтение/запись).

Термин: регистры временного хранения 40001–49999

Параметры удержания, полученные в результате операции программирования. Регистры временного хранения в домене Modbus имеют атрибут «только для чтения». В домене Campbell Scientific первая цифра в регистрах Modbus игнорируется, и поэтому они присваиваются одному массиву переменных **Dim** или **Public** (чтение/запись).

Термин: RTU/ПЛК

Дистанционные телеметрические блоки (RTU) и программируемые логические контроллеры (ПЛК) одно время использовались в индивидуальных сферах применения. Однако по мере развития технологии различие между блоками RTU и ПЛК стало не таким явным. Регистратор CR1000 подходит под определение как RTU, так и ПЛК.

8.6.2.2. Программирование для Modbus

8.6.2.2.1. Объявления (программирование для Modbus)

В таблице *Порты, флаги и переменные CRBasic и регистры Modbus* *(стр. 413)* представлена связь между клеммами, настроенными на управление, флаги или булевы логические переменные и регистрами Modbus. Протокол Modbus не различает клеммы, настроенные на управление, флаги или булевы логические переменные. Если объявляется, что клемма настроена только на управление, флаги или булевы логические переменные, то адресация к объявленному признаку осуществляется по умолчанию. Типовая программа CRBasic для приложения Modbus объявит переменные и порты, или переменные и флаги, или переменные и булевы логические переменные.

Таблица 107. Порты, флаги и переменные CRBasic и регистры Modbus

Функция CR1000	*Пример объявления CRBasic*	*Эквивалентный пример для регистра Modbus*
Клемма, настроенная на управление	`Public Port(8)`	00001–00009
Флаг	`Public Flag(17)`	00001–00018
Булева логическая переменная	`Public ArrayB(56) as Boolean`	00001–00057
Переменная	`Public ArrayV(20)`[1]	40001–40041[1] **или** 30001–30041[1]

[1] Ввиду различий между байтами и цифрами, каждой переменной из домена CR1000 соответствует два входных регистра / регистра временного хранения из домена Modbus.

8.6.2.2.2. Команды CRBasic (Modbus)

Полное описание и варианты команд приводятся в *Справке по редактору CRBasic*.

ModbusMaster()

Настраивает регистратор данных CR1000 как ведущее устройство Modbus для отправки или извлечения данных из ведомого устройства Modbus.

Синтаксис

```
ModbusMaster(ResultCode, ComPort, BaudRate, ModbusAddr,
  Function, Variable, Start, Length, Tries, TimeOut)
```

ModbusSlave()

Настраивает CR1000 как ведомое устройство ModBus.

Синтаксис

```
ModbusSlave(ComPort, BaudRate, ModbusAddr, DataVariable,
  BooleanVariable)
```

MoveBytes()

Перемещает двоичные байты данных в другое место памяти при переводе данных из формата «от старшего к младшему» в «от младшего к старшему». См. приложение *Порядок следования байтов* *(стр. 643)*.

Синтаксис

```
MoveBytes(Dest, DestOffset, Source, SourceOffset, NumBytes)
```

8.6.2.2.3. Адресация (ModbusAddr)

Устройства Modbus обладают уникальным адресом во всех сетях. Диапазон заключается в пределах от **1** до **247**. Адрес **0** зарезервирован для универсальной рассылки. Если используется NL240, необходимо применять тот же номер, что и в адресах Modbus и PakBus.

8.6.2.2.4. Поддерживаемые функциональные коды Modbus

В протоколе Modbus предусматривается множество функциональных кодов. Команды регистратора CR1000 поддерживают следующие из них.

Таблица 108. Поддерживаемые функциональные коды Modbus

Код	Название	Описание
01	Read coil/port status (Считывание состояния катушки/порта)	Считывает состояние вкл/выкл дискретного выхода (выходов) в ведомом устройстве ModBus
02	Read input status (Считывание состояния входа)	Считывает состояние вкл/выкл дискретного входа (выходов) в ведомом устройстве ModBus
03	Read holding registers (Считывание регистров временного хранения)	Считывает двоичное содержимое регистра (регистров) временного хранения в ведомом устройстве ModBus
04	Read input registers (Считывание входных регистров)	Считывает двоичное содержимое входного регистра (регистров) в ведомом устройстве ModBus
05	Force single coil/port (Принудительное переключение одиночной катушки/порта)	Принудительно включает/выключает одиночную катушку/порт в ведомом устройстве ModBus
06	Write single register (Запись одиночного регистра)	Записывает значение в регистр временного хранения в ведомом устройстве ModBus
15	Force multiple coils/ports (Принудительное переключение нескольких катушек/портов)	Принудительно включает/выключает нескольких катушек/портов в ведомом устройстве ModBus
16	Write multiple registers (Запись нескольких регистров)	Записывает значения в ряде регистров временного хранения в ведомом устройстве ModBus

8.6.2.2.5. Считывание регистров Modbus в обратном формате

Некоторым устройствам Modbus требуется обратный порядок следования байтов в словах (CDAB → ABCD). Данное условие применимо к формату с плавающей запятой или целочисленному формату данных. Поскольку ведомое устройство CR1000 использует формат ABCD, какому-либо ведущему устройству требуется ввести поправку, что возможно в некоторых случаях, или регистратор CR1000 должен выдавать слова с обратным порядком следования байтов. Чтобы изменить в CR1000 порядок следования байтов на обратный, воспользуйтесь командой **MoveBytes()**, как показано в нижеследующем примере кода.

```
for i = 1 to k
  MoveBytes(InverseFloat(i) , 2 , Float(i) ,0,2)
  MoveBytes(InverseFloat(i) ,0, Float(i) ,2,2)
next
```

В примере выше ***InverseFloat(i)*** является массивом со словом с обратным порядком следования байтов (CDAB). Массив ***Float(i)*** имеет слово с прямым порядком следования байтов (ABCD).

См. приложение *Порядок следования байтов (стр. 643)*.

8.6.2.3. Диагностика (Modbus)

Диагностика функций Modbus на регистраторе CR1000 осуществляется при помощи программного обеспечения Modbus сторонних разработчиков. Более подробную информацию можно найти по следующим ссылкам:

- www.simplyModbus.ca/FAQ.htm
- www.Modbus.org/tech.php
- www.lammertbies.nl/comm/info/modbus.html

8.6.2.4. Modbus over IP

Функциональность Modbus over IP является опциональной в регистраторе CR1000. Просим обратиться в Campbell Scientific для получения более подробной информации.

8.6.2.5. Modbus — вопрос-ответ

В: Может ли протокол Modbus использоваться для соединения со следующими характеристиками: RS-232, 7 битов данных, контроль по четности, один стоповый бит?

А: Да. Вставьте перед **ModBusMaster() / ModBusSlave()** команду **SerialOpen()** и задайте числовому формату COM-порта любой доступный формат, включая опцию 7 битов данных и контроль по четности. Команды **SerialOpen()** и **ModBusMaster()** можно использовать один раз и вставить перед **Scan().**

Связывание двух длинных 16-битных переменных Modbus в одно длинное 32-битное число Modbus.

8.6.2.6. Преобразование 16-битных длинных чисел Modbus в 32-битные

Связывание двух длинных 16-битных переменных Modbus в одно длинное 32-битное число Modbus показано в следующем примере.

CRBasic: пример 67. Связывание длинных переменных Modbus

```
'Данный пример программы демонстрирует связывание (сращивание) переменных с типом данных «длинное целое» для операций Modbus. Программа совместима со следующими или более современными версиями операционной системы:
'CR800 OS v.3
'CR1000 OS v.12
'CR3000 OS v. 5

'ПРИМЕЧАНИЕ. В CR1000 используется порядок следования слов «от старшего к младшему».

'Объявления
Public Combo As Long                          'Переменная, хранящая объединенный 32-битный
Public Register(2) As Long                    'массив, хранит две 16-битные длинные переменные
                                              'ModBus
                                              '@@@variables
                                              'Register(1) = Младшее значимое слово
                                              'Register(2) = Старшее значимое слово
Public Result                                 'Хранит результат опроса ведущего устройства ModBus
                                              '@@@query

'Дополнительные имена, использованные для уточнения
Alias Register(1) = Register_LSW              'Младшее значимое слово.
Alias Register(2) = Register_MSW              'Старшее значимое слово.

BeginProg
   'Если используются следующие числа (сперва снимите символ одинарной кавычки)
   'Combo читается как 131073 в десятичной системе счисления
   'Register_LSW=&h0001 'Младшее значимое слово.
   'Register_MSW=&h0002 'Старшее значимое слово.
```

```
  Scan(1,Sec,0,0)
    Если регистратор CR1000 является ведущим устройством ModBus, то должна использоваться команда
    ModbusMaster (вместо фиксирования переменных,
    как показано в тексте программы между командами BeginProg и SCAN).
    ModbusMaster(Result,COMRS232,-115200,5,3,Register(),-1,2,3,100)

    MoveBytes(DestVariable,DestOffset,SourceVariable,SourceOffSet,
    NumberOfBytes)
    MoveBytes(Combo,2, Register_LSW,2,2)
    MoveBytes(Combo,0, Register_MSW,2,2)
  NextScan
EndProg
```

8.6.3. TCP/IP — подробная информация

Связанные темы:
• *TCP/IP — общие сведения (стр. 91)*
• *TCP/IP — подробная информация (стр. 423)*
• *TCP/IP — команды (стр. 593)*
• *Подключения TCP/IP — список(стр. 652)*

Следующие протоколы TCP/IP поддерживаются в рамках CR1000 при использовании *сетевых подключений (стр. 652)* с постоянным стеком IP или при использовании сотового модема с активированным ключом PPP/IP. Более подробная информация о некоторых из этих протоколов приведена в следующих разделах.

- DHCP
- DNS
- FTP
- HTML
- HTTP
- микро-последовательный сервер;
- NTCIP;
- NTP;
- PakBus через TCP/IP;
- Ping;
- POP3;
- SMTP
- SNMP;
- Telnet;
- *Web API (стр. 423)*
- XML

Наиболее актуальная информация о реализации этих протоколов содержится в *Справке по редактору CRBasic*. Список команд CRBasic можно найти в приложении *TCP/IP (стр. 593)*.

Читать далее. Специальная информация касательно использования цифровых сотовых модемов для TCP/IP находится в руководствах Campbell Scientific по данным модемам. Информацию о доступных устройствах TCP/IP/PPP, в том числе номера моделей, см. в приложении *Сетевые подключения (стр. 652)*. Подробную информацию об использовании устройств TCP/IP/PPP можно найти в соответствующих руководствах (доступны по адресу *www.campbellsci.com http://www.campbellsci.com)* и в *Справке по редактору CRBasic*.

8.6.3.1. PakBus через TCP/IP и обратный вызов

После конфигурации аппаратного обеспечения возможна коммуникация PakBus® по TCP/IP. Среди этих функций:

- Отправка программ
- Вызов программ
- Установка времени CR1000
- Сбор данных
- Отображение текущей записи в таблице данных

Обратный вызов данных и связь между регистраторами данных также возможны по TCP/IP. Подробная информация и примеры программ для обратного вызова и связи между регистраторами данных доступны в руководстве по сетевым подключениям. Список номеров моделей для сетевых подключений находится в приложении *Сетевые подключения (стр. 652)*.

8.6.3.2. Веб-сервер HTTP по умолчанию

В CR1000 по умолчанию есть домашняя страница, встроенная в операционную систему. Доступ к домашней странице осуществляется по следующему адресу:

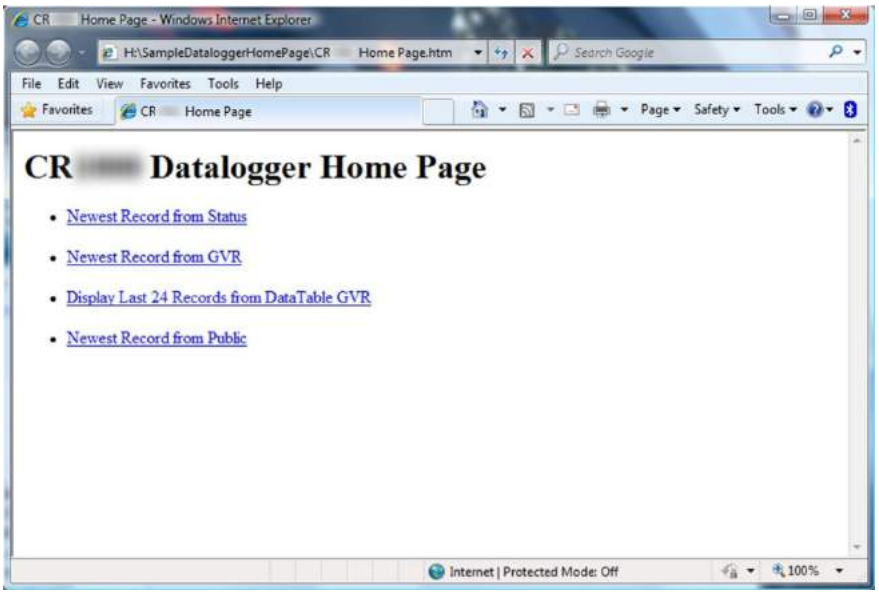
```
http:\\ipaddress:80
```

Примечание. Порт 80 подразумевается, если нет конкретного указания порта.

Как показано на рисунке *Предварительно сконфигурированная домашняя страница HTML (стр. 291)*, эта страница предоставляет ссылки на самые новые записи во всех таблицах, включая таблицу состояния (**Status**), общую таблицу (**Public**) и таблицы данных. Также доступны ссылки на последние 24 записи в каждой из таблиц данных. Если в таблице данных хранится менее 24 записей, по ссылке будут отображены все данные из таблицы.

Ссылки на новые записи автоматически обновляются каждые 10 секунд. Ссылку **Последние 24 записи** необходимо обновлять вручную, чтобы отобразить новые данные. Также автоматически создаются ссылки на любые файлы HTML, XML и JPEG, найденные на дисках CR1000. Чтобы скопировать файлы на эти диски, выберите **Управление файлами** в меню *ПО поддержки регистратора данных (стр. 512)*.

Рисунок 112. Предварительно сконфигурированная домашняя страница HTML



8.6.3.3. Пользовательский веб-сервер HTTP

Несмотря на то, что домашняя страница по умолчанию недоступна для редактирования, ее можно заменить HTML-кодом настроенной пользователем веб-страницы. Чтобы заменить домашнюю страницу по умолчанию, сохраните новую домашнюю страницу под именем *default.html* и скопируйте ее в регистратор данных. Ее можно скопировать на диск CR1000 через **Управление файлами**. При удалении *default.html* CR1000 будет использовать изначальную домашнюю страницу, заданную по умолчанию.

CR1000 можно запрограммировать на генерацию кода HTML или XML, который может просматриваться с помощью веб-браузера. В примере CRBasic *HTML* *(стр. 293)* демонстрируется использование команд CRBasic **WebPageBegin() / WebPageEnd** и **HTTPOut()** для создания кода HTML. Обратите внимание, что для HTML-кода, требующего применения кавычек, используется **CHR(34)**, в то время как обычные кавычки определяют начало и окончание буквенно-числовых строк внутри скобок в команде **HTTPOut()**. Дополнительная информация приведена в *Справке по редактору CRBasic.*

В данном примере программы домашняя страница по умолчанию заменяется с помощью **WebPageBegin** путем создания файла с именем default.html. Внешний вид новой домашней страницы по умолчанию, созданной в программе, показан на рисунке *Домашняя страница, созданная с помощью команды WebPageBegin()* *(стр. 292)*.

Логотип Campbell Scientific на веб-странице находится в файле **SHIELDWEB2.JPG**, который необходимо передать с ПК в CR1000 b сохранит на диск CPU: посредством **Управления файлами** в ПО поддержки регистратора данных.

Вторая веб-страница, показанная на рисунке *Пользовательская веб-страница числового отслеживания* *(стр. 293)* и названная «monitor.html», создана в примере программы, содержащей ссылки на таблицы данных CR1000.

*Рисунок 113. Домашняя страница, созданная с помощью команды **WebPageBegin()***

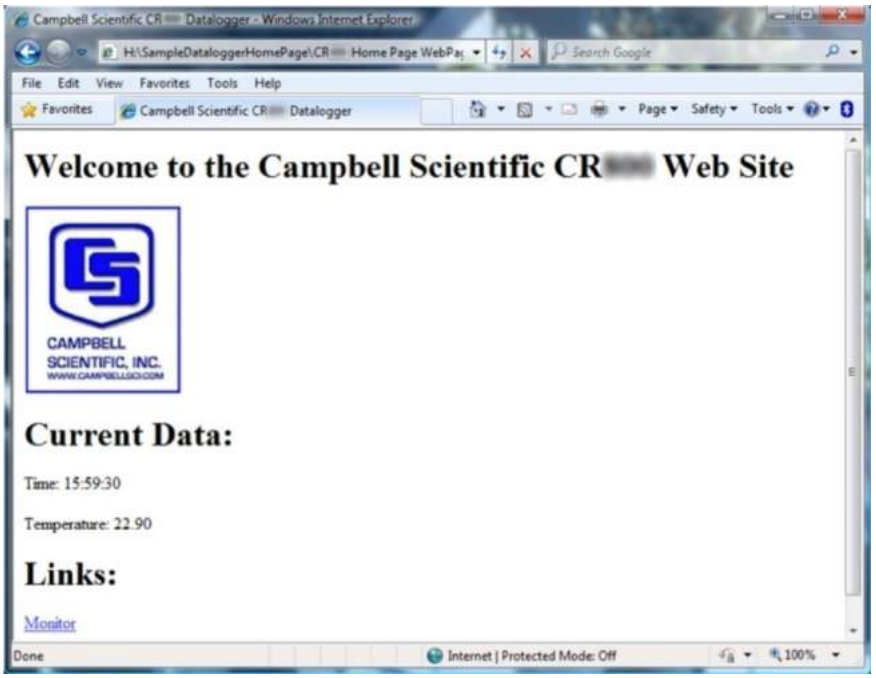
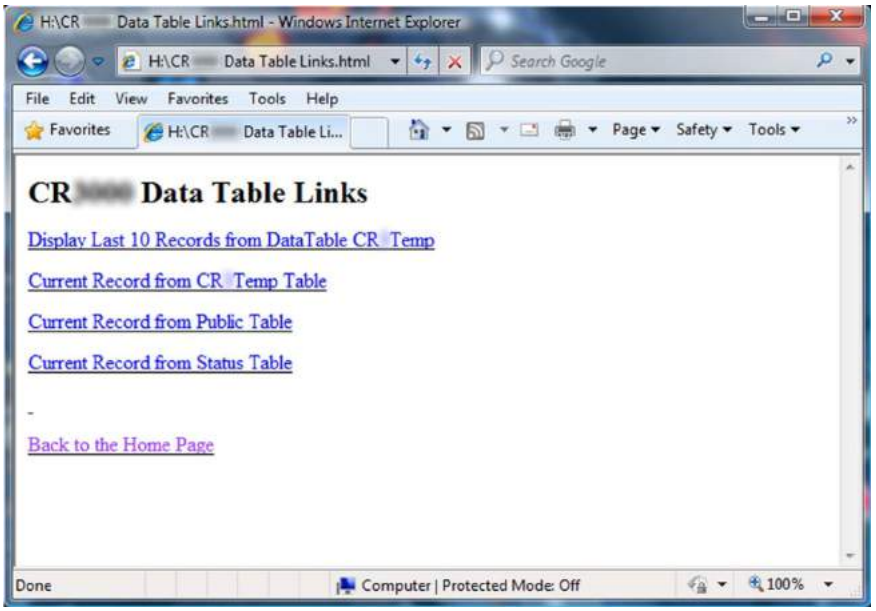


Рисунок 114. Пользовательская веб-страница числового отслеживания



CRBasic: пример 68. Пользовательская веб-страница HTML

```
Данный пример программы иллюстрирует создание пользовательской веб-страницы, находящейся в
CR1000. В этом примере программы стандартная домашняя страница заменяется с помощью WebPageBegin
путем создания файла с именем default.html. Изображение на веб-странице (в данном случае это
логотип Campbell Scientific) находится в файле SHIELDWEB2.JPG. Файл изображения должен быть
скопирован на диск CPU: устройства CR1000 с помощью функции Управления файлами в ПО поддержки
регистратора данных. Затем создается вторая веб-страница, содержащая ссылки на таблицы данных
CR1000.

ПРИМЕЧАНИЕ. Символ «_», используемый в конце некоторых строк, позволяет перенести кодовый
оператор на следующую строку.

Dim Commands As String * 200
Public Time(9), RefTemp,
Public Minutes As String, Seconds As String, Temperature As String

DataTable(CRTemp,True,-1)
   DataInterval(0,1,Min,10)
   Sample(1,RefTemp,FP2)
   Average(1,RefTemp,FP2,False)
EndTable

HTML-страница по умолчанию
WebPageBegin("default.html",Commands)
   HTTPOut("<html>")
   HTTPOut("<style>body {background-color: oldlace}</style>")
   HTTPOut("<body><title>Campbell Scientific CR1000 Datalogger</title>")
   HTTPOut("<h2>Welcome To the Campbell Scientific CR1000 Web Site!</h2>")
   HTTPOut("<tr><td style=" + CHR(34) +"width: 290px" + CHR(34) + ">")
   HTTPOut("<a href=" + CHR(34) + "http://www.campbellsci.com" + CHR(34) + ">")
   HTTPOut("<img src="+ CHR(34) +"/CPU/SHIELDWEB2.jpg"+ CHR(34) + "width=" + _
      CHR(34) +"128"+CHR(34)+"height="+CHR(34)+"155"+ CHR(34) + "class=" + _
      CHR(34) +"style1"+ CHR(34) +"/></a></td>")
   HTTPOut("<p><h2> Current Data:</h2></p>")
   HTTPOut("<p>Time: " + time(4) + ":" + minutes + ":" + seconds + "</p>")
   HTTPOut("<p>Temperature: " + Temperature + "</p>")
   HTTPOut("<p><h2> Links :</h2></p>")
   HTTPOut("<p><a href="+ CHR(34) +"monitor.html"+ CHR(34)+">Monitor</a></p>")
   HTTPOut("</body>")
   HTTPOut("</html>")
WebPageEnd

Веб-страница отслеживания
WebPageBegin("monitor.html",Commands)
   HTTPOut("<html>")
   HTTPOut("<style>body {background-color: oldlace}</style>")
   HTTPOut("<body>")
   HTTPOut("<title>Monitor CR1000 Datalogger Tables</title>")
   HTTPOut("<p><h2>CR1000 Data Table Links</h2></p>")
   HTTPOut("<p><a href="+ CHR(34) + "command=TableDisplay&table=CRTemp&records=10" + _
   CHR(34)+">Display Last 10 Records from DataTable CR1Temp</a></p>")
   HTTPOut("<p><a href="+ CHR(34) + "command=NewestRecord&table=CRTemp"+ CHR(34) + _
      ">Current Record from CRTemp Table</a></p>")
   HTTPOut("<p><a href="+ CHR(34) + "command=NewestRecord&table=Public"+ CHR(34) + _
      ">Current Record from Public Table</a></p>")
```

```
   HTTPOut(”<p><a href=”+ CHR(34) + "command=NewestRecord&table=Status" + CHR(34) + _
      ”>Current Record from Status Table</a></p>")
   HTTPOut("<br><p><a href="+ CHR(34) +"default.html"+ CHR(34) + ">Back to the Home Page _
      </a></p>")
   HTTPOut("</body>")
   HTTPOut("</html>")
WebPageEnd

BeginProg
   Scan(1,Sec,3,0)
     PanelTemp(RefTemp,250)
     RealTime(Time())
     Minutes = FormatFloat(Time(5),"%02 .0f")
     Seconds = FormatFloat(Time(6),"%02.0f")
     Temperature = FormatFloat(RefTemp, "%02.02f")
     CallTable(CRTemp)
   NextScan
EndProg
```

8.6.3.4. Сервер FTP

На CR1000 работает автоматический FTP-сервер. Это позволяет с помощью *Windows® Explorer®* получать доступ к файловой системе CR1000 через FTP, причем диски CR1000 сопоставляются с директориями или папками. Корневой директорией CR1000 может быть любой диск, однако обычно диску USR: отдается предпочтение. USR: — это диск, созданный путем распределения памяти в поле **USR: размер диска** на вкладке **Развертывание | Расширенное** сервиса CR1000 в *DevConfig*. Файлы могут копироваться и передаваться между дисками. С помощью FTP можно удалять файлы.

8.6.3.5. FTP-клиент

CR1000 может выступать в роли FTP-клиента, отправляющего или получающего файл с сервера FTP, например другого регистратора данных или веб-камеры. Это выполняется с помощью команды CRBasic **FTPClient().** Подробная информация и примеры программ приведены в руководстве по сетевому подключению Campbell Scientific (см. приложение *Сетевые подключения* *(стр. 652)*), доступном на сайте *www.campbellsci.com* или в *Справке по редактору CRBasic*.

8.6.3.6. Telnet

Telnet используется для получения доступа к тем же командам, которые доступны в *терминальном эмуляторе ПО поддержки* *(стр. 530)*. Для запуска сеанса Telnet откройте командную строку DOS и наберите:

```
Telnet xxx.xxx.xxx.xxx <Enter>
```

где xxx.xxx.xxx.xxx — это IP-адрес сетевого устройства, подключенного к CR1000.

8.6.3.7. SNMP

Простой протокол сетевого управления (SNMP) является частью набора IP-протоколов, используемых в рамках NTCIP и RWIS для отслеживания дорожной ситуации. CR1000 поддерживает SNMP, если подключено сетевое устройство.

8.6.3.8. Ping (IP)

Команда ping может использоваться для проверки доступности IP-адреса сетевого устройства, подключенного к CR1000. Чтобы воспользоваться инструментом Ping, откройте командную строку на компьютере, подключенном к сети, и наберите:

```
ping xxx.xxx.xxx.xxx <Enter>
```

где xxx.xxx.xxx.xxx — это IP-адрес сетевого устройства, подключенного к CR1000.

8.6.3.9. Микро-последовательный сервер

CR1000 можно сконфигурировать таким образом, чтобы разрешить последовательную связь через порт TCP/IP. Это полезно при коммуникации с последовательным датчиком через Ethernet с помощью микро-последовательного сервера (сторонний последовательный интерфейс Ethernet), к которому подключен последовательный датчик. Более подробную информацию см. в руководстве по сетевым подключениям и в *Справке по редактору CRBasic* для команды **TCPOpen()**. Информация о доступных сетевых подключениях приведена в приложении *Сетевые подключения* *(стр. 652)*.

8.6.3.10. Modbus TCP/IP

CR1000 может осуществлять коммуникацию Modbus по TCP/IP с помощью интерфейса Modbus TCP/IP. Для настройки Modbus TCP/IP укажите порт 502 как ***ComPort*** в командах **ModBusMaster()** и **ModBusSlave()**. Более подробная информация приведена в *Справке по редактору CRBasic*. См. *Modbus* *(стр. 411)*.

8.6.3.11. DHCP

При подключении к серверу со списком IP-адресов, доступных для присвоения, CR1000 автоматически запрашивает и получает IP-адрес по протоколу динамической конфигурации хоста (DHCP). Как только адрес присвоен, используйте команду *DevConfig, PakBusGraph, Connect* или экранную клавиатуру (CR1000KD Keyboard Display), чтобы найти в таблице состояния CR1000 (**Status**) присвоенный IP-адрес. Он отображается под именем поля ***IPInfo***.

8.6.3.12. DNS

CR1000 предоставляет клиент сервера доменных имен (DNS), который может запрашивать у DNS-сервера определение сопоставления IP-адреса с именем хоста. При положительном ответе имя хоста может равноправно использоваться с IP-адресом в некоторых командах регистратора данных.

8.6.3.13. SMTP

Простой протокол электронной почты (SMTP) является стандартным для передачи электронной почты. CR1000 можно запрограммировать на отправку электронных сообщений в соответствии со стандартным графиком или при возникновении определенных событий.

8.6.3.14. Web API

Интерфейс Web API (**A**pplication **P**rogramming **I**nterface — интерфейс прикладного программирования) регистратора CR1000 является последовательностью команд *URL* *(стр. 532)*, которые управляют ресурсами CR1000. API обеспечивает выполнение следующих функций:

- Управление данными
  - сбор данных
- Управление — логика языка программирования CRBasic обеспечивает удаленный доступ ко множеству функций управления посредством изменения значения переменной.
  - настройка переменных/флагов/портов
- Функции синхронизации часов — функции синхронизации часов позволяют веб-клиенту отслеживать и настраивать часы реального времени узлового регистратора CR1000. См. раздел «Синтаксис времени» для получения более подробной информации.
  - настройка часов CR1000
- Управление файлами — команды Web API предоставляют веб-клиенту возможность управления файлами на флеш-накопителях узлового регистратора CR1000. Примером массивов данных, нуждающихся в частом управлении, могут служить файлы изображений камеры.
  - отправка программ;
  - отправка файлов;
  - сбор файлов.

Команды API также используются в сочетании с веб-сервером RTMC Campbell Scientific — *ПО поддержки регистратора данных* *(стр. 95)*. Следующая документация сфокусирована на использовании API в регистраторах CR1000. Более подробная информация по использованию команд API в RTMC содержится в *Справке по редактору CRBasic*, который является одной из немногих программ, предназначенных для поддержки в вопросах взаимодействия между *ПК и CR1000* *(стр. 95)*.

8.6.3.14.1. Аутентификация

Схема безопасности CR1000 на основе паролей, рассмотренная в разделе *Безопасность* *(стр. 92)*, не считается достаточно надежной для использования API по следующим причинам:

1. Код безопасности явно виден в URI, поэтому может быть получен путем несанкционированного извлечения информации или просмотра монитора.
2. Диапазон действительных кодов безопасности варьируется от 1 до 65534, поэтому код безопасности может быть получен методом подбора.

Вместо данной схемы в регистраторах CR1000 необходимо использовать алгоритм Basic Access Authentication (базовая аутентификация доступа), который реализован в API. Алгоритм Basic Access Authentication использует зашифрованный файл учетной записи пользователя, **.csipasswd**, который хранится на диске CPU: модуля CR1000.

Алгоритм Basic Access Authentication предусматривает четыре уровня доступа:

- все виды доступа запрещены (уровень **0**);
- все виды доступа разрешены (уровень **1**);
- настройка переменных разрешена (уровень **2**);
- доступ только для чтения (уровень **3**).

Можно задать различные учетные записи пользователей и уровни безопасности. Файл с именем **.csipasswd** создается на диске CPU: регистратора CR1000 и редактируется в *служебной программе конфигурации устройства (DevConfig)* *(стр. 111)* — вкладка **Net Services (Сетевые службы)**, кнопка **Edit .csipasswd File (Редактировать файл .csipasswd)**. Если в открывшемся диалоговом окне **Datalogger .csipasswd File Editor (Редактор файла .csipasswd регистратора данных)** ввести имя пользователя и пароль и нажать кнопку **Apply (Применить)**, файл .**csipasswd** шифруется и сохраняется на диске CPU: регистратора CR1000. Чтобы сделать файл невидимым,

предусматривается соответствующая кнопка-флажок. Если данный флажок отмечен при сохранении, файл невозможно открыть для редактирования.

Если предпринимается попытка получения доступа к веб-серверу CR1000 без корректных учетных данных, в ответ на такой запрос регистратор CR1000 отправляет ошибку **401 Требуется авторизация**. Данная ошибка вызывает открытие в веб-браузере или клиенте диалогового окна для ввода имени пользователя и пароля. Если файл **.csipasswd** пуст или не существует, для имени пользователя назначается значение по умолчанию **anonymous** с отсутствующим паролем, а уровень безопасности возвращается на уровень по умолчанию — **«только для чтения»** (уровень безопасности по умолчанию можно изменить в *DevConfig*). Если в файле **.csipasswd** введены недействительные имя пользователя или пароль, веб-сервер CR1000 восстанавливает уровень доступа по умолчанию, связанный с учетной записью **anonymous**.

Уровень безопасности, связанный с учетной записью **anonymous**, затрагивает только команды API. Например, команда API **SetValueEx** не будет работать, если уровень безопасности API установлен на **«только для чтения»**, однако параметр ***SetValue*** программы CRBasic в команде **WebPageBegin()** будет работать. Однако, если в файле **.csipasswd** определено имя пользователя, отличное от anonymous, и задан пароль, меры безопасности будут действительны для команд API и CRBasic. Например, если числовой код безопасности задается в таблице **Status (Состояние)** регистратора CR1000 (см. раздел *Security (Безопасность)* *(стр. 92)*), но при этом файл **.csipasswd** не существует, то код безопасности должен вводиться для использования параметра ***SetValue*** в CRBasic. Если файл **.csipasswd** существует, действительные имя пользователя и пароль имеют приоритет над кодом безопасности.

8.6.3.14.2. Синтаксис команды

Команды API соответствуют синтаксическим правилам,

```
ip_adr?command=CommandName&parameters/arguments
```

где

ip_adr = IP-адрес регистратора CR1000.
CommandName = команда API.
parameters / arguments = параметры команды API и связанные аргументы. **&** используется при сцеплении параметров и аргументов в командной строке.

Некоторые команды обладают дополнительными параметрами. При этом, если данный параметр исключается, то используется аргумент по умолчанию. Некоторые команды выдают код возврата, указывающий результат команды. В следующей таблице приводится список параметров и аргументов API, а также команд, в которых они используются. Параметры и аргументы по определенным командам приводятся в следующих разделах.

Таблица 109. Команды, параметры и аргументы API

Параметр	*Команды, в которых используется параметр*	*Функция параметра*	*Аргумент (аргументы)*
uri	• **BrowseSymbols** • **DataQuery** • **ClockSet** • **ClockCheck** • **ListFiles**	Определяет источник данных.	• ***source: dl*** (регистратор данных является источником данных): по умолчанию, применяется ко всем командам, перечисленным в столбце 2. • ***tablename.fieldname***: применяется только к **BrowseSymbols** и **DataQuery**

Таблица 109. Команды, параметры и аргументы API

Параметр	*Команды, в которых используется параметр*	*Функция параметра*	*Аргумент (аргументы)*
format	• **BrowseSymbols** • **DataQuery** • **ClockSet** • **ClockCheck** • **FileControl** • **ListFiles**	Задает формат ответа.	• ***html, xml, json:*** применяются ко всем командам, перечисленным в столбце 2. • ***toa5*** и ***tob1*** применяются только к **DataQuery**
mode	**DataQuery**	Задает диапазон данных, с помощью которых необходимо отвечать.	• ***most-recent*** • ***since-time*** • ***since-record*** • ***data-range*** • ***backfill***
p1	**DataQuery**	• Максимальное количество записей (если используется аргумент ***most-recent*** (новейшие)). • Начальная дата и/или время (если используются аргументы ***since-time*** (с определенного момента) или ***date-range*** (диапазон дат)). • Начальный номер записи (если используется аргумент ***since-record*** (с определенной записи)). • Интервал в секундах (если используется аргумент ***backfill*** (обратное заполнение)).	• Целочисленное количество записей (если используется аргумент ***most-recent*** (новейшие)). • Время в определенном формате (если используются аргументы ***since-time (с определенного момента)*** или ***daterange*** (диапазон дат), см. раздел *Синтаксис времени (стр. 427)*). • Целочисленный номер записи (если используется аргумент ***since-record*** (с определенной записи)). • Целое число секунд (если используется аргумент ***backfill*** (обратное заполнение)).
p2	**DataQuery**	Задает конечную дату и/или время, если используется аргумент ***date-range*** (диапазон дат).	Время выражается в определенном формате (см. раздел *Синтаксис времени (стр. 427)*)
value	**SetValueEx**	Задает новое значение.	Числовое или **string**
время	**ClockSet**	Задает установленное время.	Время в определенном формате
action	**FileControl**	Задает действие **FileControl**.	**От 1** до **20**
file	**FileControl**	Задает первый аргумент действия **FileControl**.	Имя файла с диском
file2	**FileControl**	Задает параметр второго аргумента действия **FileControl**.	Имя файла с диском

Таблица 109. Команды, параметры и аргументы API			
Параметр	***Команды, в которых используется параметр***	***Функция параметра***	***Аргумент (аргументы)***
expr	**NewestFile**	Задает путь и выражение с подстановочными знаками для заданного набора файлов, подлежащих сбору.	Путь и выражение с подстановочными знаками

8.6.3.14.3. Синтаксис времени

Команды API могут иметь параметр временной метки. См. раздел *Функции часов (стр. 578)* для получения более подробной информации. Параметр имеет следующий формат:

```
YYYY-MM-DDTHH:MM:SS.MS
```

где

YYYY = год, представленный четырьмя знаками
MM = месяц в году, один или два знака (от 1 до 12)
DD = дни в месяце, один или два знака (от 1 до 31)
HH = часы в дне, один или два знака (от 1 до 23)
MM = минуты в часу, один или два знака (от 1 до 59)
SS = секунды в минуте, один или два знака (от 1 до 59)
MS = доли секунды, дополнительно при установке времени, до девяти знаков (от 1 до <1E9)

Временные параметры ***2010-07-27T12:00:00.00*** и ***2010-07-27T14:00:00*** используются в следующем примере URL:

```
http://192.168.4.14/?command=dataquery&uri=d1:WSN30sec.CWS900_Ts
&format=htm1&mode=date-range&p1=2010-07-27T12:00:00&p2=2010-07-
27T14:00:00
```

8.6.3.14.4. Управление данными — команда BrowseSymbols

С помощью команды **BrowseSymbols** веб-клиент может опрашивать узловые регистраторы CR1000 для получение их структуры памяти данных. Структура памяти формируется из имен таблиц, имен полей и индексов массивов. Совокупно они именуются «символами». Команда **BrowseSymbols** имеет следующий вид:

```
http://ip_address/?command=BrowseSymbols&uri=source:tablename.field
name&format=html
```

Команде BrowseSymbols требуется минимальный уровень доступа **.csipasswd*3*** (только для чтения).

Таблица 110. Параметры команды BrowseSymbols API	
uri	Дополнительный. Задает *URI (стр. 532)* для источника данных. При опросе регистратора CR1000 параметры ***uri source, tablename*** и ***fieldname*** являются дополнительными. Если источник не задан, предполагается ***dl*** (CR1000). Имя поля всегда задается совместно с именем таблицы. Если имя поля не задано, выводятся все поля. Если ***fieldname*** ссылается на массив без индекса, выводятся все поля, связанные с данным массивом. Имя таблицы является дополнительным параметром. Если имя таблицы не используется, весь синтаксис URI не требуется.
format	Дополнительный. Задает формат ответа. Значения **html, json** и **xml** являются действительными. Если данный параметр опущен или если в качестве значения используется **html**, пустое или недействительное значение, ответом является HTML.

Например:

Команда для ответа, в котором символы для всех таблиц выводятся как HTML

```
http://192.168.24.106/?command=BrowseSymbols&uri=dl:public&format=html
```

Команда для ответа, в котором символы для всех полей в одиночной таблице (MainData) выводятся как HTML

```
http://192.168.24.106/?command=BrowseSymbols&uri=dl:MainData&format=html
```

Команда для ответа, в котором символы для одного поля (Cond41) выводятся как HTML

```
http://192.168.24.106/?command=BrowseSymbols&uri=dl:MainData.Cond41&format=html
```

Ответ BrowseSymbols

Параметр **format** ***команды*** BrowseSymbols определяет формат ответа. Если формат не задан, используется формат по умолчанию — HTML. Для получения более подробной информации по форматам данных ответа см. раздел *Форматы файлов данных* *(стр. 377)*.

Ответ состоит из набора описаний дочерних символов. Каждое из этих описаний включает следующие поля:

Таблица 111. Ответ на команду BrowseSymbols API

`name`	Задает имя символа, которое может быть именем источника данных, именем станции, именем таблицы или именем столбца.
`uri`	Задает uri дочернего символа.
`type`	Задает код для типа данного символа. Имеются следующие типы символов: **6** — таблица; **7** — массив; **8** — скаляр.
`is_enabled`	Булево значение, которому присваивается истинное значение, если символ включен для запланированного сбора. В основном, применяется к источникам данных *LoggerNet*.
`is_read_only`	Булево значение, которому присваивается истинное значение, если символ считается доступным только для чтения. Ложное значение указывает на предположение, что значение символа может быть изменено с помощью команды **SetValueEx**.
`can_expand`	Булево значение, которому присваивается истинное значение, если символ имеет дочерние значения, которые можно вывести с помощью команды **BrowseSymbols**.

Если клиент задает URI для символа, который не существует, сервер отвечает пустым набором символов.

Ответ HTML

Если значение ***html*** вводится для параметра **format** ***команды*** BrowseSymbols, ответом будет HTML. Ниже приводятся примеры ответов.

Ответ в виде таблицы HTML:

Ответ BrowseSymbols

name	uri	type	is_enabled	is_read_only	can_expand
Состояние	dl: Состояние	6	true	false	true
MainData	dl:MainData	6	true	false	true
BallastTank1	dl:BallastTank1	6	true	false	true
BallastTank2	dl:BallastTank2	6	true	false	true
Public	dl:Public	6	true	false	true

Источник HTML-страницы:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN”>
<html><head>
<title>BrowseSymbols Response</title>
</head>

<body>
<h1>BrowseSymbols Response</h1>

<table border="1">
<tr>

<th>name</th><th>uri</th><th>type</th><th>is_enabled</th><th>is_
read_only</th><th>can_expand</th></tr><tr>

<td>Status</td><td>dl:Status</td><td>6</td><td>true</td><td>fals
e</td><td>true</td></tr><tr>

<td>MainData</td><td>dl:
MainData</td><td>6</td><td>true</td><td>
false</td><td>true</td></tr><tr>

<td>BallastTank1</td><td>dl:BallastTank1</td><td>6</td><td>true<
/td><td>false</td><td>true</td></tr><tr>

<td>BallastTank2</td><td>dl:BallastTank2</td><td>6</td><td>true<
/td><td>false</td><td>true</td></tr><tr>

<td>BallastTank3</td><td>dl:BallastTank3</td><td>6</td><td>true<
/td><td>false</td><td>true</td></tr><tr>

<td>BallastTank4</td><td>dl:BallastTank4</td><td>6</td><td>true<
/td><td>false</td><td>true</td></tr><tr>

<td>BallastLine</td><td>dl:BallastLine</td><td>6</td><td>true</t
d><td>false</td><td>true</td></tr><tr>

<td>Public</td><td>dl:Public</td><td>6</td><td>true</td><td>fals
e</td><td>true</td></tr>
</table>

</body></html>
```

Ответ XML

Если значение ***xml*** вводится для параметра *format* команды **BrowseSymbols**, ответ будет отформатирован как *CSIXML* *(стр. 90)* с присвоением имени корневого элемента **BrowseSymbolsResponse**. Ниже приводится пример ответа.

Пример вывода источника страницы:

```
<BrowseSymbolsResponse>
..<symbol
   name="Status"
   uri=”dl: Status"
   type="6"
   is_enabled="true"
   is_read_only="false"
   can_expand="true"/><symbol
   name="MainData"
   uri="dl:MainData"
   type="6"
   is_enabled="true"
   is_read_only="false"
   can_expand="true"/><symbol
   name="BallastTank1"
   uri="dl:BallastTank1"
   type="6"
   is_enabled="true"
   is_read_only="false"
   can_expand="true"/><symbol
   name="BallastTank2"
   uri="dl:BallastTank2"
   type="6"
   is_enabled="true"
   is_read_only="false"
   can_expand="true"/><symbol
   name="BallastTank3"
   uri="dl:BallastTank3"
   type="6"
   is_enabled="true"
   is_read_only="false"
   can_expand="true"/><symbol
   name="BallastTank4"
   uri="dl:BallastTank4"
   type="6"
   is_enabled="true"
   is_read_only="false"
   can_expand="true"/><symbol
   name="BallastLine"
   uri="dl:BallastLine"
   type="6"
   is_enabled="true"
   is_read_only="false"
   can_expand="true"/><symbol
   name="Public"
   uri="dl: Public"
   type="6"
   is_enabled="true"
   is_read_only="false"
   can_expand="true"/>
</BrowseSymbolsResponse>
```

Ответ JSON

Если значение *json* вводится для параметра *format* команды **BrowseSymbols**, ответ будет отформатирован как *CSIJSON* *(стр. 90)*. Ниже приводится пример ответа.

```
{
  "symbols": [
   {"name": "Status","uri": "dl:Status","type": 6,"is_enabled":
true,"is_read_only": false,"can_expand": true},
{"name": "MainData","uri": "dl:MainData","type":
6,"is_enabled": true,"is_read_only": false,"can_expand": true},
   {"name": "BallastTank1","uri": "dl:BallastTankl","type":
6,"is_enabled": true,"is_read_only": false,"can_expand": true},
   {"name": "BallastTank2","uri": "dl:BallastTank2","type":
6,"is_enabled": true,"is_read_only": false,"can_expand": true},
   {"name": "BallastTank3","uri": "dl:BallastTank3","type":
6,"is_enabled": true,"is_read_only": false,"can_expand": true},
   {"name": "BallastTank4","uri": "dl:BallastTank4","type":
6,"is_enabled": true,"is_read_only": false,"can_expand": true},
   {"name": "BallastLine","uri": "dl:BallastLine","type":
6,"is_enabled": true,"is_read_only": false,"can_expand": true},
   {"name": "Public","uri": "dl:Public","type": 6,"is_enabled":
true,"is_read_only": false,"can_expand": true}
]
}
```


8.6.3.14.5. Управление данными — команда DataQuery

С помощью команды **DataQuery** веб-клиент может опрашивать регистратор CR1000 для получения данных. Обычно команда **DataQuery** имеет следующий вид:

```
http://ip_address/?command=DataQuery&uri=dl:tablename.fiel
dname& format=_&mode=_&p1=_&p2=_
```

Команде **DataQuery** требуется минимальный уровень доступа **.csipasswd3** (только для чтения).

Таблица 112. Параметры команды DataQuery API

`uri`	Дополнительный. Задает *URI (стр. 532)* для запрашиваемых данных. Синтаксис: ***dl:tablename.fieldname***. Имя поля является дополнительным параметром. Имя поля всегда задается совместно с именем таблицы. Если имя поля не задано, выводятся все поля. Если fieldname ссылается на массив без индекса, выводятся все значения, связанные с данным массивом. Имя таблицы является дополнительным параметром. Если имя таблицы не используется, весь синтаксис URI не требуется, поскольку ***dl*** (CR1000) является источником данных по умолчанию.
`mode`	Обязательный. Режимы для временного диапазона собранных данных: ***most-recent*** (новейшие) возвращает данные, начиная с новейшего номера записи. ***p1*** задает максимальное число записей. ***since-time*** (с определенного момента) возвращает новейшие данные с определенного момента. ***p1*** задает начальную временную метку (см. раздел *Синтаксис времени (стр. 427)*). ***since-record*** (с определенной записи) возвращает *записи (стр. 525)*, начиная с определенного номера записи. Номер записи задается посредством ***p1***. Если номер записи не присутствует в таблице, регистратор CR1000 возвращает все данные, начиная с самой ранней записи. ***date-range*** (диапазон дат) возвращает данные в определенном диапазоне дат. Диапазон дат задается с помощью ***p1*** и ***p2***. Возвращаемые данные содержат данные, начиная с даты заданной ***p1***, а не ***p2*** (полуоткрытый интервал). ***backfill*** (обратное заполнение) возвращает данные, хранящиеся в течение определенного интервала (например, все данные за последний 1 час). Интервал задается в секундах с помощью ***p1***.

<table>
<tr><td>p1</td><td>Дополнительный. Задает:
Максимальное количество записей (most-recent)
Начальная дата и/или время (since-time, date-range). См. раздел Синтаксис времени (стр. 427) для получения информации по формату.
Начальный номер записи (since-record)
Интервал в секундах (backfill)</td></tr>
<tr><td>p2</td><td>Дополнительный. Задает:
конечная дата и/или время (date-range). См. раздел Синтаксис времени (стр. 427) для получения информации по формату.</td></tr>
<tr><td>format</td><td>Дополнительный. Задает формат выводимых данных. Если данный параметр опущен или если в качестве значения используется html, пустое или недействительное значение, данные выводятся в формате HTML.
<table>
<tr><th>format Option</th><th>Формат выводимых данных</th><th>Поле типа контента ответного заголовка HTTP</th></tr>
<tr><td>html</td><td>HTML</td><td>text/html</td></tr>
<tr><td>xml</td><td>CSIXML</td><td>text/xml</td></tr>
<tr><td>json</td><td>CSIJSON</td><td>application/json</td></tr>
<tr><td>toa5</td><td>TOA5</td><td>text/csv</td></tr>
<tr><td>tob1</td><td>TOB1</td><td>binary/octet-stream</td></tr>
</table>
Примечание. Если используется json и веб-сервер располагает большим набором данных для отправки, веб-сервер может разбить данные на несколько запросов, установив значение true для флага more в выводе CSIJSON. Флаг more не отображается, если полный набор данных возвращается в первый раз.</td></tr>
</table>

Например:

Команда:

```
http://192.168.24.106/?command=DataQuery&uri=dl:MainData&mode=da
te-range&p1=2012-09-14T8:00:00&p2=2012-09-14T9:00:00
```

Ответ: получить все данные из таблицы MainData в пределах диапазона от p1 до p2

Команда:

```
http://192.168.24.106/?command=DataQuery&uri=d1:MainData.Cond41&
format=htm1&mode=most-recent&p1=70
```

Ответ: получить пять последних записей из таблицы MainData

Команда:

```
http://192.168.24.106/?command=DataQuery&uri=d1:MainData.Cond41&
format=html&mode=since-time&p1=2012-09-14T8:00:00
```

Ответ: получить все записи поля Cond41, начиная с заданной даты и времени

Команда:

```
http://192.168.24.106/?command=DataQuery&uri=dl:MainData.Cond41&
format=html&mode=since-record&p1=4700
```

Ответ: получить все записи, начиная с заданной записи

Команда:

```
http://192.168.24.106/?command=DataQuery&uri=dl:MainData.Cond41&
format=html&mode=backfill&p1=7200
```

Ответ: выполнить обратной заполнение, начиная с момента в 3600 секунд назад

Ответ на команду DataQuery

Параметр **format** ***команды*** DataQuery определяет формат ответа.
Для получения более подробной информации по форматам данных ответа см. раздел *Форматы файлов данных* *(стр. 377)*.

Если значение ***html*** вводится для параметра *format* команды **DataQuery**, ответом будет HTML. Ниже приводятся примеры ответов.

Ответ HTML

Ответ в виде таблицы HTML:

Имя таблицы: BallastLine

TimeStamp	Запись	Induced_Water
2012-08-21 22:41:50.0	104	66
2012-08-21 22:42:00.0	105	66
2012-08-21 22:42:10.0	106	66
2012-08-21 22:42:20.0	107	66
2012-08-21 22:42:30.0	108	66

Источник HTML-страницы:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN”
"http ://www.w3.org/TR/htm14/loose.dtd">
<HTML><HEAD><TITLE>Table Display</TITLE><meta http-
equiv="Pragma" content="no-cache"><meta http-equiv="expires"
content="0">
</HEAD><BODY>
<h1>Table Name: BallastLine</h1>
<table border="1" cellpadding="2" cellspacing="0">
<tr valign="middle" align="center">
<th nowrap>TimeStamp</th>
<th nowrap>Record</th>
<th nowrap>Induced_Water</th>
</tr>
<tr valign="middle" align="center">
<td nowrap>2012-08-21 22:41:50.0</td>
<td nowrap>104</td>
<td nowrap>66</td>
</tr>
<tr valign="middle" align="center">
<td nowrap>2012-08-21 22:42:00.0</td>
<td nowrap>105</td>
<td nowrap>66</td>
</tr>
<tr valign="middle" align="center">
<td nowrap>2012-08-21 22:42:10.0</td>
<td nowrap>106</td>
<td nowrap>66</td>
```

```
</tr>
<tr valign="middle" align="center">
<td nowrap>2012-08-21 22:42:20.0</td>
<td nowrap>107</td>
<td nowrap>66</td>
</tr>
<tr valign="middle" align="center">
<td nowrap>2012-08-21 22:42:30.0</td>
<td nowrap>108</td>
<td nowrap>66</td>
</tr>
</table>
</BODY></HTML>
```

Ответ XML

Если значение ***xml*** вводится для параметра *format* команды **DataQuery**, ответ будет отформатирован как CSIXML. Ниже приводится пример ответа.

```
<?xml version="1.0" standalone="yes"?>
<csixml version="1.0">
<head>
<environment>
<station-name>Q2</station-name>
<table-name>BallastLine</table-name>
<model>CR1000</model>
<serial-no>18583</serial-no>
<os-version>CR1000.Std.25</os-version>
<dld-name>CPU:IndianaHarbor_081712.CR1</dld-name>
<dld-sig>33322</dld-sig>
</environment>
<fields>
<field name="Induced_Water" type="xsd:float"
process="Smp"/></fields>
</head>
<data>
<r time="2012-08-21T22:41:50" no="104">
<v1>66</v1></r><r time="2012-08-21T22:42:00" no="105">
<v1>66</v1></r><r time="2012-08-21T22:42:10" no="106">
<v1>66</v1></r><r time="2012-08-21T22:42:20" no="107">
<v1>66</v1></r><r time="2012-08-21T22:42:30" no="108">
<v1>66</v1></r></data>
</csixml>
```

Ответ JSON

Если значение ***json*** вводится для параметра *format* команды **DataQuery**, ответ будет отформатирован как CSIJSON. Ниже приводится пример ответа.

```
{
.."head": {
...."transaction": 0,
...."signature": 26426,
.... "environment": {
.... "station_name":       "Q2",
.... "table_name": "BallastLine",
.... "model": "CR1000",
.... "serial_no": "18583",
.... "os_version": "CR1000.Std.25",
.... "prog_name": "CPU :IndianaHarbor_081712.CR1"
```

```
....},
...."fields": [{
.... "name": "Induced_Water",
.... "type": "xsd:float",
.... "process": "Smp",
.... "settable": false}]
    },
.... "data": [{
.... "time": "2012-08-21T22:41:50",
.... "no":  104,
.... "vals": [66]
    },{
.... "time": "2012-08-21T22:42:00",
.... "no":  105,
.... "vals": [66]
    },{
.... "time": "2012-08-21T22:42:10",
.... "no":  106,
.... "vals": [66]
    },{
.... "time": "2012-08-21T22:42:20",
.... "no":  107,
.... "vals": [66]
    },{
.... "time": "2012-08-21T22:42:30",
.... "no":  108,
.... "vals": [66]
}]}
```

Ответ TOA5

Если значение ***toa5*** вводится для параметра *format* команды **DataQuery**, ответ будет отформатирован как Campbell Scientific TOA5. Ниже приводится пример ответа.

```
"TOA5","TXSoil","CR1000","No_SN","CR1000.Std.25","TexasRun_1b.CR
2","12645","_1Hr"
"TIMESTAMP","RECORD","ID","_6_inch","One","Two","Three","Temp_F_
Avg","Rain_in_Tot"
"TS","RN","","","","","","","","
"" ,"",  "Smp","Smp","Smp","Smp","Smp","Avg","Tot"
"2012-05-03 17:00:00",0,0,-0.8949984,-0.95232,-0.8949984,-
0.8637322,2.144136,0.09999999
"2012-05-03 18:00:00",1,0,-0.9106316,-0.9731642,-0.9210536,-
0.8845763,72.56885,0
"2012-05-03 19:00:00",2,0,-0.9210536,-0.9679532,-0.9106316,-
0.8637322,72.297,0
"2012-05-03 20:00:00",3,0,-0.8624293,-0.9145398,-0.8624293,-
0.8311631,72.68445,0
"2012-05-03 21:00:00",4,0,-0.8949984,-0.9471089,-0.9002095,-
0.8585211,72.79237,0
"2012-05-03 22:00:00",5,0,-0.9262648,-0.9731642,-0.9158427,-
0.8793653,72.75194,0
"2012-05-03 23:00:00",6,0,-0.8103188,-0.8624293,-0.8103188,-
0.7686304,72.72644,0
"2012-05-04 00:00:00",7,0,-0.9158427,-0.9627421,-0.9158427,-
0.8689431,72.67271,0
"2012-05-04 01:00:00",8,0,-0.8598238,-0.9015122,-0.8598238,-
0.8129244,72.64571,0
"2012-05-04 02:00:00",9,0,-0.9158427,-0.9575311,-0.9054205,-
0.8689431,72.5931,0
```

```
”2012-05-04 03:00:00",10,0,-0.8754569,-0.9275675,-0.8910902,-
0.8546127,72.53336,0
"2012-05-04 04:00:00",11,0,-0.8949984,-0.9575311,-0.9106316,-
0.8793653,72.47779,0
"2012-05-04 05:00:00",12,0,-0.9236593,-0.9705587,-0.908026,-
0.8715487,72.4006,0
"2012-05-04 06:00:00",13,0,-0.9184482,-0.9601365,-0.902815,-
0.8819707,72.23279,0
"2012-05-05 11:00:00",0,5,-0.9106316,-0.941898,-0.8897874,-
0.8637322,4.740396,0
"2012-05-05 12:00:00",1,5,-0.9067233,-0.9640449,-0.9015122,-
0.8702459,71.16611,0
"2012-05-05 13:00:00",2,5,-0.8897874,-0.9366869,-0.8793653,-
0.8428879,70.93591,0
"2012-05-05 14:00:00",3,5,-0.9041178,-0.9510173,-0.8884846,-
0.8676404,70.78558,0
"2012-05-05 15:00:00",4,5,-0.9002095,-0.9627421,-0.9002095,-
0.8689431,70.66192,0
"2012-05-05 16:00:00",5,5,-0.9054205,-0.95232,-0.9054205,-
0.8741542,70.53237,0
"2012-05-05 17:00:00",6,5,-0.9158427,-0.9731642,-0.9002095,-
0.8637322,70.4076,0
"2012-05-05 18:00:00",7,5,-0.9223565,-0.969256,-0.9015122,-
0.8910902,70.33669,0
"2012-05-05 19:00:00",8,5,-0.8923929,-0.9445034,-0.8923929,-
0.8507045,70.25033,0
"2012-05-05 20:00:00",9,5,-0.9119344,-0.9640449,-0.9171454,-
0.8754569,70.1702,0
"2012-05-05 21:00:00",10,5,-0.930173,-0.9822836,-0.9197509,-
0.8832736,70.1116,0
"2012-05-05 22:00:00",11,5,-0.9132372,-0.9653476,-0.908026,-
0.8611265,70.0032,0
"2012-05-05 23:00:00",12,5,-0.9353842,-0.9822836,-0.930173,-
0.8936957,69.83805,0
```

Ответ TOB1

Если значение tob5 вводится для параметра *format* команды **DataQuery**, ответ будет отформатирован как Campbell Scientific TOB5. Ниже приводится пример ответа.

Пример:

```
"TOB1","11467","CR1000","11467","CR1000.Std.20","CPU
:file format.CR1","61449","Test"
"SECONDS","NANOSECONDS","RECORD","battfivoltfiMin","
PTemp"
"SECONDS","NANOSECONDS","RN","",""
"","","","Min","Smp"
"ULONG","ULONG","ULONG","FP2","FP2"
376
}Ÿp' E1HŒŸp' E1H>Ÿp' E1HaŸp' E1H1Ÿp'
E1H
```

8.6.3.14.6. Управление — команда SetValueEx

С помощью команды **SetValueEx** веб-клиент может устанавливать значение переменной CRBasic для узлового регистратора CR1000.

```
http://ip_address/?command=SetValueEx&uri=dl:table.variable&valu
e=x.xx
```

Команде **SetValueEx** требуется минимальный уровень доступа **.csipasswd2** (разрешена настройка переменных).

Таблица 113. Параметры команды SetValueEx API

`uri`	Задает переменную, которую необходимо настроить в следующем формате: dl:tablename.fieldname
`value`	Задает переменную, которую необходимо настроить
`format`	В следующей таблице приводится список дополнительных форматов вывода для результирующих кодов **SetValueEx**. Если значение не задано, результирующие коды выводятся в формате HTML. ***Значение вывода результирующего кода*** / ***Формат вывода результирующего кода*** / ***Поле типа контента ответного заголовка HTTP*** ***html*** / HTML / ***text/html*** ***json*** / CSIJSON / ***application/json*** ***xml*** / CSIXML / ***text/xml*** Пример: ***&format=html*** Задает формат ответа. Значения **html, json** и **xml** являются действительными. Если данный параметр опущен или если в качестве значения используется **html**, пустое или недействительное значение, ответом является HTML.

Например:

```
http://192.168.24.106/?command=SetValueEx&uri=dlipublic.NaOH_Set
pt_Bal2&value=3.14
```

Ответ: общей переменной settable_float присвоено значение 3.14.

```
http://192.168.24.106/?command=SetValueEx&uri=dl:public.flag&val
ue=-1&format=html
```

Ответ: общей булевой переменной Flag(1) присвоено значение True (-1).

Ответ SetValueEx

Параметр *format* команды **SetValueEx** определяет формат ответа. Если формат не задан, используется формат по умолчанию — HTML.Для получения более подробной информации по форматам данных ответа см. раздел *Форматы файлов данных* *(стр. 377)*.

В ответах содержится два поля. В выводимых данных в формате XML, поля являются атрибутами.

Таблица 114. Ответ на команду SetValue API

outcome	**0** — произошла неопознанная ошибка; **1** — успешное выполнение; **5** — только чтение; **6** — недействительное имя таблицы; **7** — недействительное имя поля; **8** — недействительный индекс имени поля; 9 — недействительный тип данных; **10** — отказ при обмене данными с регистратором данных; **12** — блокировка средствами безопасности регистратора данных; **15** — недействительная авторизация веб-клиента.
description	Текстовое описание выводимого кода.

Ответ HTML

Если значение ***html*** вводится для параметра *format* команды **SetValueEx**, ответом будет HTML. Ниже приводятся примеры ответов.

Ответ в виде таблицы HTML:

SetValueExResponse

outcome	outcome-code
description	description-text

Источник HTML-страницы:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html><head>
<title>SetValueExResponse</title>
</head>

<body>
<h1>SetValueExResponse</h1>

<table border="1">
   <tr>
        <td>outcome</td>
        <td>outcome-code</td>
   </tr>
   <tr>
        <td>description</td>
        <td>description-text</td>
   </tr>
</table>


</body></html>
```

Ответ XML

Если значение ***xml*** вводится для параметра *format* команды **SetValueEx**, ответ будет отформатирован как CSIXML с присвоением имени корневого элемента **SetValueExResponse**. Ниже приводится пример ответа.

```
<SetValueExResponse outcome="outcome-code"
descri ption="descri ption-text"/>
```

Ответ JSON

Если значение ***json*** вводится для параметра *format* команды **SetValueEx**, ответом будет CSIJSON. Ниже приводится пример ответа.

```
{
"outcome": outcome-code,
"description": description
}
```

8.6.3.14.7. Функции часов — команда ClockSet

С помощью команды **ClockSet** веб-клиент может настраивать часы реального времени в CR1000. Команда **ClockSet** имеет следующий вид:

```
http://ip_address/?command=ClockSet&format=html&time=YYYY-MM-
DDTHH:MM:SS.MS
```

Команде ClockSet требуется минимальный уровень доступа **.csipasswd1** (разрешен полный доступ).

Таблица 115. Параметры команды ClockSet API	
`uri`	Если данный параметр исключен или если ему присвоено значение «datalogger» (**uri=dl**) имеет пустую строку (**uri=**), то команда направляется в веб-сервер регистратора CR1000.[1]
`format`	Задает формат ответа. Значения **html, json** и **xml** являются действительными. Если данный параметр опущен или если в качестве значения используется **html**, пустое или недействительное значение, ответом является HTML.
`время`	Задает время, на основе которого устанавливаются часы реального времени в CR1000. Данное значение должно соответствовать формату, описанному для входных временных меток в разделе *Синтаксис времени* *(стр. 427)*.
[1] Дополнительно задает URI для настраиваемой исходящей станции *LoggerNet*	

Пример:

```
http://192.168.24.106/?command=ClockSet&format=html&time=2012-9-
14T15:30:00.000
```

Ответ: установить часы реального времени узлового CR1000 на 3:30 PM 14 сентября 2012.

Ответ ClockSet

Параметр *format* команды **ClockSet** определяет формат ответа. Если формат не задан, используется формат по умолчанию — HTML. Для получения более подробной информации по форматам данных ответа см. раздел *Форматы файлов данных* *(стр. 377)*.

В ответах содержатся три поля, как описано в следующей таблице:

Таблица 116. Ответ на команду ClockSet API	
`outcome`	**1** — часы были настроены; **5** — отказ при обмене данными с CR1000; **6** — обмен данными с CR1000 отключен; **8** — был задан недействительный URI.
`время`	Задает значение часов CR1000, перед тем как изменить его.
description	Строка, описывающая результирующий код.

Ответ HTML

Если значение ***html*** вводится для параметра *format* команды **ClockSet**, ответом будет HTML. Ниже приводятся примеры ответов.

Ответ в виде таблицы HTML:

Ответ ClockSet

outcome	1
время	2011-12-01 11:42:02.75
description	Часы были настроены

Источник HTML-страницы:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN”
"http://www.w3.org/TR/htm14/loose.dtd">
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN"><html>
<head><title>ClockSet Response</title></head>
<body>
<h1>ClockSet Response</h1>
<table border="1">
<tr><td>outcome</td><td>1</td>
</tr><td>time</td>
<td>2011-12-01 11:42:02.75</td>
</tr><tr><td>description</td><td>The clock was
set</td></tr></table></body></html>
```

Ответ XML

Если значение ***xml*** вводится для параметра *format* команды **ClockSet**, ответ будет отформатирован как *CSIXML* *(стр. 90)* с присвоением имени корневого элемента **ClockSetResponse**. Ниже приводится пример ответа.

```
<ClockSetResponse outcome="1" time="2011-12-01T11:41:21.17"
description=”The clock was set"/>
```

Ответ JSON

Если значение ***json*** вводится для параметра *format* команды **ClockSet**, ответ будет отформатирован как *CSIJSON* *(стр. 90)*. Ниже приводится пример ответа.

```
{"outcome": 1,"time": "2011-12-01T11:40:32.61","description": "
The clock was set"}
```

8.6.3.14.8. Функции часов — команда ClockCheck

Команда **ClockCheck** позволяет веб-клиенту считывать значение часов реального времени из узлового CR1000. Команда DataQuery имеет следующий вид:

```
http ://ip_address/?command=ClockCheck&format=html
```

Команде **ClockCheck** требуется минимальный уровень доступа **.csipasswd3** (только для чтения).

Таблица 117. Параметры команды ClockCheck API	
`uri`	Если данный параметр исключен или если ему присвоено значение «datalogger» (**uri=dl**) имеет пустую строку (**uri=**), то возвращаются данные часов реального времени узлового CR1000.[1]
`format`	Задает формат ответа. Значения **html, json** и **xml** являются действительными. Если данный параметр опущен или если в качестве значения используется **html**, пустое или недействительное значение, ответом является HTML.
[1] Дополнительно задает URI для проверяемой исходящей станции *LoggerNet*	

Пример:

```
http://192.168.24.106/?command=ClockCheck&format=html
```

Ответ: проверяет часы реального времени узлового CR1000 запрашивает ответ в виде HTML-таблицы.

Ответ ClockCheck

Параметр *format* команды **ClockCheck** определяет формат ответа. Если формат не задан, используется формат по умолчанию — HTML. Для получения более подробной информации по форматам данных ответа см. раздел *Форматы файлов данных* *(стр. 377)*.

В ответах содержатся три поля, как описано в следующей таблице:

Таблица 118. Ответ на команду ClockCheck API

`outcome`	Коды, которые задают результат выполнения команды **ClockCheck**. Коды, выделенные серым шрифтом, не являются действительными вводами для CR1000: **1** — часы были проверены; **2** — часы были настроены[1]; **3** — отказ во время сеанса *LoggerNet*; **4** — недействительный вход в систему *LoggerNet*; **5** — блокирование средствами безопасности *LoggerNet*; **6** — отказ при обмене данными с заданной станцией; **7** — обмен данными с заданной станцией отключен; **8** — блокировка средствами безопасности регистратора данных; **9** — недействительное имя станции LoggerNet; **10** — устройство *LoggerNet* занято; **11** — заданный URI не содержит ссылки на станцию *LoggerNet*.
`время`	Задает текущее значение часов реального времени CR1000[2]. Данное значение будет действительно, только если для значения результата устанавливается **1**. Данное значение будет отформатировано таким же образом, каким форматируются временные метки записи для ответа на команду **DataQuery**.
description	Текстовая строка, описывающая результат.

[1] *LoggerNet* может комбинировать новую транзакцию проверки часов с подлежащими обработке транзакциями настройки часов *LoggerNet*
[2] или сервер *LoggerNet*

Ответ HTML

Если значение ***html*** вводится для параметра *format* команды **ClockCheck**, ответом будет HTML. Ниже приводятся примеры ответов.

Ответ в виде таблицы HTML:

Ответ ClockCheck

outcome	1
время	2012-08-24 15:44:43 59
description	Часы были проверены

Источник HTML-страницы:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/htm14/loose.dtd">
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN"><html>
<head><title>ClockCheck Response</title></head>
<body>
<h1>ClockCheck Response</h1>
<table border="1">
<tr><td>outcome</td><td>1</td>
</tr><td>time</td>
<td>2012-08-24 15:44:43.59</td>
</tr><tr><td>description</td><td>The clock was checked</td></tr>
</table></body></html>
```

Ответ XML

Если значение ***xml*** вводится для параметра *format* команды **ClockCheck**, ответ будет отформатирован как *CSIXML* *(стр. 90)* с присвоением имени корневого элемента **ClockCheckResponse**. Ниже приводится пример ответа.

```
<ClockCheckResponse outcome="1" time="2012-08-24T15:50:50.59"
description="The clock was checked"/>
```

Ответ JSON

Если значение ***json*** вводится для параметра *format* команды **ClockCheck**, ответ будет отформатирован как *CSIJSON* *(стр. 90)*. Ниже приводится пример ответа.

Пример:

```
{
   "outcome": 1,
   "time": "2012-08-24T15:52:26.22",
   "description": " The clock was checked"
}
```

8.6.3.14.9. Управление файлами — отправка файла в регистратор данных

Файл можно отправить в CR1000 с помощью запроса **HTTPPut**. Для отправки файла требуется уровень доступа к файлу **.csipasswd** не ниже **1** (разрешен полный доступ). В отличие от других команд web API, формирование запроса PUT из адресной строки браузера невозможно. Вместо этого, используйте JavaScript на веб-странице или программу *Curl. exe. Программа Curl. exe* находится в папке с программными файлами *LoggerNet RTMC* или ее можно скачать на сайте http://curl.haxx.se. Командная строка программы *Curl.exe* имеет следующий вид (параметры командной строки описываются в сопутствующей таблице):

```
curl -XPUT -v -S -T "filename.ext" --user username:password
http://IPAdr/drive/
```

Таблица 119. Параметры запроса Curl HTTPPut

Параметр	*Описание*
`-XPUT`	Дает распоряжение программе *Curl.exe* использовать команду **HTTPPut**
`-v`	Дает распоряжение программе *Curl.exe* распечатать все данные, выводимые на экран
`-S`	Дает распоряжение программе *Curl.exe* отобразить ошибки
`-T «filename.ext»`	Имя файла для отправки в CR1000 (указывается в кавычках)
`username`	Имя пользователя в файле .csipasswrd

password	Пароль в файле .csipasswrd
IPAdr	IP-адрес регистратора CR1000
drive	Флеш-накопитель регистратора CR1000

Например:

Чтобы загрузить операционную систему в CR1000, откройте окно ввода команды («окно DOS») и выполните следующую команду в виде непрерывной строки:

```
curl -XPUT -v -S -T
"c:\campbellsci\lib\OperatingSystems\CR1000.Std.25.obj" --user
harri sonford:lostark1 http ://192.168.24.106/cpu/
```

Ответ:

```
* About to connect() to 192.168.7.126 port 80 (#0)
* Trying 192.168.7.126... connected
* Connected to 192.168.7.126 (192.168.7.126) port 80 (#0)
* Server auth using Basic with user 'fredtest'
>PUT /cpu/myron%22Ecr1 HTTP/1.1
>Authorization: Basic ZGF2ZW11ZWs6d29vZnk5NTU1
>User-Agent: curl/7.21.1 (i386-pc-win32) libcurl/7.21.1
OpenSSL/0.9.8o zlib/1.2.5 libidn/1.18 libssh2/1.2.6
>Host: 192.168.7.126
>Accept:*/*
>Content-Length: 301
>Expect: 100-continue
>
*Done waiting for 100-continue
<HTTP/1.1 200 OK
<Date: Fri, 2 Dec 2011 05:31:50
<Server: CR1000.Std.25
<Content-Length: 0
<
* Connection #0 to host 192.168.7.126 left intact
* Closing connection #0
```

Когда файл с расширением .OBJ загружается на диск CPU: регистратора CR1000, последний распознает файл как новую операционную систему (ОС) и фактически не загружает его на диск CPU:. Скорее, он сохраняет его. По завершении сохранения CR1000 перезагружается и компилирует новую ОС таким же способом, как если бы файл был отправлен через *ПО поддержки регистратора данных (стр. 95)* — экран **Connect** (Подключить).

Другие файлы, отправленные на диск регистратора CR1000, работают так же, как если бы они работали в *ПО поддержки регистратора данных (стр. 95)* — **File Control (Управление файлами)**. Исключением является то, что невозможно задать настройки выполнения программы CRBasic. Чтобы обеспечить выполнение программного файла, используйте команду **FileControl** интерфейса web API. Программу Curl.exe можно использовать для выполнения обеих операций, как показано далее.

Загрузите программу на диск CPU: регистратора CR1000 (в конце URL должно указываться /**cpu**/):

```
curl -XPUT -v -S -T "program.CR1" --user username:password
"http://192.168.24.106/cpu/"
```

Скомпилировать и выполнить программу и отметить ее как программу, подлежащую выполнению при включении электропитания. - **XGET** не требуется, поскольку это команда по умолчанию в программе Curl.exe.

```
curl -v -S --user username:password
"http://192.168.24.106/?command=FileControl&file=CPU:program.CR1
&action=1"
```

Обе операции можно скомбинировать в командном файле.

8.6.3.14.10. Управление файлами — команда FileControl

С помощью команды **FileControl** веб-клиент может выполнять операции с файловой системой на узловом регистраторе CR1000. Команда **FileControl** имеет следующий вид:

```
http://ip_address/?command=FileControl&file=drive:filename.dat&a
ction=x
```

Команде FileControl требуется уровень доступа к файлу **.csipasswd** не ниже **1** (разрешен полный доступ).

Таблица 120. Параметры API команды FileControl

action	**1** — Скомпилировать и выполнить файл, заданный параметром ***file***, и отметить его как программу, подлежащую выполнению при включении электропитания. **2** — Отметить файл, заданный параметром ***file***, как программу, подлежащую выполнению при включении электропитания. **3** — Отметить файл, заданный параметром ***file***, как скрытый. **4** — Удалить файл, заданный параметром ***file***. **5** — Отформатировать устройство, заданное параметром ***file***. **6** — Скомпилировать и выполнить файл, заданный параметром ***file***, не удаляя существующие таблицы данных. **7** — Остановить выполняемую на текущий момент программу. **8** — Остановить выполняемую на текущий момент программу и удалить связанные таблицы данных. **9** — Выполнить полный сброс памяти. **10** — Скомпилировать и выполнить программу, заданную параметром ***file***, но не изменять программу, на текущий момент отмеченную для выполнения при включении электропитания. **11** — Приостановить выполнение выполняемой на текущий момент программы. **12** — Возобновить выполнение выполняемой на текущий момент программы. **13** — Остановить выполняемую на текущий момент программу, удалить связанные с ней таблицы данных, выполнить программу, заданную параметром ***file***, и отметить тот же файл как программу для выполнения при включении электропитания. **14** — Остановить выполняемую на текущий момент программу, удалить связанные с ней таблицы данных, выполнить программу, заданную параметром ***file***, не изменяя программу для выполнения при включении электропитания. **15** — Переместить файл, заданный параметром ***file2***, в имя, заданное параметром file. **16** — Переместить файл, заданный параметром ***file2***, в имя, заданное параметром ***file***, остановить выполняемую на текущий момент программу, удалить связанные с ней таблицы данных, выполнить программу, заданную параметром ***file2***, отметив ее как программу, подлежащую выполнению при включении электропитания. **17** — Переместить файл, заданный параметром ***file2***, в имя, заданное параметром ***file***, остановить выполняемую на текущий момент программу, удалить связанные с ней таблицы данных, выполнить программу, заданную параметром ***file2***, не изменяя программу, подлежащую выполнению при включении электропитания. **18** — Скопировать файл, заданный параметром ***file2***, в имя, заданное параметром ***file***. **19** — Скопировать файл, заданный параметром ***file2***, в имя, заданное параметром ***file***, остановить выполняемую на текущий момент программу, удалить связанные с ней таблицы данных, выполнить программу, заданную параметром ***file2***, отметив ее как программу, подлежащую выполнению при включении электропитания. **20** — Скопировать файл, заданный параметром ***file2***, в имя, заданное параметром ***file***, остановить выполняемую на текущий момент программу, удалить связанные с ней таблицы данных, выполнить программу, заданную параметром ***file2***, не изменяя программу, подлежащую выполнению при включении электропитания.
file	Задает первый параметр для операции управления файлом. Данный параметр должен быть задан для значений ***action1, 2, 3, 4, 5, 6,10,13,14, 15, 16, 17, 18, 19*** и ***20***.
file2	Задает второй параметр для операции управления файлом. Данный параметр должен быть задан для значений ***action15, 16, 17, 18, 19*** и ***20***.
format	Задает формат ответа. Значения **html, json** и **xml** являются действительными. Если данный параметр опущен или если в качестве значения используется **html**, пустое или недействительное значение, ответом является HTML.

Пример:

```
http://192.168.24.106/?command=FileControl&file=USR:APITest.dat&
action=4
```

Ответ: APITest.dat удаляется с диска USR: регистратора CR1000.

```
http://192.168.24.106/?command=FileControl&file=CPU:IndianaJones
_090712_2.CR1&action=1
```

Ответ: установить для программного файла значение Run Now (Выполнить сейчас).

```
http://192.168.24.106/?command=FileControl&file=USR:FileCopy.dat
&file2=USR:FileName.dat&action=18
```

Ответ: скопировать из file2 в file.

Ответ FileControl

Все форматы вывода содержат следующие параметры. Для любого ***action*** (действие) (например, **9**), осуществляющего сброс, ответ возвращается до завершения операций команды.

Таблица 121. Ответ на команду FileControl API

`outcome`	Нулевое ответное значение указывает на успешное выполнение. Ненулевое значение указывает на ошибку.
`holdoff`	Задает количество секунд до того, как веб-клиент предпримет следующую попытку обмена данными со станцией. Нулевое значение указывает на то, что обмен данными можно возобновить немедленно. Данные параметр необходим ввиду того, что многие команды направят распоряжение на перезапуск регистратора CR1000. В случае отправки операционной системы регистратору данных могут потребоваться десятки секунд для копирования образа из памяти во флеш-память и выполнения проверки, необходимой для загрузки новой операционной системы. Во время перезапуска регистратор CR1000 будет недоступен.
`description`	Подробные сведения по результирующему коду.

Пример:

```
192.168.24.106/?command=FileControl&action=4&file=cpu:davetest.c
r1
```

Ответ: удалить файл davetest.cr1 с диска CPU: узлового CR1000.

Если значение ***html*** вводится для параметра *format* команды **FileControl**, ответом будет HTML. Ниже приводятся пример ответа.

Ответ FileControl

outcome	0
holdoff	0
description	Файл удален

8.6.3.14.11. Управление файлами — команда ListFiles

С помощью команды **ListFiles** веб-клиент может получить список каталогов и файлов в узловом CR1000. Команда **ListFiles** имеет следующий вид:

```
http://ip_address/drive/?command=ListFiles
```

Команде **ListFiles** требуется уровень доступа к файлу **.csipasswd** не ниже **3** (только чтение).

Таблица 122. Параметры API команды ListFiles	
`format`	Задает формат ответа. Значения **html, json** и **xml** являются действительными. Если данный параметр опущен или если в качестве значения используется **html**, пустое или недействительное значение, ответом является HTML.
`uri`	Если данный параметр исключен или если ему присвоено значение «datalogger» (**uri=dl**) имеет пустую строку (**uri=**), то файловая система будет отправлена с узлового CR1000.[1]
[1] Дополнительно задает URI для станции регистратора данных *LoggerNet*, с которой будет получен список файлов.	

Например:

```
http://192.168.24.106/?command=ListFiles
```

Ответ: выводит структуру дисков узлового CR1000 (CPU:, USR:, CRD: и USB:).

```
http://192.168.24.106/CPU/?command=ListFiles
```

Ответ: выводит список файлов на диске CPU: узлового CR1000.

Ответ ListFiles

Формат ответа зависит от значения параметра *format* запроса команды. Ответ содержит информацию по каждому из файлов или каталогов, к которым можно получить доступ через веб-сервер CR1000. Информация по каждому файлу включает следующую:

Таблица 123. Ответ на команду ListFiles API	
`path`	Задает путь к файлу относительно пути URL.
`is_dir`	Если данное булево значение имеет истинное значение, это указывает, что объектом является каталог.
`size`	Целочисленная переменная, указывающая размер файла в байтах (is dir имеет значение false) или количество свободных байтов для каталога.
`last_write`	Строка, связанная только с теми файлами, которые определяют дату и время, когда файл был записан в последний раз.
`run_now`	Логический атрибут, применяемый регистратором CR1000 к программным файлам, которые отмечены как исполняемые на текущий момент.
`run_on_power_up`	Логический атрибут, применяемый регистратором CR1000 к программным файлам, которые отмечены для выполнения при включении электропитания или перезапуске CR1000.
`read_only`	Логический атрибут, применяемый регистратором CR1000 к файлу, который отмечен только для чтения.
`paused`	Логический атрибут, применяемый регистратором CR1000 к файлу, который отмечен для выполнения, но выполнение программы на текущий приостановлено.

Ответ HTML

Если значение ***html*** вводится для параметра *format* команды **ListFiles**, ответом будет HTML. Ниже приводятся примеры ответов.

Ответ в виде таблицы HTML:

Ответ ListFiles

Path	Is Directory	Size	Last Write	Run Now (Выполнить сейчас)	Run On Power Up	Read Only	Paused
ЦП/	true	443904	2012-06-22T000000	false	false	false	false
CPU/ModbusMasterTCPExample.CR1	false	967	2012-07-10T18:21:44	false	false	false	false
CPU/CS475-Test.CR1	false	828	2012-07-16T14:16:50	false	false	false	false
CPU/DoubleModbus SlaveTCP.CR1	false	1174	2012-07-31T171800	false	false	false	false
CPU/untitled.CR1	false	1097	2012-08-07T104820	false	false	false	false

Источник HTML-страницы:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN”
"http://www.w3.org/TR/htm14/loose.dtd">
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN"><html>
<head><title>ListFiles Response</title></head>
<body><h1>ListFiles Response</h1><table border="1">
<tr><td><b>Path</b></td>
<td><b>Is Directory</b></td>
<td><b>Size</b></td>
<td><b>Last Write</b></td>
<td><b>Run Now</b></td>
<td><b>Run On Power Up</b></td>
<td><b>Read Only</b></td>
<td><b>Paused</b></td></tr><tr>
<td>CPU/</td>
<td>true</td>
<td>443904</td>
<td>2012-06-22T00: 00:00</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td></tr><tr>
<td>CPU/ModbusMasterTCPExample.CR1</td>
<td>false</td>
<td>967</td>
<td>2012-07-10T18:21:44</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td></tr><tr>
<td>CPU/CS475-Test.CR1</td>
<td>false</td>
<td>828</td><td>2012-07-16T14:16:50</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td></tr><tr>
<td>CPU/DoubleModbusSlaveTCP.CR1</td>
<td>false</td>
<td>1174</td>
<td>2012-07-31T17:18:00</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td></tr><tr>
```

```
<td>CPU/untitled.CR1</td>
<td>false</td>
<td>1097</td>
<td>2012-08-07T10:48:20</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td></tr><tr>
</table>
```

Шаблон источника страницы:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html><head>
<title>ListFiles Response</title>
</head>
<body>
<h1>ListFiles Response</h1>
<table border="1">
   <tr>
     <td><b>Path</b></td>
     <td><b>Is Directory</b></td>
     <td><b>Size</b></td>
     <td><b>Last Write</b></td>
     <td><b>Run Now</b></td>
     <td><b>Run On Power Up</b></td>
     <td><b>Read Only</b></td>
     <td><b>Paused</b></td>
   </tr>
   <tr>
     <td>CPU:</td>
     <td>true</td>
     <td>50000</td>
     <td>YYYY-mm-dd hh:mm:ss.xxx</td>
     <td>false</td>
     <td>false</td>
     <td>false</td>
     <td>false</td>
   </tr>
   <tr>
     <td>CPU:lights-web.cr1</td>
     <td>false</td>
     <td>16994</td>
     <td>YYYY-mm-dd hh:mm:ss.xxx</td>
     <td>true</td>
     <td>true</td>
     <td>false</td>
     <td>false</td>
   </tr>
</table>
```

Ответ XML

Если значение ***xml*** вводится для параметра *format* команды **ListFiles**, ответ будет отформатирован как *CSIXML* *(стр. 90)* с присвоением имени корневого элемента **ListFilesResponse**. Ниже приводится пример ответа.

```
<ListFilesResponse>
   <file
     is_dir="true"
```

```
    path=”CPU:”
    size="50000"
    last_write="yyyy-mm-ddThh:mm:ss. xxx"
    run_now="false"
    run_on_power_up="false"
    read_only="false"
    paused="false" />
  <file
    is_dir="false"
    path="CPU:lights-web.cr1"
    last_write="yyyy-mm-ddThh:mm:ss.xxx"
    size="16994"
    run_now="true"
    run_on_power_up="true"
    read_only="false"
    paused="false"/>
</ListFilesResponse>
```

Ответ JSON

Если значение ***json*** вводится для параметра *format* команды **ListFiles**, ответ будет отформатирован как *CSIJSON* *(стр. 90)*. Ниже приводится пример ответа.

```
{
 "files": [
   {
    "path": "CPU:",
    "is_dir": true,
    "size": 50000,
    "last_write": "yyyy-mm-ddThh:mm:ss.xxx",
    "run_now": false,
    "run_on_power_up": false,
    "read_only": false,
    "paused": false
   },
   {
    "path": "CPU:lights-web.cr1",
    "is_dir": false,
    "size": 16994,
    "last_write": "yyyy-mm-ddThh:mm:ss. xxx",
    "run_now": true,
    "run_on_power_up": true,
    "read_only": false,
    "paused": false
   },
 ]
}
```

8.6.3.14.12. Управление файлами — команда NewestFile

С помощью команды **NewestFile** веб-клиент может запрашивать файл, например программу или изображение, у узлового CR1000. Если выражение содержит подстановочный знак (*), возвращается новейший файл из набора, имя которого соответствует выражению. Например, веб-страница может быть предназначена для отображения новейшего изображения, снятого камерой, подключенной к CR1000. Команда **NewestFile** имеет следующий вид:

```
http://192.168.13.154/?command=NewestFile&expr=drive:filename.ex
t
```

Где **filename** может быть выражено подстановочным знаком (*).

Команде **NewestFile** требуется уровень доступа к файлу **.csipasswd** не ниже **3** (только чтение) для всех файлов, кроме программных. Для доступа к программным файлам требуется уровень доступа не ниже **1** (разрешен полный доступ).

Таблица 124. Параметры API команды NewestFile	
`expr`	Задает полный путь и выражение с подстановочными знаками для заданного набора файлов[1]. ***expr=USR: *jpg*** выбирает новейший из набора файлов на диске USR:, который имеет расширение .jpg.
[1] Веб-сервер на ПК ограничивает пути на узловом компьютере, кроме разрешенных в применимом файле конфигурации сайта (.sources.xml), чтобы не предотвратить веб-доступ ко всем файловым системам, доступным узловому компьютеру.	

Пример:

```
http://192.168.24.106/?command=NewestFile&expr=USR:*.jpg
```

Ответ: веб-сервер получает новейший JPG-файл на диске USR: узлового CR1000.

Примечание. Для получения какого-либо файла, независимо от времени его создания, используется url-ссылка http://ip_address/drive/filename.ext. Имя необходимого файла определяется с помощью команды **ListFiles**.

Ответ на команду NewestFile

Веб-сервер передает содержимое новейшего файла, соответствующего выражению, представленному в ***expr***. Если файлы, соответствующие запросу, отсутствуют, сервер выдает ошибку код ответа HTTP **404 Not Found.**

8.7. ПО поддержки регистратора данных — подробная информация

Подробнее:
• *ПО поддержки регистратора данных — краткое руководство (стр. 46)*
• *ПО поддержки регистратора данных — общие сведения (стр. 95)*
• *ПО поддержки регистратора данных — подробная информация (стр. 450)*
• *ПО поддержки регистратора данных — списки (стр. 654)*

ПО поддержки регистратора данных обеспечивает генерирование и редактирование программ, вывод данных и мониторинг данных в реальном масштабе времени.

- Бесплатное приложение PC200WStarter можно скачать на сайте *www.campbellsci.com/downloads* (*http://www.campbellsci.com/downloads*). Оно поддерживает прямое соединение RS-232 между ПК и CR1000 и включает программный модуль *Short Cut* для создания программ CR1000. Кроме того, оно имеет в своем составе средства настройки часов регистратора данных, отправки программ, мониторинга датчиков и просмотра и сбора данных по месту эксплуатации.
- *Приложения LoggerLinkMobile* являются простыми, но многофункциональными средствами, обеспечивающими возможность обмена данными между устройствами на базе операционных систем iOS или Android с регистраторами CR1000 с поддержкой протокола IP. Приложения позволяют выполнять задачи технического обслуживания в полевых условиях, такие как просмотр и сбор данных, настройка часов и загрузка программ.
- *ПО поддержки регистратора данных PC400* поддерживает множество способов обмена информацией, ручной сбор данных и индикаторов текущего контроля. Оно включает приложения *Short Cut* и *CRBasic Editor* для создания программ для регистраторов CR1000. *PC400* не поддерживает сложные способы обмена данными, такие как связь между телефонной линией и беспроводной сетью, маршрутизация PakBus® или запланированный сбор данных.

- *ПО поддержки регистратора данных LoggerNet* обеспечивает поддержку комбинированных вариантов связи, пользовательских индикаторов текущего контроля и запланированный сбор данных. Оно включает приложения *Short Cut* и *CRBasic Editor* для создания программ для регистраторов CR1000. Оно также включает средства для настройки конфигурации, диагностики и администрирования сетей регистраторов данных. Для более требовательных областей применения предусматриваются приложения *LoggerNet Admin* и *LoggerNet Remote*.
- *LNLINUX — сервер LoggerNet на базе ОС Linux* с приложением *LoggerNet Remote* представляет собой решение для тех пользователей, которым требуется работа сервера *LoggerNet* в среде Linux. Программный комплекс включает Linux-версию сервера *LoggerNet* и Windows-версию приложения *LoggerNet Remote*. Клиентские приложения на базе Windows в *LoggerNet Remote* выполняются на отдельном компьютере и используются для управления сервером *LoggerNet* на базе Linux.
- *VISUALWEATHER — ПО для метеостанций* поддерживает метеостанции производства Campbell Scientific. В версии 3.0 или выше поддерживаются пользовательские метеостанции или предварительно настроенные станции ET107, ET106 и MetDatal. Программное обеспечение позволяет инициализировать настройку, выполнить опрос станции, а также формировать отчет по данным одной или нескольких метеостанций.

Примечание. Подробную информацию о ПО Campbell Scientific можно найти на сайте *www.campbellsci.com http://www.campbellsci.com*. Для получения рекомендации по выбору ПО, подходящего для вашей сферы деятельности, обратитесь к техническому специалисту Campbell Scientific.

8.8. Экранная клавиатура — подробная информация

Связанные темы:
• *Экранная клавиатура — общие сведения* *(стр. 83)*
• *Экранная клавиатура — подробная информация* *(стр. 451)*
• *Экранная клавиатура — список* *(стр. 651)*
• *Пользовательские меню — общие сведения* *(стр. 84, стр. 581)*

Читать далее. См. *Пользовательские меню* *(стр. 182)*.

Клавиатура доступна для применения с регистраторами CR1000.
См. приложение *Экранные клавиатуры* *(стр. 651)* для получения более подробной информации по доступным экранным клавиатурам. В данном разделе описывается порядок использования экранной клавиатуры с помощью стандартных меню. Некоторые клавиши обладают специальными функциями, которые рассматриваются ниже.

Примечание. Несмотря на то, что экранная клавиатура не требуется для управления регистратором CR1000, она является полезным инструментом диагностики и отладки.

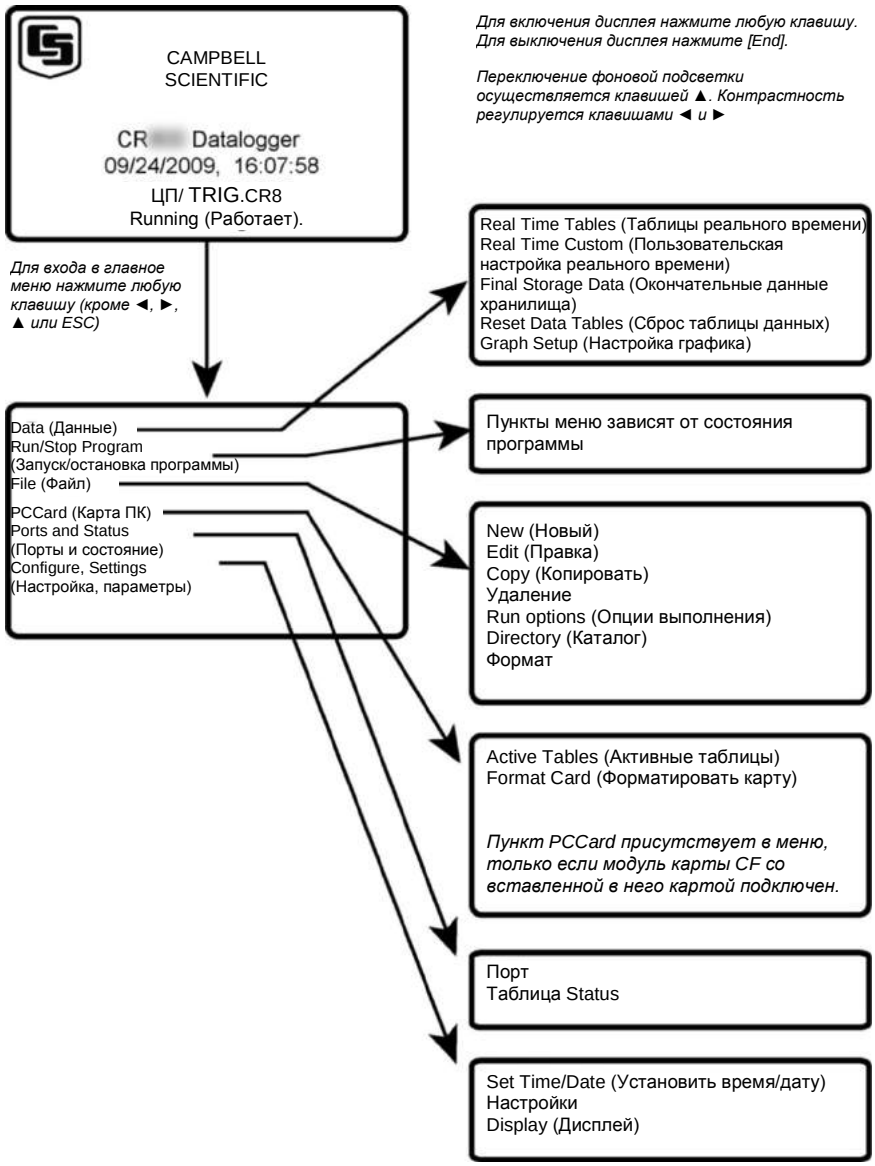
Таблица 125. Специальные функции клавиш электронной клавиатуры

Клавиша	*Специальная функция*
[2] и **[8]**	Переход на одну строку вверх или вниз по списку при каждом нажатии
[Enter]	Выбор строки или переключение опции в строке, на которой находится курсор
[Esc]	Возврат на один уровень меню вверх
[Home]	Перенос курсора на верхнюю строку в списке

Таблица 125. Специальные функции клавиш электронной клавиатуры

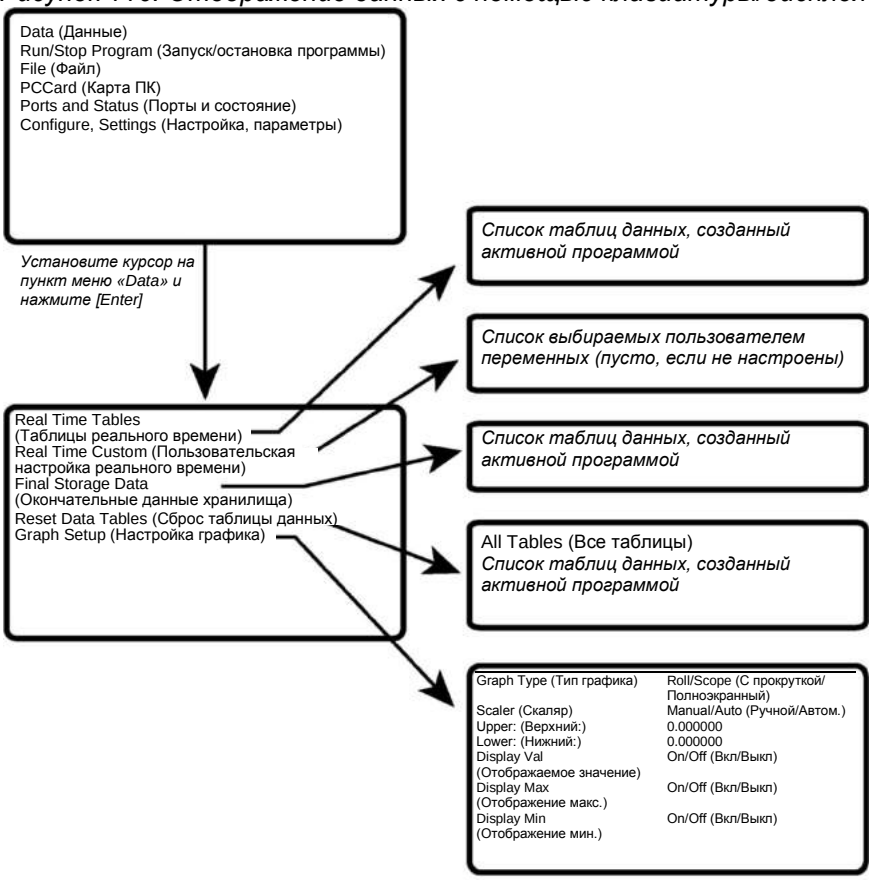
Клавиша	*Специальная функция*
[End]	Перенос курсора на нижнюю строку в списке
[Pg Up]	Перенос курсора на один экран вверх
[Pg Dn]	Перенос курсора на один экран вниз
[BkSpc]	Удаление символа слева
[Shift]	Изменение регистра текстового символа
[Num Lock]	Переход к цифровой клавиатуре
[Del]	Удаление
[Ins]	Вставка/изменение настройки графика
[Graph]	График

Рисунок 115. Использование клавиатуры/дисплея



8.8.1. Дисплей данных

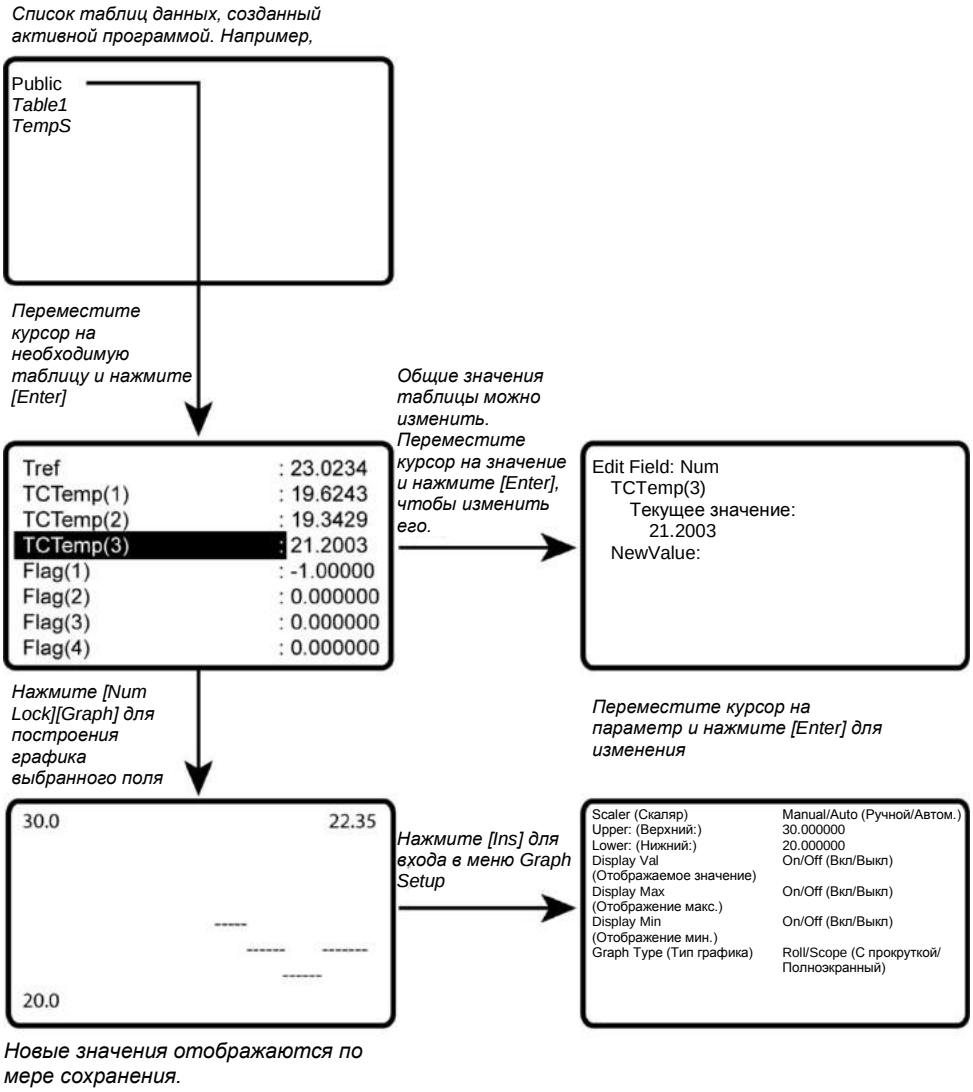
Рисунок 116. Отображение данных с помощью клавиатуры/дисплея



Для значения Scope требуется, чтобы был выбран ручной скаляр

8.8.1.1. Таблицы и графики в реальном масштабе времени

Рисунок 117. Таблицы и графики в реальном масштабе времени

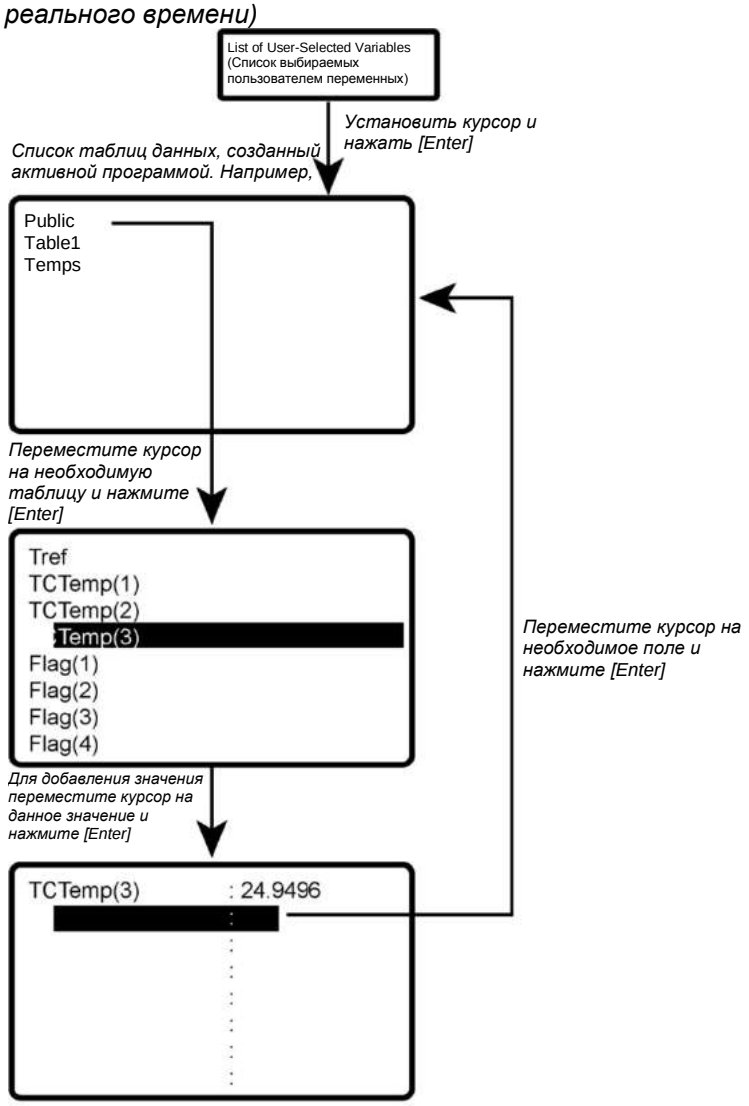


8.8.1.2. Real-Time Custom (Пользовательская настройка реального времени)

Экранную клавиатуру CR1000KD можно настроить на отображение пользовательской настройки реального времени. Регистратор CR1000 будет хранить настройку до тех пор, пока выполняется определяющая программа.

Читать далее. Также можно запрограммировать пользовательские меню. См. раздел *Пользовательские меню (стр. 182)* для получения более подробной информации.

Рисунок 118. Real Time Custom (Пользовательская настройка реального времени)

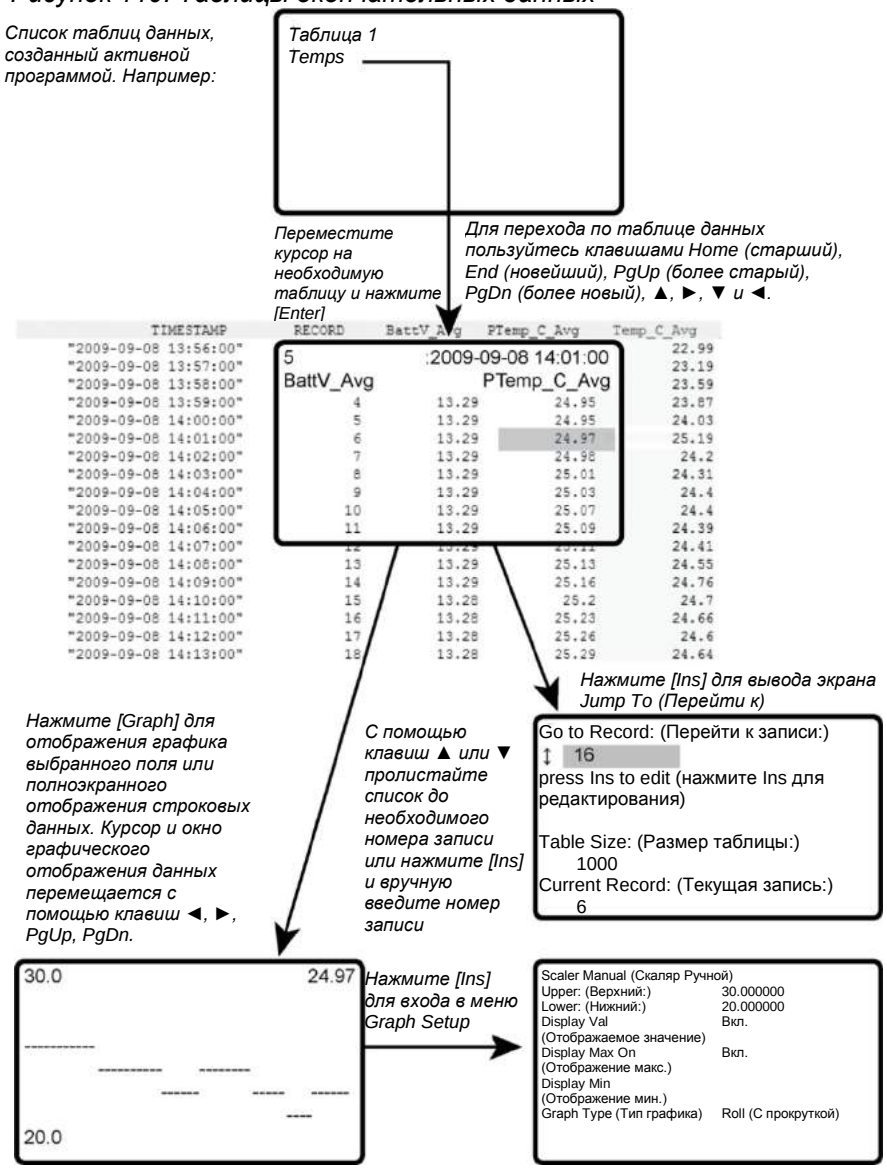


Новые значения отображаются по мере сохранения.

Для удаления поля переместите курсор на это поле и нажмите [DEL]

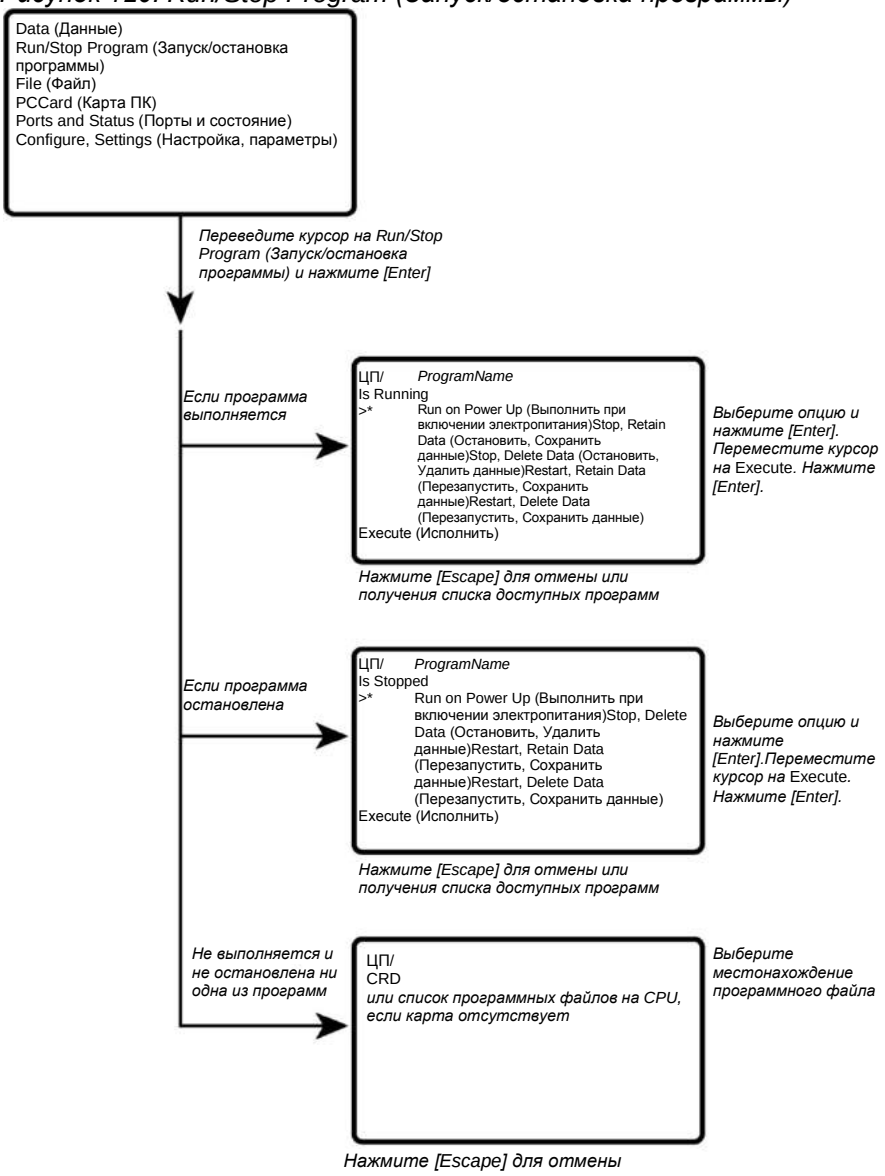
8.8.1.3. Таблицы окончательных данных

Рисунок 119. Таблицы окончательных данных



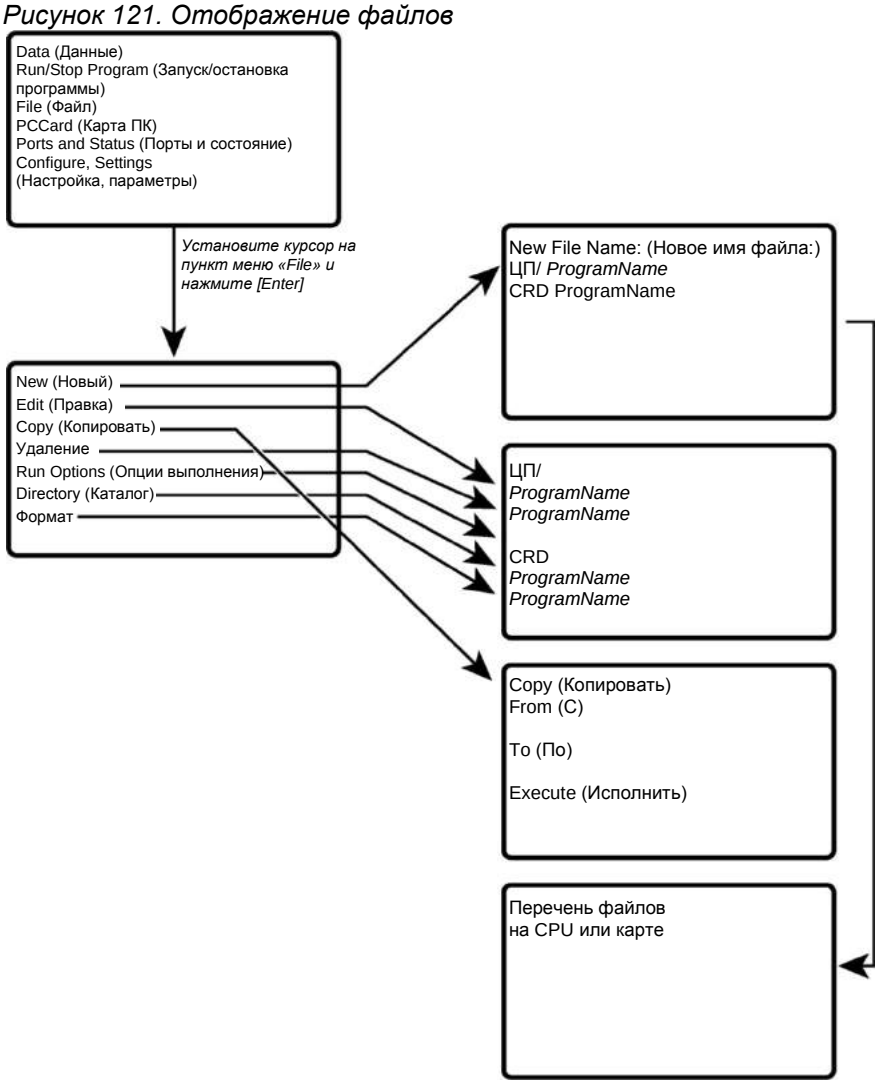
8.8.2. Run/Stop Program (Запуск/остановка программы)

Рисунок 120. Run/Stop Program (Запуск/остановка программы)



8.8.3. Отображение файлов

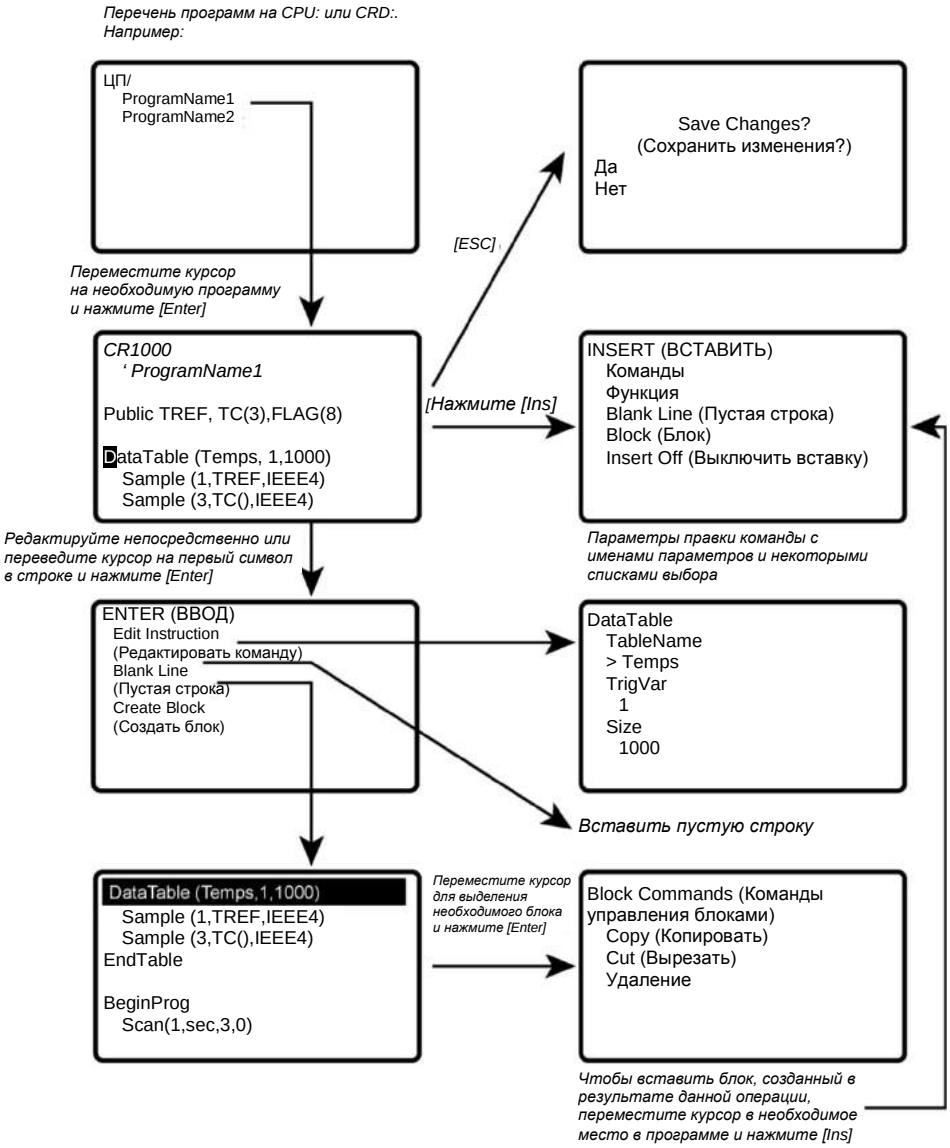
Рисунок 121. Отображение файлов



8.8.3.1. Файл: Edit (Файл: Правка)

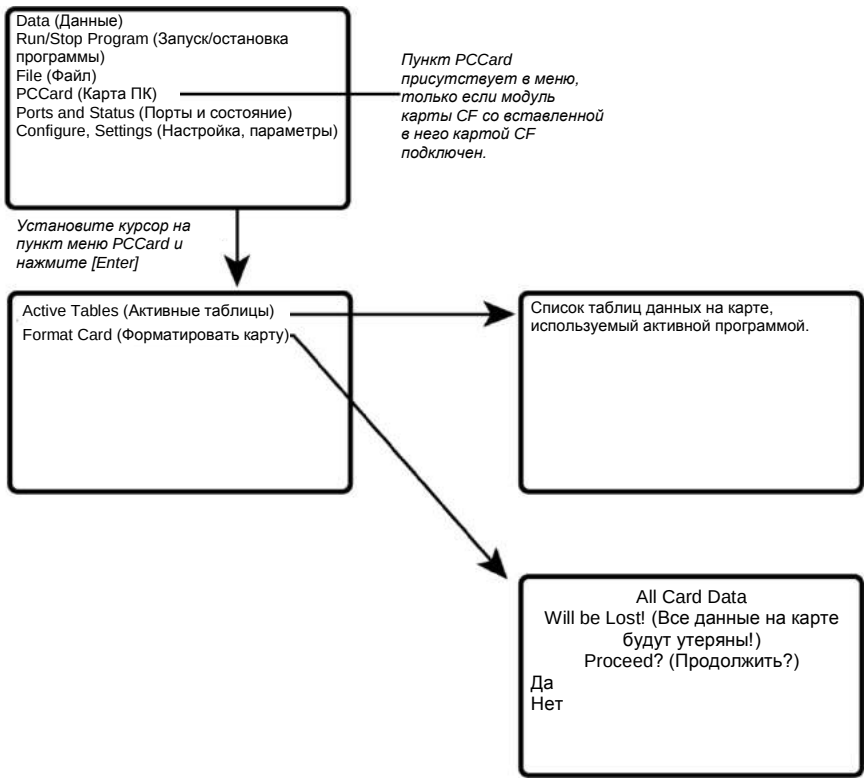
Для записи и редактирования программ регистратора данных рекомендуется использовать *CRBasic Editor*. При внесении незначительных изменений с помощью экранной клавиатуры CR1000KD перезапустите программу для активации изменений, но учтите, что все переменные и другие значение будут сброшены, если не запрограммировано какое-либо другое действие. Учитывайте, что единственная копия изменений хранится в CR1000, до тех пор пока программа не будет выведена при помощи ПО поддержки регистратора данных или съемной памяти.

Рисунок 122. Файл: Edit (Файл: Правка)



8.8.4. Отображение PCCard (карт памяти)

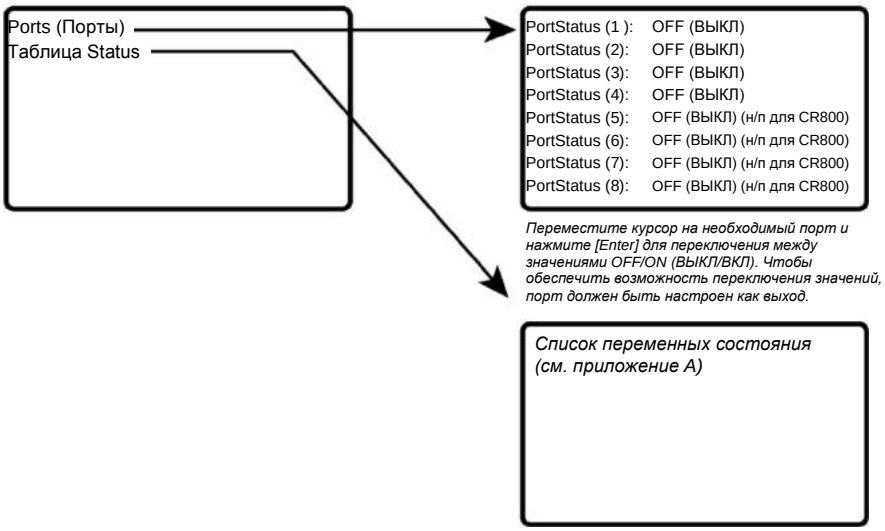
Рисунок 123. Отображение PCCard (карт CF)



8.8.5. Порты и состояние

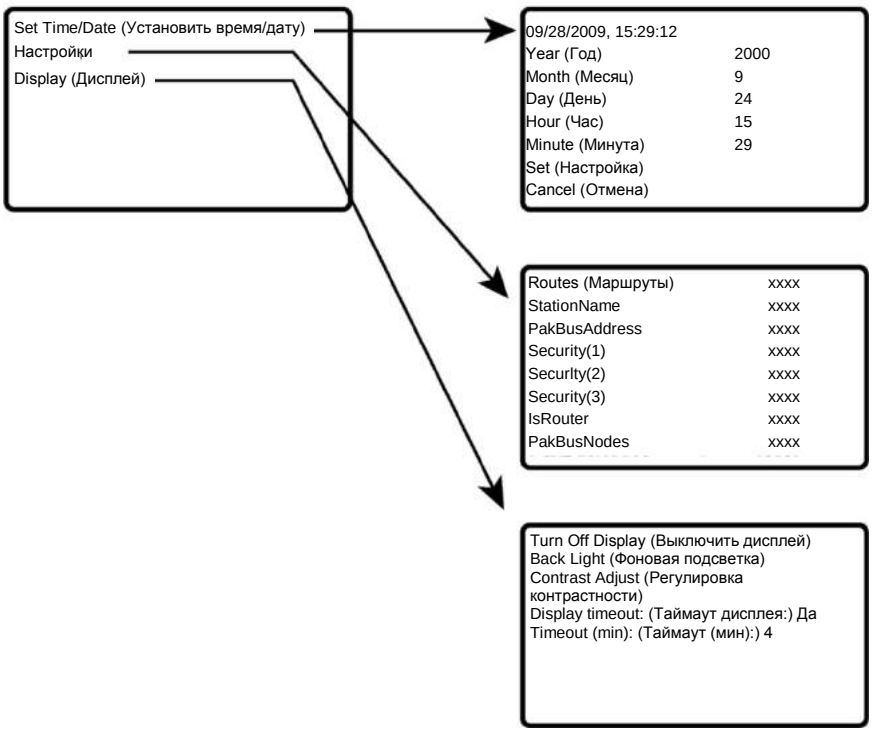
Читать далее. См. приложение «Регистры».

*Рисунок 124. Состояние клемм (портов) **C***



8.8.6. Настройки

Рисунок 125. Настройки



8.8.6.1. Настройка времени/даты

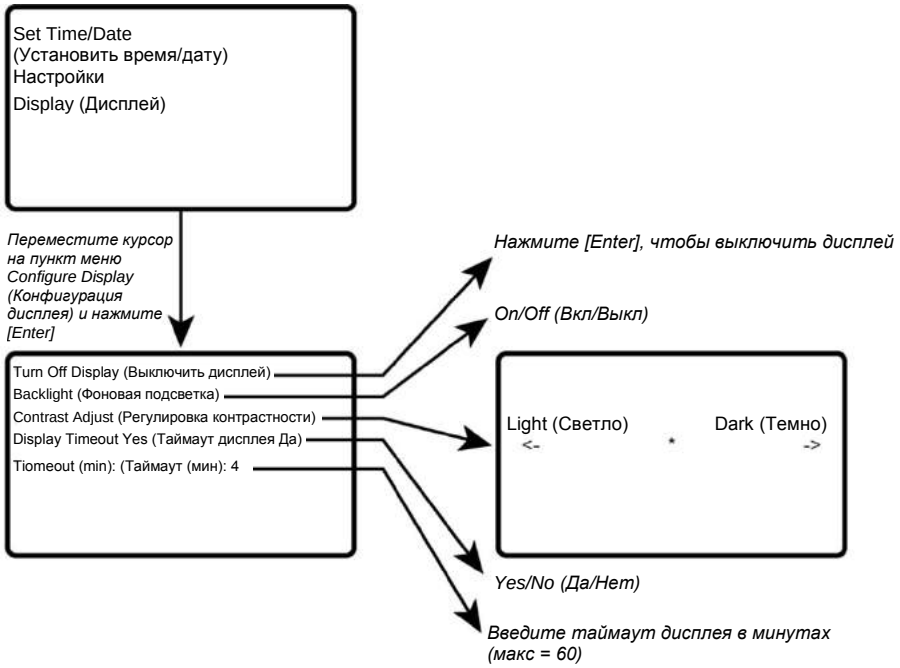
Переместите курсор на элемент времени и нажмите **Enter** для изменения. Затем переместите курсор на **Set** и нажмите **Enter**, чтобы применить изменения.

8.8.6.2. Настройки PakBus

В меню **Settings** переместите курсор на пункт PakBus® и нажмите **Enter** для изменения. После изменения нажмите клавишу **Enter**, чтобы применить изменения.

8.8.7. Конфигурация дисплея

Рисунок 126. Конфигурация дисплея



8.9. Сжатие программных файлов и файлов ОС — вопрос-ответ

В: Что такое Gzip?

А: Gzip — это формат GNU файла zip-архива. Данный формат файла и алгоритмы, используемые для его создания, имеют открытый исходный код и бесплатны для использования в любых целях. Файлы с расширением .gz прошли алгоритм сжатия данных для уменьшения размера. Для получения более подробной информации посетите сайт www.gnu.org.

В: Есть ли разница между форматами Gzip и zip?

А: Несмотря на все подобие, Gzip и zip используют различные форматы и алгоритмы сжатия файлов. С регистратором CR1000 совместимы программные файлы и файлы ОС, сжатые при помощи только Gzip.

В: Зачем сжимать программу или операционную систему перед отправкой в регистратор данных CR1000?

A: Благодаря сжатию файла есть вероятность существенного сокращения его размера. Фактическая степень сжатия зависит, в основном, от количества и близости резервных информационных блоков в файле. Сокращение размера файла означает сокращение числа байтов, которые необходимо передать при отправке файла в регистратор данных. За счет сжатия существенно снижается длительность передачи данных по низкоскоростным или каналам связи с большой задержкой, а также сокращается плата за пользование линиями, если используются тарифы за переданный трафик. Сжатие представляет особое преимущество при передаче программ или ОС по низкоскоростным наземным радиоканалам, спутниковым каналам или каналам мобильной связи с ограниченной пропускной способностью.

B: Поддерживает ли регистратор CR1000 формат Gzip?

A: Версия 25 стандартной операционной системы CR1000 поддерживает прием сжатых файлов и ОС в формате Gzip.

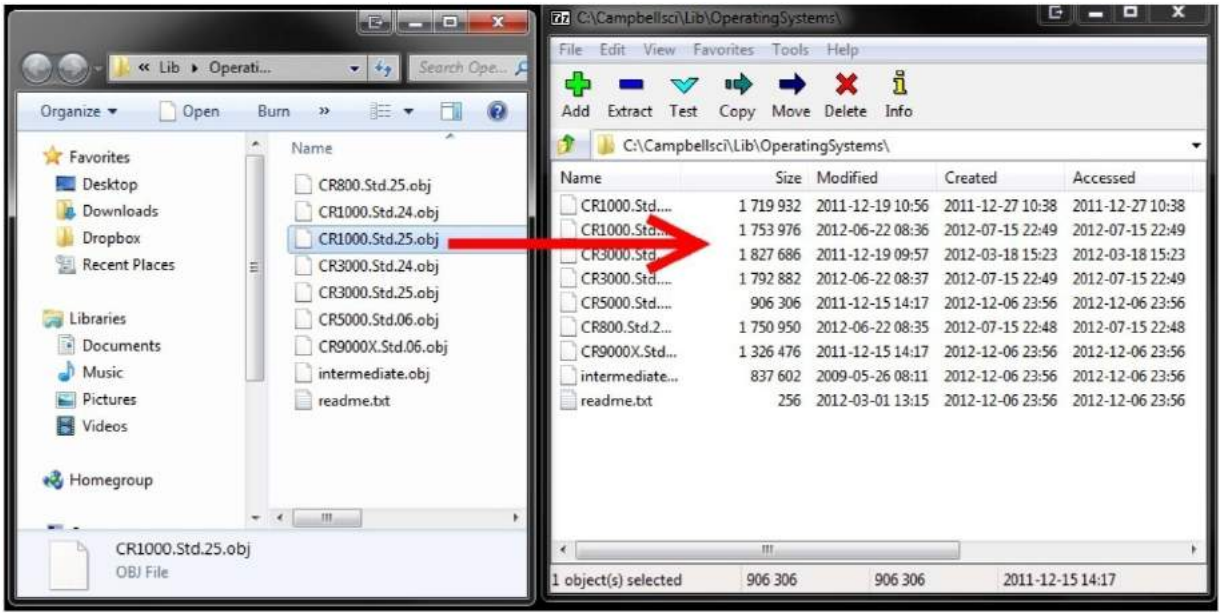
B: Как сжать программу или операционную систему в формате Gzip?

A: Существует множество служебных программ, которые могут создать Gzip-файл. В данном документе рассматривается применение диспетчера файлов *7-Zip*. *7-Zip* является бесплатной служебной программой с открытым исходным кодом, совместимой с ОС *Windows*®. Скачать и получить инструкции по установке можно на сайте http://www.7-zip.org/. После установки *7-Zip* создание файла Gzip осуществляется в четыре этапа:

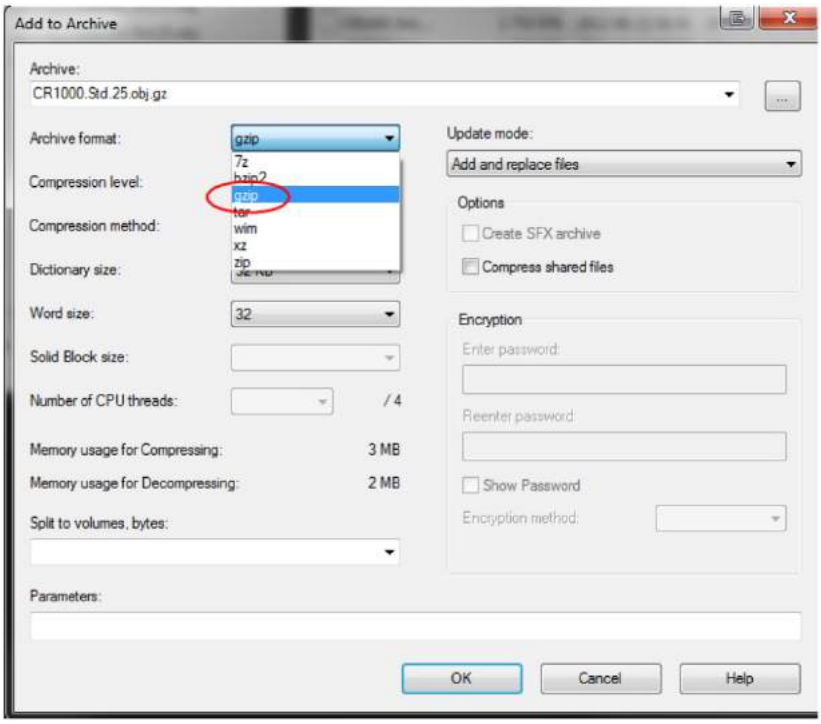
a) Откройте *7-Zip*.

b) Перетащите программу или операционную систему для сжатия в открытое окно.

c) Когда появится запрос, выберите формат архива «Gzip».



c) Когда появится запрос, выберите формат архива «Gzip».



d) Нажмите ***OK***.

Результирующие имена файлов будут иметь вид «myProgram.cr1.gz» и «CR1000.Std.25.obj.gz». Следует учитывать, что имена файлов заканчиваются расширением «.gz». Перед расширением «.gz» должно указываться расширение исходного файла (.cr1, .obj), как показано на рисунке.

B: Как отправить сжатый файл в CR1000?

A: Сжатый файл Gzip можно отправить в регистратор данных CR1000 нажатием на команду **Send Program (Отправить программу)** в *ПО поддержки регистратора данных (стр. 95)*. Сжатые программы также можно отправить с помощью команды **HTTP PUT** на веб-сервер CR1000. Регистратор CR1000 автоматически не разархивирует и не использует сжатые файлы, отправленные с помощью **File Control (Управление файлами)**, FTP или низкоуровневой загрузки ОС. Однако данные файлы можно вручную разархивировать, отметив для них атрибут **Run Now (Выполнить сейчас)** с помощью **File Control (Управление файлами), FileManage()** и HTTP.

Примечание. Зашифрованные программы сжимаются лишь с незначительной степенью (см. раздел **FileEncrypt()** в *Справке по редактору CRBasic*), поскольку в процессе шифрования не формируется большое число повторяемых последовательностей байтов. Степень Gzip-сжатия файлов, которые были созданы с применением алгоритмов сжатия, такие как JPEG или MPEG-4, также незначительна.

Таблица 126. Типовые результаты сжатия файлов в формате Gzip

File (Файл)	*Размер исходного файла в байта*	*Размер сжатого файла в байта*
Операционная система CR1000	1,753,976	671,626
Небольшая программа	2,600	1,113
Большая программа	32,157	7,085

8.10. Карты памяти и номера записей

Связанные темы:
• *Карта памяти (CRD: диск) — общие сведения* *(стр. 89)*
• *Карта памяти (CRD: диск) — подробная информация* *(стр. 376)*
• *Карты памяти и номера записей* *(стр. 466)*
• *Выходные данные: запись высокочастотные данные на карты памяти* *(стр. 205)*
• *Ошибки файловой системы* *(стр. 389)*
• *Устройства хранения данных — список* *(стр. 653)*
• *Примеры форматов файлов данных* *(стр. 379)*
• *Таблица устройств для хранения данных* *(стр. 373)*

Номера записей в таблице данных, если в объявлении таблицы данных используется **CardOut()** или **TableFile()** с ***Option 64***, подчиняются следующим правилам:

1. На картах памяти (диск CRD:) и во внутренней памяти (CPU) хранятся копии таблиц данных в двоичном формате TOB3. Номера записей, собираемые с дисков CRD: и CPU, содержатся в записях **DataRecordSize** в таблице **Status (Состояние)**.
2. В определениях таблицы, сообщаемых *ПО поддержки регистратора данных* *(стр. 95)*, регистратор CR1000 сообщает большее число номеров записей, зарегистрированных в таблице **Status (Состояние)**, если таблицы не являются таблицами с ограниченным заполнением.
3. Если какая-либо область данных имеет признак ограниченного заполнения, то та область, которая первая достигает предела заполнения, останавливает все хранилища окончательных данных, даже если в неограниченной области имеется доступное пространство. За счет этого количество записей ограничивается до минимального значения одной из двух областей, если обе эти области имеют ограничение по заполнению.
4. Если **CardOut()** или **TableFile()** с ***Option 64*** присутствует, независимо от того, вставлена карта или нет, пространство под таблицу данных CPU выделяется в размере не менее 5 кБ, поэтому, по крайней мере, имеется минимальное буферное пространство для хранения данных на диске CRD: (данная операция выполняется в фоновом режиме, когда у регистратора CR1000 появляется возможность скопировать данные на карту). Поэтому, к примеру, для таблицы данных, состоящей из одного четырехбайтного образца, без активации по интервалам, 20 байт на запись, включая заголовок/колонтитул TOB3 16 байт, выделяется 258 записей во внутренней памяти для любой программы, определяющей менее 258 записей (опять же только в том случае, если **CardOut()** или **TableFile()** с ***Option 64*** присутствует). Программы, в которых задано более чем 258 записей, выводят информацию, заданную пользователем, без ограничения минимального числа записей.
5. Если команда **CardOut()** или **TableFile()** с ***Option 64*** используется при отсутствующей карте, в таблице **Status (Состояние)** указывает нуль байт.
6. В пространстве таблиц данных внутренней памяти и карты памяти выделяется около 2 кБ дополнительного пространства (около 100 дополнительных записей в вышеприведенном примере), для того чтобы свести к минимуму для кольцевой памяти возможность перезаписи старых данными новыми, когда *ПО поддержки регистратора данных* *(стр. 95)* предпримет попытку одновременного сбора старых данных. Эти дополнительные записи не выводятся в таблице **Status (Состояние)** и не направляются в ПО поддержки регистратора данных и, соответственно, не могут быть собраны.
7. Если команда **CardOut()** или **TableFile()** с ***Option 64*** задана для ограниченного наполнения, все пространство, зарезервированное для записей на карте, регистрируется перед прекращением записи окончательных данных в память, включая дополнительные 2 кБ, выделенные для устранения конфликта, связанного с сохранением новейших данных во время чтения старейших, когда область памяти не

ограничена по заполнению или является кольцевой. По этой причине, если CPU не останавливается заранее или имеет кольцевой тип, а не ограниченный по заполнению, то на карте будет храниться большее число записей, чем изначально выделено, т. е. приблизительно на 2 кБ больше записей при условии отсутствия ошибок. В точке, в которой прекращается запись окончательных данных, регистратор CR1000 пересчитывает количество записей, отображает их в таблице **Status (Состояние)** и сообщает в ПО поддержки регистратора данных новое определение таблицы. Более того, если таблица фиксирует данные относительно быстро, могут иметь место дополнительные записи, которые уже хранились в буфере CPU до полного прекращения фиксации окончательных данных. Это приводит к тому, что может быть собрано несколько незаявленных записей. Например, на CR1000, на котором фиксируются четырехбайтные значения с интервалом 10 мс, CPU не настроен на ограниченное заполнение, CRD: настроен на ограниченное заполнение через 500 записей — после прекращения фиксации окончательных данных, на диске CRD: имелось 603 записи, представленные в таблице **Status (Состояние)** (дополнительные 103 ввиду дополнительных 2 кБ, выделенных под кольцевую буферизацию), однако собрано может быть 608 записей, поскольку CPU потребовалось 50 мс (или 5 записей), чтобы прекратить фиксацию 5 записей после того, как карта была остановлена.

8. Следует учитывать, что только диск CRD: продолжит фиксацию, до тех пор пока его записи не будут заполнены; диск CPU: прекратит фиксацию, когда будет зафиксировано запрограммированное число записей.
9. Следует учитывать, что команда **O** в режиме терминала позволяет визуализировать процессы, происходящие на дисках CPU: и CRD: — фактически выделенный размер, где они находятся в настоящий момент и т. п.

8.11. Безопасность — подробная информация

Связанные темы:
• *Безопасность — общие сведения (стр. 92)*
• *Безопасность — подробная информация (стр. 467)*

CR1000 поставляется без активных средств обеспечения безопасности. По умолчанию протоколы RS-232, Telnet, FTP и HTTP, которые обеспечивают высокий уровень доступа к данным CR1000, и программа CRBasic не защищены паролем.

Вы можете защитить CR1000 от ошибок или подделки данных. Защита может требоваться в следующих случаях:

- сбор конфиденциальных данных;
- использование критически важных систем;
- использование сетей, доступ к которые имеют многие пользователи.

При наличии угроз безопасности, в частности проникновения по протоколу TCP/IP, необходимо загрузить в CR1000 последнюю версию *операционной системы (стр. 86)*, отключить неиспользуемые протоколы и защитить те, которые используются. Для обеспечения безопасности можно предпринять следующие действия:

- установка блокировки кода доступа;
- установка пароля PakBus/TCP;
- задание имени пользователя и пароля для FTP;
- установка ключа шифрования PakBus AES-128;
- установка файла формата .csipasswd для защиты HTTP и веб-API;
- подписи записей;
- шифрование файлов программы, если они содержат конфиденциальную информацию;
- для усиления защиты файлы программы можно скрыть;
- установка CR1000 и источник питания в корпус, закрываемый на ключ.

Примечание. Все функции безопасности можно изменить, имея физический

доступ к CR1000. Если требуется абсолютная безопасность, CR1000 должен храниться в безопасном месте.

8.11.1. Уязвимости

И хотя в прошлом подход «безопасность через непонятность» мог обеспечивать достаточную степень безопасности, регистраторы данных Campbell Scientific все чаще применяются в областях, требующих повышенной безопасности. Для разработки мер противодействия злонамеренным действиям или непреднамеренной порче требуется понимание, в каких местах системы могут быть взломаны и как противостоять потенциальным угрозам.

Примечание. Старые операционные системы CR1000 более уязвимы для атак по сравнению с последними обновлениями. Обновления можно бесплатно скачать на сайте *www.campbellsci.com.*

Уязвимости представлены в следующих основных моментах:

- Экранная клавиатура CR1000KD
  - Если клавиша **Del** удерживается нажатой во время включения электропитания CR1000, это приводит к прекращению загрузки программы и предоставляет окно длительностью 120 секунд, чтобы начать изменение или отключить коды безопасности в редакторе настроек (а не в таблице **Status (Состояние)**) при помощи экранной клавиатуры.
  - Обход системы безопасности при помощи экранной клавиатуры не дает телекоммуникационный доступ без предварительной корректировки кода безопасности.
  - **Примечание.** Данные функции не работаю в клавиатурах CR1000KD с серийным номером меньше 1263. Свяжитесь с Campbell Scientific для получения информации по обновлению операционной системы CR1000KD.
- LoggerNet
  - Доступ ко всем функциям и данным регистратора данных можно легко получить через интерфейсы **RS-232** и Ethernet с помощью ПО поддержки регистратора данных Campbell Scientific.
  - Cora — команда **find-logger-security-code**
- Telnet;
  - Подробный просмотр IP-трафика. IP-трафик может выявить потенциальную конфиденциальную информацию, такую как имена пользователей и пароли для входа на FTP, а также информацию для подключения к серверу, включая IP-адреса и номера портов.
  - Просмотр трафика последовательной связи с другими регистраторами данных и устройствами. Например, измеритель уровня мощности с поддержкой протокола Modbus.
  - Просмотр данных в таблицах **Public** и **Status (Состояние).**
  - Просмотр программы регистратора данных, которая может содержать конфиденциальную информацию, являющуюся объект интеллектуальных прав, коды безопасности, имена пользователей, пароли, информацию о соединении и комментарии, в которых уточняются или выявляются коды.
- FTP
  - Отправка и изменение программ регистратора данных.
  - Отправка данных, которые были записаны в файл.
- HTTP
  - Отправка программ регистратора данных.
  - Просмотр данных таблиц.
  - Получение архивных записей или других файлов, содержащихся на дисках регистратора данных.

- Расширенные права доступа предоставляются, если присутствует файл .csipasswd, поэтому примите меры к тому, чтобы пользователи с правами администратора имели учетные данные с надлежащей степенью безопасности.

8.11.2. Блокировка посредством кодов доступа

Блокировка посредством кодов доступа (исторически именуемых в регистраторах данных Campbell Scientific просто «кодами безопасности») является старейшим способом обеспечения безопасности регистратора данных. Блокировка посредством кодов доступа может эффективно заблокировать непреднамеренную порчу или помешать неопытным хакерам, действующим по телекоммуникационным каналам связи, не использующим протокол IP. Однако любой серьезный хакер с физическим доступом к регистратору данных или телекоммуникационному оборудованию может с минимальным трудом обойти защиту посредством пятизначных кодов доступа. Системы с адекватной степенью безопасности на основе блокировки посредством кода доступа, вероятно, сводятся к следующим видам связи:

- частные сети с радиодоступом, не использующие протокол IP;
- прямые соединения (проводной RS-232, ближней связи, многоточечные, волоконно-оптические);
- спутниковая связь, не использующая протокол IP;
- наземная линия телефонной связи, не использующая протокол IP, в которой телефонный номер не публикуется;
- мобильный телефон, в котором отключен протокол IP, обеспечивающий исключительно последовательное соединение.

Можно установить до трех уровней блокировки. Действительными кодами безопасности являются числа от **1** до **65535** (**0** присваивается в случае отсутствия безопасности).

Примечание. Несмотря на то, что код доступа может иметь отрицательное значение, положительный код должен вводиться для разблокировки регистратора CR1000. Данный положительный код рассчитывается по следующей формуле: 65536 + (отрицательный код безопасности), т. е. код безопасности –1111 должен вводиться как 64425 для разблокировки CR1000.

Блокировка посредством кода доступа включается следующими способами:

- Таблица **Status (Состояние)** — регистры **Security(1), Security(2)** и **Security(3)** являются перезаписываемыми переменными в таблице **Status (Состояние)**, в которую записываются коды доступа для уровней доступа с 1 по 3, соответственно.
- Настройки экранной клавиатуры CR1000KD.
- *Служебная программа конфигурации устройства (DevConfig)* — пароли безопасности с 1 по 3 вводятся на вкладке **Deployment** (Развертывание).
- Команда **SetSecurity()** — **SetSecurity()** исполняется только при компиляции программы. Ее можно поместить между командами **BeginProg** и **Scan()**.

Примечание. Удаление команды **SetSecurity()** из программы CRBasic не равносильно записи **SetSecurity(0,0,0).** Настройки сохраняются, если загружается новая программа, в которой отсутствует команда **SetSecurity()**.

Level 1 (Уровень 1) должен быть настроен до настройки **Level 2 (Уровень 2).** **Level 2** (Уровень 2) должен быть настроен до настройки **Level 3** (Уровень 3). Если уровень установлен на 0, то любой следующий уровень будет установлен на 0. Например, если для уровня 2 установлено значение 0, то уровень 3 автоматически устанавливается на 0. Уровни разблокируются в обратном порядке: уровень 3 перед уровнем 2, уровень 2 перед уровнем 1. Если уровень разблокирован, любой следующий уровень также разблокируется, т. е. при разблокировке уровня 1 (вводится код безопасности для **Level 1**) уровни 2 и 3 также разблокируются.

Функции, затрагиваемые каждым уровнем безопасности:

- Уровень 1 — сбор данных, настройка часов и настройка переменных в таблице **Public** не ограничена и не требуют кода безопасности. Если код **Security1** введен, значения чтения/записи в таблице **Status (Состояние)** можно изменять, а также можно изменить или вывести программу регистратора данных.
- Уровень 2 — сбор данных не ограничен, код безопасности не требуется. Если пользователь вводит код **Security2**, можно изменять часы регистратора данных и переменные в таблице **Public**.
- Уровень 3 — когда задан данный уровень, все виды связи с регистратором данных запрещаются до тех пор, пока не будет введен код безопасности. Если введен код **Security3**, можно просматривать и получать данные регистратора данных (кроме данных, вывод которых запрещен командой **TableHide()** в программе CRBasic). Если введен код **Security2**, можно получать данные, настраивать общие переменные и настраивать часы. Если введен код **Security1**, разрешается полный доступ ко всем функциям.

8.11.2.1. Обход блокировки посредством кода доступа

Блокировку посредством кода доступа можно обойти при помощи экранной клавиатуры CR1000KD регистратора данных. Если клавиша **Del** удерживается нажатой во время включения электропитания CR1000, это приводит к прекращению загрузки программы и предоставляет окно длительностью 120 секунд, чтобы начать изменение или отключить коды безопасности в редакторе настроек (а не в таблице **Status (Состояние)**) при помощи экранной клавиатуры.

Обход системы безопасности при помощи экранной клавиатуры не дает телекоммуникационный доступ без предварительной корректировки кода безопасности.

Примечание. Данные функции не работают в клавиатурах CR1000KD с серийным номером меньше 1263. Свяжитесь с Campbell Scientific для получения информации по обновлению операционной системы CR1000KD.

8.11.3. Пароли

Пароли используются для обеспечения безопасности при использовании связи по протоколу IP. Они задаются в различных телекоммуникационных схемах при помощи файла .csipasswd, команд PakBus в приложении CRBasic, команд TCP/IP в приложении CRBasic и в настройках CR1000.

8.11.3.1. .csipasswd

Файл .csipasswd представляет собой файл, который создается и редактируется в программе *DevConfig* *(стр. 111)* и находится на диске CPU: регистратора CR1000. В нем хранятся учетные данные (имена пользователей и пароли), необходимые для доступа к функциям регистратора данных при использовании связи по протоколу IP. См. *Web API* *(стр. 423)* для получения более подробной информации по файлу .csipasswd.

8.11.3.2. Команды PakBus

Следующие команды PakBus приложения CRBasic предусматривают возможность защиты паролем:

- **ModemCallBack()**
- **SendVariable()**
- **SendGetVariables()**
- **SendFile()**
- **GetVariables()**
- **GetFile()**
- **GetDataRecord()**

8.11.3.3. Команды TCP/IP

Следующие команды CRBasic, обслуживающие функции IP-протокола регистратора CR1000, предусматривают возможность защиты паролем:

- **EMailRecv()**
- **EMailSend()**
- **FTPClient()**

8.11.3.4. Настройки — пароли

Настройки, доступ к которым можно получить с помощью программы *DevConfig* *(стр. 111)*, позволяют вводить следующие пароли:

- **пароль PPP;**
- **пароль PakBus/TCP;**
- **пароль FTP;**
- **пароль TLS (безопасность на транспортном уровне (Transport Layer Security, TLS) активна);**
- **пароль для секретного ключа протокола TLS;**
- **ключ шифрования связи PakBus стандарта AES-128** *(стр. 471)***.**

См. раздел *Состояние, настройки и DTI (регистры* *(стр. 114)**)* для получения более подробной информации.

8.11.4. Шифрование файла

Шифрование доступно для программных файлов CRBasic и обеспечивает средство защиты проприетарного кода или устойчивость программы ко взлому. Шифрование возможно для файлов .CR<X> или файлов, заданных командой **Include()**. Регистратор CR1000 дешифрует программные файлы в процессе работы. В то время как другие типы файлов могут быть зашифрованы, средства для дешифрования не предусматриваются.

Инструмент шифрования редактора *CRBasic Editor* (**Menus | File | Save and Encrypt** (Меню | Файл | Сохранить и зашифровать)) создает зашифрованную копию исходного файла в памяти ПК. Зашифрованный файл называется по имени исходного файла, но к нему добавляется «_enc». Исходный файл не меняется. По команде **FileEncrypt()** осуществляется шифрование файлов, находящихся в памяти CR1000. Зашифрованный файл записывается с заменой исходного, и ему присваивается имя исходного файла. По команде **Encryption()** осуществляется шифрование и дешифрование содержимого файла.

Одним из способов применения шифрования файлов является защита проприетарного кода с предоставлением возможности его копирования.

8.11.5. Шифрование связи

PakBus является ключевым коммуникационным протоколом CR1000. За счет шифрования определенной части передаваемых по протоколу PakBus данных достигается высокий уровень безопасности. Дополнительная информация приведена в разделе *Шифрование PakBus* *(стр. 406)*.

8.11.6. Скрытие файлов

Функция скрытия программных файлов CRBasic предоставляет инструмент, используемый отдельно или в сочетании с шифрованием файлов, для обеспечения защиты проприетарного кода от копирования или устойчивости от взлома. Файлы .CR<X> или файлы, заданные командой **Include()**, можно скрыть при помощи команды **FileHide()**. Регистратор CR1000 может разместить и использовать скрытые файлы в процессе работы, однако файл или имя файла не будет доступно для просмотра. См. раздел *Управление файлами* *(стр. 382)* для получения более подробной информации.

8.11.7. Подписи

Системы регистрации и мониторинга и программные подписи являются важными компонентами схемы безопасности. Более подробная информация по использованию подписей приводится в разделах *Программирование для использования подписей* *(стр. 169)* и *Подписи: примеры программ* *(стр. 178)*.

9. Обслуживание — подробная информация

Связанные темы:
• *Обслуживание — общие сведения (стр. 93)*
• *Обслуживание — подробная информация (стр. 473)*

- Защита CR1000 от воздействия влажности и сырости.
- Периодическая замена внутренней литиевой батареи.
- Отправка в Campbell Scientific для заводской калибровки каждые три года.

9.1. Защита от влаги — подробная информация

Защита от влаги — общие сведения (стр. 93)
Защита от влаги — подробная информация (стр. 99)
Защита от влаги — изделия (стр. 660)

Когда уровень влажности достигает точки росы, появляется конденсат, который может повредить CR1000. Поэтому пользователь должен обеспечить эффективный контроль влажности прибора.

Модуль CR1000 защищен от влаги пакетом силикагеля, который вкладывается на заводе-изготовителе. При каждом ремонте CR1000 в Campbell этот пакет заменяется. В обычной ситуации модуль открывать не следует, кроме замены внутренней литиевой батареи.

Для обеспечения дополнительной защиты внутрь корпуса необходимо поместить специальный осушитель.

9.2. Замена внутренней батареи

ВНИМАНИЕ! Риск пожара, взрыва и сильного ожога. Неправильное использование или установка внутренней литиевой батареи может привести к серьезным травмам. Запрещается перезаряжать, разбирать, нагревать до температуры свыше 100 °C, припаивать к ячейке, сжигать или погружать содержимое в воду. Утилизация отработанных литиевых батарей должна выполняться в соответствии со всеми местными требованиями.

В CR1000 встроена литиевая аккумуляторная батарея, которая питает часы и SRAM, когда CR1000 не подключен к источнику питания. CR1000 не получает электропитание от литиевой батареи, если она полностью заряжена *источником питания (стр. 85)*. Если CR1000 хранится при комнатной температуре, срок эксплуатации литиевой батареи составляет около трех лет (меньше при экстремальных температурах). При непрерывной работе CR1000 от источника питания ячейки литиевой батареи служат гораздо дольше.

Когда электропитание подается от внешнего источника, регистратор CR1000 измеряет напряжение литиевой батареи каждые 24 часа. Данное напряжение отображается в таблице **Status (Состояние)** (см. приложение *Таблица состояния* и настройки *(стр. 603)*) в поле **Литиевая батарея**. Напряжение новой батареи составляет приблизительно 3,6 В пост. тока. Замените батарею, если напряжение падает приблизительно до 2,7 В пост. тока.

- При извлечении литиевой батареи (или при падении напряжения ниже 2,7 В пост. тока, когда основной источник питания регистратора CR1000 отключен), программа CRBasic и большинство настроек сохраняются, однако пропадают следующие элементы:

- Настройки «выполнить сейчас» и «выполнить при включении электропитания»
- Журналы маршрутизации и коммуникации (ведение возобновляется без вмешательства пользователя).
- Время. После замены батареи значение времени необходимо настроить заново.
- Таблицы данных конечной памяти.

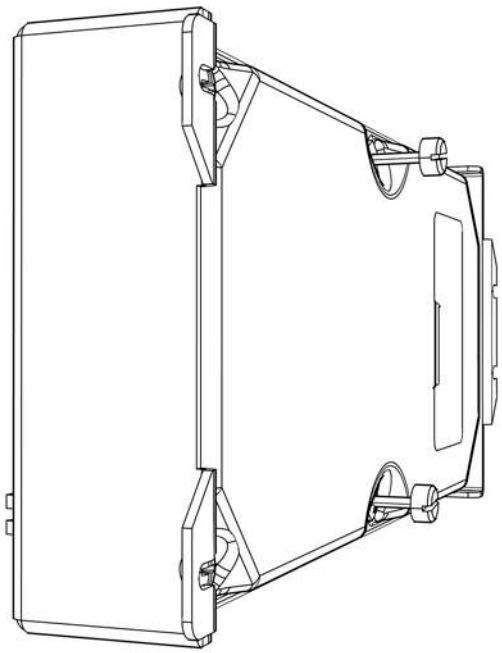
Новую литиевую батарею можно приобрести в Campbell Scientific или у другого поставщика. В таблице *Технические характеристики внутренней литиевой батареи* *(стр. 474)* приводятся каталожные номера батарей и основные технические характеристики.

Таблица 127. Технические характеристики внутренней литиевой батареи

Изготовитель	Tadiran
Номер модели Tadiran	TL-5902/S
Каталожный номер Campbell Scientific, Inc.	13519
Напряжение	3.6 V
Емкость	1,2 A*ч
Скорость саморазряда	1 %/год при 20 °C
Диапазон рабочих температур	от –55 до 85 °C

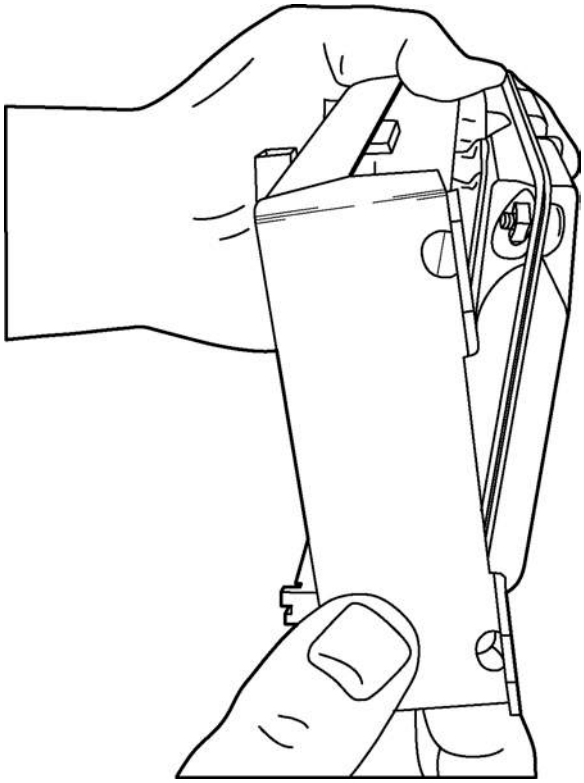
При установке модуля на коммутационную панель убедитесь, что модуль надежно посажен или подключен к коммутационной панели, вручную сжав детали между собой.

Рисунок 127. Ослабьте винты крепления



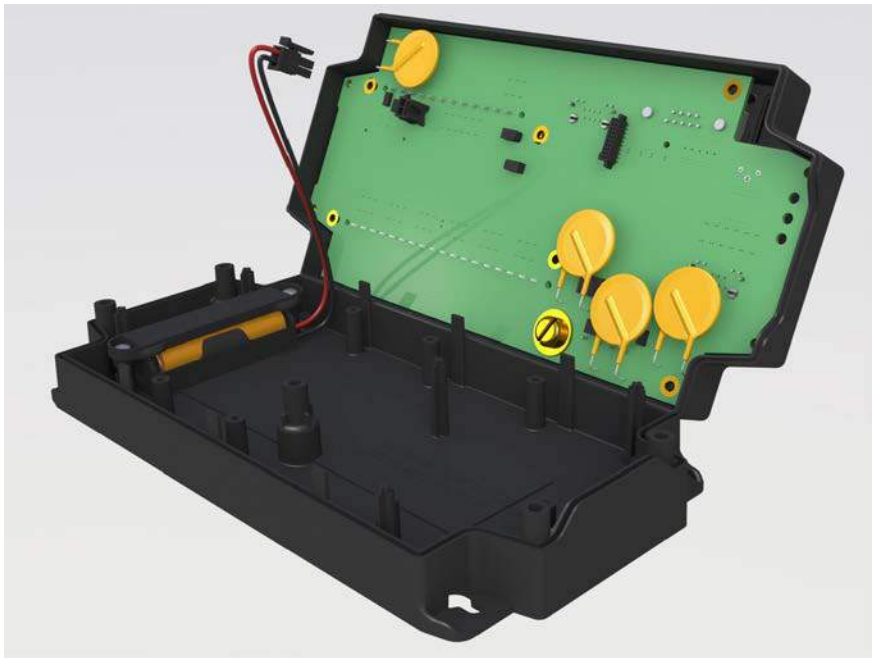
Полностью ослабьте (только ослабьте) два винта с накатанной головкой. Они должны остаться ввинченными в модуль.

Рисунок 128. Отведите край от панели



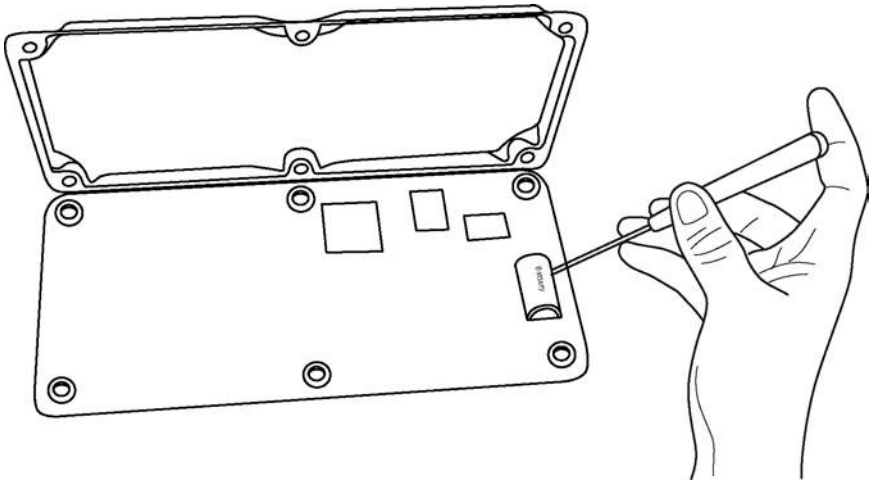
Отведите один край коробки от коммутационной панели, чтобы отсоединить ее от трех внутренних креплений.

Рисунок 129. Отверните гайки, чтобы разобрать коробку



Выверните шесть гаек, затем откройте коробку.

Рисунок 130. Снимите и замените батарею



Снимите литиевую батарею, приподняв ее небольшой плоской отверткой. Сборка CR1000 осуществляется в порядке, обратном разборке. Уделите особое внимание тому, чтобы крышка коробки была плотно посажена на трех креплениях.

9.3. Порядок заводской калибровки или ремонта

Связанные темы
- *Автоматическая калибровка — общие сведения (стр. 92)*
- *Автоматическая калибровка — подробная информация (стр. 344)*
- *Автоматическая калибровка — ошибки (стр. 490)*
- *Компенсация напряжения смещения (стр. 323)*
- *Заводская калибровка (стр. 94)*
- *Заводская калибровка или процедуры ремонта (стр. 476)*

Если регистратор CR1000 отправляется в Campbell Scientific для калибровки или ремонта, сперва обратитесь к инженеру по прикладному применению Campbell Scientific. Если модуль CR1000 неисправен, будьте готовы выполнить диагностику, общаясь при этом с инженером по прикладному применению по телефону. Множество проблем можно решить по телефону. Если требуется калибровка или ремонт, при отправлении изделия необходимо выполнить следующие процедуры:

Возврат продуктов возможен только после получения авторизованного разрешения. Ниже приведена контактная информация для заказчиков из США и из стран, прямо обслуживаемых Campbell Scientific, Inc. Аффилированные компании обеспечивают ремонт дефектных продуктов для заказчиков на своей территории. Чтобы узнать, какая компания Campbell Scientific обслуживает вашу страну, пройдите по ссылке *www.campbellsci.com* .

Чтобы получить авторизованное разрешение на возврат материалов (RMA), свяжитесь с Campbell Scientific, Inc. по телефону +1 (435) 227-2342. После того, как технические специалисты рассмотрят Ваше обращение, вам будет выдано авторизованное разрешение на возврат. Запишите номер разрешения на внешней стороне транспортной тары. Почтовый адрес для возврата дефектных изделий:

CAMPBELL SCIENTIFIC, INC.
Авторизованное разрешение на возврат #_____
815 Уэст 1800 Норт
США, Юта, г. Логан, 84321-1784

При возврате изделия заказчик заполняет «Заявление об очистке и отсутствии загрязнений» и исполняет все перечисленные в данной форме требования. Форма заявления доступна на вебсайте Campbell по ссылке *www.campbellsci.com/repair*. Заполненная форма пересылается по электронной почте на адрес *repair@campbellsci.com* либо факсом на номер +1 (435-2279579) 227-9106. Мы не принимаем изделия, для которых заказчик не предоставил данное заявление. Если в течение трех дней с даты доставки изделия заявление не было получено или было получено в частично заполненном виде, изделие возвращается заказчику с оплатой почтовых расходов за его счет. Campbell Scientific оставляет за собой право отказаться от обслуживания изделий, подвергавшихся загрязнению, потенциально опасному для здоровья или безопасности наших сотрудников.

10. Диагностика

Если система не работает должным образом, просим обратиться к инженеру по прикладному применению Campbell Scientific за консультацией. Если используются датчики, периферийные устройства или телекоммуникационное оборудование, ознакомьтесь с руководствами по этим изделиям для получения дополнительной информации.

Примечание. Если изделие производства Campbell Scientific необходимо вернуть на ремонт или повторную калибровку, в первую очередь потребуется номер *авторизованного разрешения на возврат материалов* *(стр. 3)***.** Свяжитесь с инженером по прикладному применению Campbell Scientific.

10.1. Диагностика — средства первой необходимости

- Мультиметр (включающий в себя вольт- и омметр). Для этих целей вполне подходят недорогие ($20) приборы. Более дорогие приборы имеют дополнительные режимы работы, которые достаточно полезны в некоторых ситуациях.
- Мобильный или спутниковый телефон с контактами инженеров по прикладному применению Campbell Scientific. Созвонитесь с ними до того, как отправляться на место проведения работ. Инженер по прикладному применению может дать некоторые советы, благодаря которым вы сможете лучше подготовиться до выхода на место проведения работ.
- Документация по изделию в доступном формате и удобочитаемом виде по месту установки. Отражение солнечного света, пыль и влага зачастую затрудняет использование электронных устройств.

10.2. Диагностика — основная процедура

1. Проверьте напряжение основного источника питания на клеммах **POWER IN** на лицевой панели CR1000.
2. Проверьте провода и кабели на наличие следующих несоответствий:
   - ослабленные точки соединения;
   - поврежденные разъемы;
   - перерезанные провода;
   - повреждение изоляции, в результате чего влага проникает в кабель. Проникновение воды, независимо от того, контактирует она с жилой или нет, может привести к отказу системы. Вода может увеличить диэлектрическую постоянную кабеля в степени, достаточной для создания помех сигналам датчика, либо может проникнуть в сам датчик, что повлечет за собой выход из строя электронного блока датчика.
3. Проверьте программу CRBasic. Если программа была написана исключительно при помощи *Short Cut*, то маловероятно, что программа является источником проблемы. Если программа была написана или редактировалась в редакторе *CRBasic Editor*, логические и синтаксические ошибки могли легко прокрасться в код. Чтобы выполнить поиск ошибок, создайте «раздетую» версию программы или разбейте ее на несколько небольших блоков, чтобы проверить их по отдельности. Например, если в датчике не работает преобразование сигнала в данные, создайте программу, с помощью которой измерения проводятся только с помощью этого датчика и сохраняются данные, отсутствующие по всем другим входами и данным. Перед тем как отправиться на место проведения диагностических работ, напишите такие минипрограммы, если это возможно.

10.3. Диагностика — источники ошибок

Системы сбора данных — сложные, количество возможных конфигураций —

бесконечно, а количество перестановок групп каналов — невообразимо. Тем не менее, используя систематический подход на основе принципов независимого определения, можно установить первопричину большинства ошибок и применить к ним соответствующие меры по устранению.

Ошибки обозначаются различными средствами, некоторые из которых фактически обмениваются данными с использованием слова **Error** (Ошибка). Аспекты, указывающие на то, что необходимо уделить повышенное внимание:

- **ошибка;**
- **NAN;**
- **INF;**
- быстрое изменение показаний;
- неверные показания;

Эти аспекты проявляются в различных формах и в различных местах.

Ключевой концепцией диагностики является *независимая проверка*, которая основывается на использовании внешних инструментов для проверки функционирования неисправного компонента системы. Например, мультиметр является независимым измерительным прибором, который может использоваться для проверки сигнала датчика, сопротивления датчика, подачи электропитания, целостности кабелей, возбуждения, контрольных выходов и т. д.

Отличным местом для начала проверки являются данные, генерируемые системой. В первую очередь, вы должны уметь с первого взгляда определять, соответствуют ли данные номинальному диапазону. Например, возьмем случай измерения фотосинтетического фотонного потока (PPF). Номинальный диапазон PPF составляет от 0 (темно) до около 2000 мкмоль $м^{-2}$ $с^{-1}$. Если измеренное значение меньше 0 или больше чем 2000, где-то в системе имеется ошибка. Если измеренное значение равно 1000 в условиях ясного летнего неба, где-то в системе имеется ошибка.

Источники ошибок обычно находятся в одной или нескольких следующих категориях:

- Программа CRBasic
  - если программа была полностью написана приложением Short Cut, ошибки маловероятны;
  - если программа была написана или отредактирована человеком, наличие ошибок более вероятно;
  - назначение каналов, коды диапазона входных сигналов и аргументы режима измерения являются распространенным источником ошибок.
- Аппаратные средства
  - весьма распространено неверное подключение датчиков или источников питания;
  - повреждение оборудования;
  - вода, влажность, разряд молнии, переходное напряжение, ЭМП;
  - видимые признаки повреждений;
  - самодиагностика;
  - ошибки сторожевой схемы.
- Встроенное ПО
  - ошибки операционной системы — редкое, но вероятное явление.
- ПО поддержки регистратора данных
  - ошибки крайне редки, но случаются.
- Ошибки, вызванные внешними факторами

10.4. Диагностика — таблица состояния

Информация из таблицы **Status (Состояние)** может пролить свет на множество проблем. В приложении *Таблица состояния и настройки (стр. 603)* представлены регистры таблицы **Status (Состояние)**, а также некоторые советы по использованию этой информации при диагностике.

См. раздел *Таблица состояния как средство отладки (стр. 485)*. Многие из этих ошибок совпадают с аналогичными ошибками в служебной программе Station Status (Состояние станции), входящей в комплекс ПО поддержки регистратора данных.

10.5. Программирование

Выполните анализ данных сразу после развертывания, чтобы убедиться, что измерение и хранение данных регистратором CR1000 осуществляется должным образом. Большинство проблем с измерением и хранением данных происходит по причине программных ошибок CRBasic.

10.5.1. Программа не компилируется

Несмотря на то, что компилятор в *CRBasic Editor* утверждает, что программа компилируется успешно, программа может не выполняться или не компилироваться в CR1000. Это редкий случай, но причины могут скрываться в следующем:

- CR1000 имеет другую (обычно более старую) операционную систему, которая не полностью совместима с компилятором на ПК. В случае возникновения сомнений проверьте версии обеих ОС. Версия компилятора на ПК отображается в первой строке результатов компилирования.
- Программа требует выделения большого объема памяти для таблиц данных или переменных, а регистратор CR1000 не обладает адекватной памятью. Обычно это обозначается при компиляции в результатах компилирования. Если происходит ошибка данного типа, проверьте следующее:
    - Копии старых программ на диске CPU:. CR1000 хранит копии всех программных файлов, пока они не будут удалены, диск не будет отформатирован или не будет загружена новая операционная система при помощи *DevConfig (стр. 111)*.
    - Если создан диск USR:, не имеет ли он слишком большой размер. Диск USR: может использовать память, необходимую для работы программы.
    - Программа, написанная для регистратора CR1000 с памятью объемом 4 МБ загружена в CR1000 с объемом памяти 2 МБ.
    - Карта памяти (CF) недоступна, когда программа пытается получить доступ к диску CRD:. Это может быть проблемой только в том случае, если команда **TableFile()** или **CardOut()** включена в программу.

10.5.2. Программа компилируется / не работает корректно

Если программа компилируется, но не работает корректно, частой причиной данной ошибки является несоответствие времени. Ни редактор *CRBasic Editor*, ни компилятор CR1000 не предпринимают попытку проверить, успевает ли CR1000 выполнить все поставленные программой задачи за выделенный промежуток времени. Если выполнение программы ограничено по времени, уделите больше внимания времени исполнения. Проверьте время измерения и обработки в таблице **Status (Состояние)** (**MeasureTime, ProcessTime, MaxProcTime**) по всем случаям сканирования, затем попробуйте поэкспериментировать с командой **InstructionTimes()** в программе. Анализ результатов команды **InstructionTimes()** может быть затруднен ввиду

многозадачности регистратора, однако это может быть полезным инструментом тонкой настройки программы.

10.5.3. NAN и ±INF

NAN (not-a-number, не число) и **±INF** (бесконечность) являются словами данных, указывающими на исключительные события в работе регистратора данных. **NAN** — это константа, которую можно использовать в выражениях, как показано в нижеприведенной части кода, которая устанавливает контрольную функцию CRBasic (флаг), если направление ветра имеет значение **NAN**:

```
If WindDir = NAN Then
   WDFlag = False
Else
   WDFlag = True
EndIf
```

NAN также может использоваться в сочетании с функцией отключения переменной (***DisableVar***) в командах обработки выхода (хранения данных), как показано в примере CRBasic *Использование NAN для фильтрации данных (стр. 484)*.

10.5.3.1. Измерения и NAN

С помощью **NAN** указывается недействительное измерение.

10.5.3.1.1. Измерение напряжения

CR1000 предусматривает возможность измерения напряжения в следующих диапазонах, определяемых пользователем: ±5000 мВ, ±2500 мВ, ±250 мВ и ±25 мВ. Входные сигналы, выходящие за пределы указанных диапазонов, обозначаются с помощью **NAN**. Если установлена автоматическая настройка диапазона для выбора наилучшего диапазона входных сигналов, **NAN** указывает на то, что один или оба измерения в последовательности автоматически настроенных диапазонов выходят за допустимые пределы. См. раздел *Ошибки калибровки (стр. 490)*.

Вход по напряжению, не соединенный с датчиком, не имеет соединения с землей, а измеренное напряжение зачастую приблизительно равно напряжению предыдущего измерения. Измерения с помощью незаземленных приборов могут смещаться по времени и имитировать действительные измерения. Вариант кода диапазона **C** (обнаружение размыкания на входе/неопределенное значение синфазного режима) используется для присвоения результата NAN разомкнутым (незаземленным) входам.

10.5.3.1.2. Измерения SDI-12

NAN загружается в первую переменную **SDI12Recorder()** при следующих условиях:

- CR1000 занят командами терминалов;
- если команда недействительна;
- если датчик прекращает работу со значением CR LF, а данные при этом отсутствуют.
- если для количества значений выводится **0** в ответ на команду **M!** или **C!**.

10.5.3.2. Математика с плавающей запятой, NAN и ±INF

В таблице *Математические выражения и результаты CRBasic (стр. 483)* приводятся математические выражения, их форма CRBasic и результаты математических вычислений с плавающей запятой IEEE, загруженные в переменные, объявленные как FLOAT или STRING.

10.5.3.3. Типы данных, NAN и ±INF

NAN и **±INF** представляются различным образом в зависимости от объявленного типа данных переменных. Более того, они регистрируются по-разному в зависимости от типа данных конечной памяти, выбранного совместно с типом данных объявленной переменной, используемой в качестве источника (таблица *Типы данных переменных и FS с NAN и ±INF* *(стр. 483)*). Например, **INF**, в переменной, объявленной **As LONG**, представлено целочисленным значением — **2147483648**. Если данная переменная используется в качестве источника и если слово конечной памяти выбирается как UINT2, то оно сохраняется со значением 0.

Таблица 128. Математические выражения и результаты CRBasic

Выражение	*Выражение CRBasic*	*Result*
0 / 0	0 / 0	**NAN**
∞ - ∞	(1 / 0) - (1 / 0)	**NAN**
$(-1)^{\infty}$	-1 ^ (1 / 0)	**NAN**
0 • -∞	0 • (-1 • (1 / 0))	**NAN**
±∞ / ±∞	(1 / 0) / (1 / 0)	**NAN**
1^{∞}	1 ^ (1 / 0)	**NAN**
0 • ∞	0 • (1 / 0)	**NAN**
x / 0	1 / 0	**INF**
x / -0	1 / -0	**INF**
-x / 0	-1 / 0	**-INF**
-x / -0	-1 / -0	**-INF**
∞^0	(1 / 0) ^ 0	**INF**
0^{∞}	0^ (1 / 0)	**0**
0^0	0 ^ 0	**1**

Таблица 129. Типы данных переменных и конечной памяти с NAN и ±INF

Тип переменной	*Тестовое выражение*	*Переменные типа Public и Dim*	*Тип данных конечной памяти и связанные хранимые значения*							
			FP2	*IEEE4*	*UINT2*	*UNIT4*	*STRING*	*BOOL*	*BOOL8*	*LONG*
As FLOAT	1 / 0	**INF**	**INF**[1]	**INF**[1]	**65535**[2]	**4294967295**	**+INF**	**TRUE**	**TRUE**	**2,147,483,647**
	0 / 0	**NAN**	**NAN**	**NAN**	**0**	**2147483648**	**NAN**	**TRUE**	**TRUE**	**-2,147,483,648**
As LONG	1 / 0	**2,147,483,647**	**7999**	**2,147484E09**	**65535**	**2147483647**	**2147483647**	**TRUE**	**TRUE**	**2,147,483,647**
	0 / 0	**2,147,483,648**	**-7999**	**-2,147484E09**	**0**	**2147483648**	**-2147483648**	**TRUE**	**TRUE**	**-2,147,483,648**
As Boolean (Как логический)	1 / 0	**TRUE**	**-1**	**-1**	**65535**	**4294967295**	**-1**	**TRUE**	**TRUE**	**-1**
	0 / 0	**TRUE**	**-1**	**-1**	**65535**	**4294967295**	**-1**	**TRUE**	**TRUE**	**-1**
As STRING	1 / 0	**+INF**	**INF**	**INF**	**65535**	**2147483647**	**+INF**	**TRUE**	**TRUE**	**2147483647**
	0 / 0	**NAN**	**NAN**	**NAN**	**0**[3]	**2147483648**	**NAN**	**TRUE**	**TRUE**	**-2147483648**

[1] Кроме случаев, когда **Average()** выводит **NAN.**

[2] Кроме случаев, когда **Average()** выводит **0.**

[3] **65535** в операционных системах до версии 28.

10.5.3.4. Обработка выхода и NAN

Если результатом измерения или обработки является **NAN**, любой выходной процесс ***DisableVar*** **= FALSE**, сюда входят измерение **NAN**. Например,

```
Average(1,TC_TempC,FP2,False)
```

результатом данной команды будет **NAN**, который хранится как данные конечной памяти для данного интервала.

Тем не менее, если команде ***DisableVar*** присваивается значение **TRUE** при каждом результате **NAN** для измерения, в выходной процесс будут включены измерения, не имеющие результата NAN. Пример CRBasic *Использование NAN для фильтрации данных (стр. 484)* демонстрирует использование условных выражений для присваивания команде ***DisableVar*** значения **TRUE**, чтобы отфильтровать **NAN** в выходных процессах.

Примечание. Если все измерения имеют результат **NAN, NAN** будет сохранен как данные конечной памяти вне зависимости от применения ***DisableVar***.

CRBasic: пример 69. Использование NAN для фильтрации данных

```
Данный пример программы демонстрирует использование NAN для фильтрации данных, используемых в
функциях обработки выходных данных, таких как средние, максимальные и минимальные значения.
@@averages, maxima, and minima.

Объявление переменных и единиц измерения
Public TC_RefC
Public TC_TempC
Public DisVar As Boolean

Описываем таблицы данных
DataTable(TempC_Data,True,-1)
   DataInterval(0,30,Sec,10)
   Average(1,TC_TempC,FP2,DisVar)              Выходной процесс
EndTable

Основная программа
BeginProg
   Scan(1,Sec,1,0)

      Измерение эталонной температуры термопары
      PanelTemp(TC_RefC,250)

      Измерение температуры термопары
      TCDiff(TC_TempC,1,mV2_5,1,TypeT,TC_RefC,True,0,250,1.0,0)

      Фильтр DisVar
      If TC_TempC = NAN Then
        DisVar = True
      Else
        DisVar = False
      EndIf

      Вызов таблицы данных и хранение данных
      CallTable(TempC_Data)

   NextScan
EndProg
```

10.5.4. Таблица состояния, как средство отладки

Связанные темы:
• *Состояние, настройки и информация о таблицах данных (Status/Settings/DTI)* *(стр. 603)*
• *Общее назначение таблицы состояния* *(стр. 604)*
• *Таблица состояния, как средство отладки* *(стр. 485)*

При разработке программы или если в программе предполагается наличие проблемы, проверьте таблицу **Status (Состояние)** регистратора CR1000. Критически важными регистрами таблицы **Status (Состояние)** являются: **CompileResults, SkippedScan, SkippedSlowScan, SkippedRecord, ProgErrors, MemoryFree, VarOutOfBounds, WatchdogErrors** и **Calibration**.

10.5.4.1. CompileResults

CompileResults выводить сообщения, сформированные регистратором CR1000 при загрузке и компиляции программы. Сообщения могут быть также добавлены по мере выполнения программы. Сообщения об ошибках могут быть невидимы ввиду ограниченного отображения. Большую часть данной информации можно гораздо проще получить посредством *ПО поддержки регистратора данных* *(стр. 95)* — отчет о состоянии станции. В сообщении содержится следующая информация:

- программа скомпилирована успешно;
- предупреждения о возможных проблемах;
- ошибки выполнения программы;
- переменные, вызвавшие условия выхода за границы;
- информация сторожевых схем;
- ошибки памяти;

Предупреждающие сообщения выводятся компилятором CRBasic для уведомления о том, что некоторые ожидаемые функции могут не работать. Предупреждения отличаются от сообщений об ошибках тем, что программа сохраняет работоспособность при обнаружении предупреждающего условия.

Редко ошибка обозначается сообщением типа **mem3 fail**. Данные сообщения могут быть вызваны случайным повреждением внутренней памяти. Если они регулярно выявляются с помощью данной программы, это указывает на ошибку операционной системы. Сообщения **Mem3 fail** не являются ошибками пользователя и крайне редко являются ошибками аппаратных средств. Сообщайте о всех случаях возникновения данной ошибки инженеру по прикладному применению Campbell Scientific, особенно если проблема повторяется. Маловероятно, чтобы программа, генерирующая данные ошибки, работала корректно.

Примеры некоторых наиболее распространенных предупреждающих сообщений представлены в таблице *Примеры предупреждающих сообщений (стр. 486)*.

Таблица 130. Примеры предупреждающих сообщений

Сообщение	*Значение*
ЦП/ DEFAULT.CR1 -- Compiled in PipelineMode. (Скомпилировано в PipelineMode.) **Error(s) in CPU:NewProg.CR1: (Ошибки в CPU:NewProg.CR1:)** **line 13: Undeclared variable Battvolt. (строка 13: Необъявленная переменная Battvolt.) @@@Undeclared variable Battvolt.**	Произошла ошибка компиляции новой программы, отправленной в регистратор данных, и регистратор данных вернулся к выполнению DEFAULT.cr1.
Warning: Cannot open include file CPU: Filename.cr1(Предупреждение! Невозможно открыть вложенный файл CPU: Filename.cr1) @@@Cannot open include file CPU: Filename.cr1	Имя файла, указанное в команде Include, не совпадает ни с одним из файлов, находящихся на указанном диске. Поскольку он не найден, часть кода, на которую ссылается команда Include, не будет выполнятся.
Warning: Cannot open voice.txt(Предупреждение! Невозможно открыть voice.txt) Cannot open voice.txt	voice.txt, файл, необходимый для использования с голосовым телефонным модемом COM310, не был найден на диске CPU:.
Warning: COM310 word list cannot be a variable.(Предупреждение! Список слов COM310 не может быть переменной.)	Параметр *Phrases* команды **VoicePhrases()** был назначен имени переменной, а не необходимой строке с разделенными запятой словами из файла Voice.TXT.
Warning: Compact Flash Module not detected: CardOut not used.(Предупреждение! Модуль Compact Flash не обнаружен: команда CardOut не используется.) Модуль Compact Flash не обнаружен: Команда CardOut не используется.	**Команды CardOut()** в программе будут игнорироваться, поскольку карта CompactFlash (CF) не была обнаружена при компиляции программы.
Warning: EndIf never reached at runtime.(Предупреждение! Команда EndIf не была достигнута во время выполнения программы.) @@@EndIf never reached at runtime.	Программа не выполнит команду **EndIf**. В этом случае, причиной является то, что в команде **Scan()** параметр *Count* имеет значение 0, что создает бесконечную петлю в программной логике.

Таблица 130. Примеры предупреждающих сообщений

Сообщение	*Значение*
@@@Warning: Internal Data Storage Memory was re-initialized.(Предупреждение! Внутренняя память для хранения данных была повторно инициализирована.)	В результате отправки новой программы конечная память была перераспределена. Предыдущие данные больше недоступны.
Warning: Machine self-calibration failed.(Предупреждение! Сбой при самокалибровке машины.)	Указывает на проблему с аналоговым измерительным оборудованием во время самокалибровки. Данная ошибка может быть вызвана тем, что недействительный внешний сигнал датчика напряжением свыше внутреннего напряжения ±8 В пост. тока подается на вход по напряжению. При устранении недействительного сигнала и выключение-включение питания регистратора данных самокалибровка запустится снова. Если ошибка не проявляется при выключении-включении питания, проблема устранена. Если недействительные внешние сигналы отсутствуют, но во время выполнения самодиагностики при включении питания снова происходит ошибка, регистратор CR1000 необходимо направить на ремонт квалифицированному специалисту.
Warning: Slow Seq 1, Scan 1, will skip scans if running with Scan 1(Предупреждение! Slow Seq 1, Scan 1 пропустят сканирование, если программа выполняется с Scan 1)	**Скорость сканирования SlowSequence**<= основной скорости сканирования. Это приведет к пропуску сканирования с параметром **SlowSequence**.
Warning: Table [tablename] is declared but never called. (Предупреждение! Таблица [имя таблицы] объявлена, но не была вызвана.)	Данные не будут сохранены в [tablename], поскольку в программе отсутствует команда **CallTable()**, которая ссылается на данную таблицу.
Warning: Units: **a units name that is more than 38 char a... too long will be truncated to 38 chars.(Предупреждение! Блоки: имена блоков, длина которых превышает 38 символов, будут сокращены до 38 символов.)**	Метка, присвоенная с аргументом **Units**, имеет чрезмерную длину и будет сокращена до максимально допустимой длины.
Warning: Voice word TEH is not in Voice.TXT file(Предупреждение! Голосовое слово TEH не содержится в файле Voice.TXT.)	Неправильно написанное слово TEH в команде **VoiceSpeak()** не найдено в файле Voice.TXT и не будет произнесено голосовым модемом.

10.5.4.2. SkippedScan

Пропуск сканирования возникает, если для длинной программы задан короткий интервал сканирования, несколько команд **Scan() / NextScan** выходят за пределы **SubScan() или SlowSequence** или в результате других операций, выполнением которых занят процессор на момент начала сканирования. Случайный пропуск сканирования допускается, но необходимо принять меры по их недопущению. Пропуски сканирования могут ухудшить измерения частоты, производимые на клеммах, настроенных на импульсный вход. Ошибка возникает по причине того, что подсчеты в ходе сканирования и последующего пропущенного сканирования воспринимаются регистратором CR1000 как произошедшие в течение одного сканирования. Измеренная частота может быть выше фактической. Учитывайте, что сканирования, которые хранят данные, не пропускаются. Если какие-либо сканирования пропускаются регулярно, может потребоваться оптимизация программы регистратора данных или сокращения объема обработки данных под управлением процессора.

Пропущенные сканирования в режиме Pipeline Mode указывают на необходимость увеличения максимальной глубины буферизации. Попробуйте увеличить число буферов сканирования (третий параметр команды **Scan()**) до значения, превышающего представленное в регистре **MaxBuffDepth** в таблице

Status (Состояние).

10.5.4.3. SkippedSlowScan

CR1000 автоматически запускает медленную последовательность для обновления таблицы калибровки. Если медленная последовательность калибровки пропускается, CR1000 попробует повторить данный шаг процесса калибровки в следующий раз. Это просто увеличит время калибровки.

10.5.4.4. SkippedRecord

SkippedRecord обычно увеличивается инкрементами, когда пропускается событие записи в таблицу данных, что обычно происходит при пропуске сканирования. **SkippedRecord** не увеличивается всеми событиями, которые оставляют пробелы в данных, в том числе циклы включения-выключения питания регистратора CR1000.

10.5.4.5. ProgErrors

Должен иметь значение **0**. В противном случае установите причину.

10.5.4.6. MemoryFree

Значение менее 4 кБ является слишком малым и может повлечь за собой возникновение ошибок, связанных с буфером памяти.

10.5.4.7. VarOutOfBounds

Связанные темы:
• *Объявление массивов (стр. 135)*
• Массивы мультипликаторов и смещений
• *VarOutOfBounds (Выход переменной за пределы) (стр. 488)*

При программировании при помощи массивов переменных внимание необходимо уделять соответствию размера массива потребностям программы. Например, если операция пытается осуществить запись в 16 элементах массива **ExArray(),** а для **ExArray()** было объявлено 15 элементов (например, **Public** ExArray(15)), счетчик ошибок выполнения программы **VarOutOfBound** увеличивается в таблице **Status (Состояние)** в каждом случае отсутствия шестнадцатого элемента.

CR1000 предпринимает попытку выявить ошибки **VarOutOfBound** на момент компиляции (не следует путать с предварительным компилятором *CRBasic Editor*, который не предпринимает таких попыток). Когда ошибка **VarOutOfBound** обнаруживается при компиляции, CR1000 пытается вывести выходящие за пределы переменные в конце сообщения **CompileResults** в таблице **Status (Состояние)**. Например, CR1000 может обнаружить, что **ExArray()** не обладает достаточным размером и записать **Warning:Variable ExArray out of bounds (Предупреждение! Переменная ExArray выходит за пределы)** в регистре **CompileErrors**.

CR1000 не выявляет все ошибки, связанные с выходом за пределы, поэтому необходимо особенно внимательно следить за тем, чтобы все массивы имели подходящий размер.

10.5.4.8. Ошибки сторожевых схем

Ошибки сторожевых схем указывают на то, что в CR1000 произошел критический отказ и он перезагрузился. Наличие нескольких сторожевых схем указывает на то, что CR1000 работает должным образом, и не представляют проблему.

Вероятные основные причины приводятся ниже в порядке от наиболее к наименее вероятной:

- переходное напряжение;
- слишком быстрое выполнение программы CRBasic;

- много следующих друг за другом команд **PortSet()** без задержек;
- высокая скорость передачи последовательных данных по нескольким портам с очень большими пакетами данных.

Если не подходит ни одна из вышеперечисленных причин, обратитесь к инженеру по прикладному применению Campbell Scientific за поддержкой. Причинами для обращения за поддержкой могут служить следующие:

- повреждение памяти. Проверьте наличие ошибок памяти при помощи команды **M** в *режиме терминала (стр. 501);*
- проблемы с операционной системой;
- аппаратная проблема.

Ошибки сторожевой схемы могут приводить к нарушениям связи, которые, в свою очередь, затрудняют процесс поиска и устранения неполадок. Экранная клавиатура CR1000KD будет зачастую выступать в качестве пользовательского интерфейса при сбое связи. Информацию по критическим отказам CR1000 можно найти в трех местах.

- Поле **WatchdogErrors** в таблице **Status (Состояние)** *(стр. 603)*
- Файл Watchdog.txt на диске *CPU: (стр. 374)*. Между возникновением ошибки и созданием файла Watchdog.txt может пройти некоторое время. Не все ошибки являются причиной создания файла. При каждом случае создания файла watchdog.txt просим обратиться к инженеру по прикладному применению Campbell Scientific за консультацией.
- Информация по критическому отказу может указываться в конце регистра **CompileResults** в таблице **Status (Состояние)** *(стр. 603)*.

10.5.4.8.1. Ошибки сторожевой схемы таблицы состояния

Ненулевые значения указывают на критический отказ CR1000, который мог быть вызван проблемами с электропитанием или переходным напряжением или с операционной системой или оборудованием. Если проблемы с электропитанием или переходным напряжением исключены, возможно, необходимо обновить операционную систему CR1000 или отправить его на *ремонт (стр. 3)* в Campbell Scientific.

10.5.4.8.2. Файл Watchdoginfo.txt

Файл **WatchdogInfo.txt** создается на диске CPU: в случае программного перезапуска CR1000 (в отличие от аппаратного перезапуска, в результате которого увеличивается счетчик регистра **WatchdogError** в таблице **Status (Состояние)**). Формирование файла **WatchdogInfo.txt** случается достаточно редко. При каждом случае создания данного файла просим обратиться к инженеру по прикладному применению Campbell Scientific.

Поиск и устранение неполадок, не связанных с определением причины формирования сторожевой схемы, является достаточно трудоемким процессом. Просим обратиться к инженеру по прикладному применению Campbell Scientific за консультацией. Ключевыми элементами поиска являются следующие:

- Ожидает ли несколько задач одинаковых ресурсов? Это всегда вызывает ошибку программного обеспечения.
- В более новых операционных системах имеется информация по областям памяти. Если встречаются записи, подобные **ColorX: fail**, это указывает на повреждение памяти.
- Информация по коммуникационной памяти также может дать подсказки к срабатыванию сторожевых схем PakBus и TCP. Например, если COM1 является источником срабатывания сторожевой схемы, то информация об устройстве, подключенном к данному порту, и скорости передачи данных и частоты обмена данными через этот порт является ценным источником.

10.6. Диагностика — операционные системы

Обновление операционной системы CR1000 может иногда привести к устранению проблемы. Операционные системы можно бесплатно скачать на сайте *www.campbellsci.com/downloads* (*http://www.campbellsci.com/downloads*).

Операционные системы проходят всестороннее тестирование, проводимое профессиональной командой тестировщиков, перед выпуском. Однако функционирование всех новых компонентов системы сбора данных должно тщательно проверяться и тестироваться специалистами по системной интеграции и конечным пользователем.

10.7. Диагностика — ошибки автоматической калибровки

Связанные темы
- *Автоматическая калибровка — общие сведения (стр. 92)*
- *Автоматическая калибровка — подробная информация (стр. 344)*
- *Автоматическая калибровка — ошибки (стр. 490)*
- *Компенсация напряжения смещения (стр. 323)*
- *Заводская калибровка (стр. 94)*
- *Заводская калибровка или процедуры ремонта (стр. 476)*

Ошибки автоматической калибровки встречаются редко. В случае их возникновения причиной обычно является аналоговый вход, превышающий *входные пределы (стр. 310)* регистратора CR1000.

- Убедитесь, что напряжение на всех аналоговых входах не превышает ±5 В пост. тока, измерив его между входом и клеммой **G** (заземление). Измерения должны проводиться с помощью *мультиметра (стр. 520)*.
- Убедитесь в отсутствии конденсата, который может иногда приводить к утечкам с клеммы источника 12 В пост. тока.
- Проверьте надежность подсоединения провода заземления к датчику, запитанному с клеммы **12 V** или **SW12**.
- При отсутствии мультиметра по одному отсоединяйте датчики, для которых требуется подача электропитания от 9 до 16 В пост. тока. Если восстанавливаются нормальные результаты измерений, причина проблемы найдена.

10.8. Связь

10.8.1. RS-232

Несоответствие скорости передачи данных между CR1000 и *ПО поддержки регистратора данных (стр. 95)* зачастую является причиной коммуникационных проблем. По умолчанию скорость передачи данных в бодах регистратора CR1000 регулируется автоматически для соответствия скорости, заданной программным обеспечением. Однако настройки, измененные в CR1000 под потребности определенного устройства RS-232, такого как интеллектуальный датчик, датчик или модем, могут ограничить пропускную способность порта RS-232 до одного значения скорости передачи данных в бодах. Если это значение скорости передачи данных в бодах совпадает и вводится в параметры ПО поддержки, соединение может быть установлено. После установления соединения настройку скорости передачи данных в бодах в CR1000 можно изменить. Наводящую информацию о том, какая скорость передачи данных в бодах установлена, можно найти, проанализировав текущую и предыдущую программы CR1000 по команде **SerialOpen()**, поскольку **SerialOpen()** задает скорость передачи данных в бодах. Скорость передачи данных в бодах также может указываться в документации изготовителя предыдущего устройства RS-232.

10.8.2. Связь с несколькими ПК

CR1000 может обмениваться данными с несколькими ПК одновременно. Например, CR1000 может быть узлом интернет-сети PakBus, обменивающимся данными с удаленной копией *LoggerNet*. Специалист на месте установки может обеспечить обмен данными между CR1000 и *PC200W* с помощью последовательного соединения, если PakBus-адреса узловых ПК различаются. Все программные средства поддержки регистратора данных Campbell Scientific имеют возможность изменения PakBus-адресов ПК.

10.8.3. Ошибки коммуникационной памяти

CommsMemFree() является массивом, состоящим из трех регистров в таблице **Status (Состояние)** *(стр. 603)* и выводящим ошибки коммуникационной памяти. В общих словах, если значение регистра **CommsMemFree()** равняется или приблизительно равняется нулю, возможно, потребуется обратиться в Campbell Scientific за поддержкой по выявлению и устранению потенциально серьезной проблемы связи. В разделах *CommsMemFree(l)* *(стр. 491)*, *CommsMemFree(2)* *(стр. 492)* и *CommsMemFree(3)* *(стр. 493)* подробно рассматриваются ошибки коммуникационной памяти.

10.8.3.1. CommsMemFree(1)

CommsMemFree(1): количество буферов, используемых во всех видах связи, кроме случаев использования экранной клавиатуры CR1000KD. По два разряда на каждую категорию размера буфера. Старшие значимые разряды задают количество больших буферов. Младшие значимые разряды задают количество малых буферов. Когда протокол *TLS* *(стр. 531)* не активен, используются четыре категории буферов: **очень малая, малая, средняя** и **большая**. Когда протокол TLS активен, используется дополнительная пятая категория — **огромная**, и под каждую категорию выделяется большее количество буферов.

Если требуется буфер определенного размера, наименьший пул подходящего размера, располагающий не менее чем одним свободным буфером, выделяет буфер и сокращает число резервных буферов на единицу. По завершении сеанса связи буфер возвращается в пул, а число буферов данного размера увеличивается на единицу.

Когда TLS активен, число буферов, выделенных для категории **tiny (очень малая)**, может отображаться только как число очень малых буферов по модулю, разделенное на 100.

CommsMemFree(1) кодируется с помощью следующего выражения:

```
CommsMemFree(1) = tiny + lil*100 + mid*10000 + med*1000000 +
lrg*100000000
```

где

tiny = количество доступных пакетов размером 16 байт;

lil = количество пакетов малого размера (≈100 байт);

mid = количество пакетов среднего размера (≈530 байт);

med = количество пакетов большого размера (≈3 кБ);

lrg = количество доступных пакетов крупного размера (≈18 кБ), в первую очередь для TLS.

Следующие выражения используются для выбора индивидуальных значений из **CommsMemFree(1)**:

```
tiny = CommsMemFree(1) % 100
lil = (CommsMemFree(1) / 100) % 100
mid = (CommsMemFree(1) / 10000) % 100
med = (CommsMemFree(1) / 1000000) % 100
Irg = (CommsMemFree(1) / 100000000) % 100
```

Таблица 131. Пример значений по умолчанию CommsMemFree(l) и их использование, TLS не активен

Категория буфера	***Условие: сброс, TLS не активен.*** *Счетчик буфера:* ***CommsMemFree(1) = 15251505.***	***Пример использования***	
		Условие: ***используется, TLS не активен.*** *Счетчик буфера:* ***CommsMemFree(1) = 13241504.***	***Количество используемых буферов (счетчик сброса - счетчик использования)***
очень малая	05	04	1
малая	15	15	0
средняя	25	24	1
большая	15	13	2
огромная			

Таблица 132. Пример значений по умолчанию CommsMemFree(l) и их использование, TLS активен

Категория буфера	***Условие: сброс, TLS активен. Счетчик буфера: CommsMemFree(1) = 230999960.***	***Пример использования***	
		Условие: TLS активен, активные соединения TLS отсутствуют. Подключение к LoggerNet по TCP/IP. Счетчик буфера: CommsMemFree(1) = 228968437.	***Количество используемых буферов (счетчик сброса - счетчик использования)***
очень малая	160	137	23
малая	99	84	15
средняя	99	96	3
большая	30	28	2
огромная[1]	2	2	0

[1] Если активны почтовые клиенты, использующие протокол TLS, «огромная» категория будет сокращена наряду с некоторыми другими категориями.

10.8.3.2. CommsMemFree(2)

CommsMemFree(2) отображает число «кусков» памяти в памяти *«keep»* *(стр. 519)*, используемых в коммуникации. Оно включает память, используемую для маршрутизации PakBus и списков соседних узлов, структур задержки связи и структур соединения TCP/IP. Параметр **PakBusNodes**, которому присваивается значение по умолчанию **50**, входит в состав **CommsMemFree(2).** Дублирование **PakBusNodes** на **100** дублей **CommsMemFree(2)** с ≈300 до ≈600 (при условии, что большая сеть PakBus не была обнаружена только что). Чем больше обнаруженная сеть PakBus и чем больше число одновременных соединений TCP, тем меньшим будет число **CommsMemFree(2)**. Значение 50 параметра **PakBusNodes** обычно является достаточным, и его можно уменьшить в небольших сетях для освобождения памяти, если необходимо. Уменьшение значения **PakBusNodes** на единицу высвобождает 224 байта памяти. Если значение **CommsMemFree(2)** падает и остается на низком уровне без видимых причин (очень редкий случай), просим

обратиться к инженеру по прикладному применению Campbell Scientific, поскольку операционную систему CR1000, вероятно, необходимо настроить.

10.8.3.3. CommsMemFree(3)

CommsMemFree(3) указывает три двузначных поля справа (самый младший разряд) налево (самый старший разряд):

- **lilfreeq** = доступны «малые» пакеты IP;
- **bigfreeq** = доступны «большие» пакеты IP;
- **rcvdq** = пакеты IP в очереди «получено» (все еще не обработаны).

Если при запуске не происходит обмена данными TCP/IP, данное поле будет считываться как 1530, что будет интерпретировано как 30 **lilfreeq** и 15 **bigfreeq** доступно, без пакетов в **rcvdq**. Интерфейсы Ethernet и/или PPP предоставляют данные для **rcvdq**. Если показанием **CommsMemFree(3)** является 21428, до два пакета находятся в очереди «получено», 14 пакетов **bigfreeq** свободно (один используется) и 28 пакетов **lilfreeq** свободно (два используется). Эти три значения также указываются в информации *IP trace (IP-трассировка)* *(стр. 518)* каждые 30 секунд как **lilfreeq, bigfreeq** и **recvdq**. Если количество свободных пакетов **lilfreeq** или **bigfreeq** снижается и приблизительно равняется нулю или если значение **rcvdq** повышается и остается на высоком уровне (оба варианта очень редки), просим обратиться к инженеру по прикладному применению Campbell Scientific, поскольку операционную систему CR1000, вероятно, необходимо настроить.

CommsMemFree(3) кодируется следующим образом:

```
CommsMemFree(3) = lilfreeq + bigfreeq*100 + rcvdq*10000 +
sendq*1000000
```

где

lilfreeq = количество доступных малых пакетов TCP

bigfreeq = количество больших пакетов TCP

rcvdq = количество входных пакетов, на текущий момент ожидающих обслуживания

sendq = количество выходных пакетов, ожидающих отправки

Следующие выражения можно использовать для выбора значений из переменной **CommsMemFree(3)**:

```
lilfreeq = CommsMemFree(3) % 100
bigfreeq = (CommsMemFree(3) / 100) % 100
rcvdq = (CommsMemFree(3) / 10000) % 100
sendq = (CommsmemFree(3) / 1000000) % 100
```

10.9. Диагностика — источники электропитание

Связанные темы:
• Источники питания — технические характеристики
• *Источники питания — краткое руководство (стр. 44)*
• *Источники питания — общие сведения (стр. 85)*
• *Источники питания — общие сведения (стр. 100)*
• *Источники питания — продукты (стр. 657)*
• *Источники питания (стр. 101)*
• *Диагностика — источники питания (стр. 494)*

10.9.1. Диагностика источников электропитания — общие сведения

К системам электропитания относятся: батареи, зарядные регуляторы и первичные источники питания, такие как солнечные панели и трансформаторы переменного тока в переменный и переменного тока в постоянный, подключенные к сети электропитания. Если источник питания не работает надлежащим образом, все эти элементы необходимо проверить.

В разделе *Порядок проведения диагностики и устранения проблем (стр. 495)* приводятся блок-схемы по диагностике и регулировке силового оборудования, поставленного Campbell Scientific:

- тест напряжения аккумуляторной батареи критерий;
- тест цепи зарядки (если используется нерегулируемая солнечная панель);
- тест цепи зарядки (если используется трансформатор);
- настройка цепи зарядки.

Если узлы источника электропитания работают должным образом и система имеет периферийное оборудование с высоким потреблением тока, например спутниковый передатчик, проверьте, рассчитан ли источник питания на соответствующую мощность. Информацию по источникам питания Campbell Scientific можно найти на сайте *www.campbellsci.com*. Базовая информация приводится в приложении *Источники питания (стр. 657)*.

10.9.2. Диагностика источников электропитания — примеры — 8 10 30

Признак неисправности:

- программа CRBasic не выполняется;
- **параметр Low12VCount** таблицы **Status (Состояние)** имеет слишком большое значение.

Вероятное поврежденное оборудование:

- батареи;
- зарядные устройства/регуляторы;
- солнечные панели;
- трансформаторы.

Вероятные способы устранения:

- может потребоваться замена или подзарядка батарей;
- может потребоваться ремонт или перекалибровка зарядных устройств/регуляторов;
- может потребоваться ремонт или замена солнечных панелей.

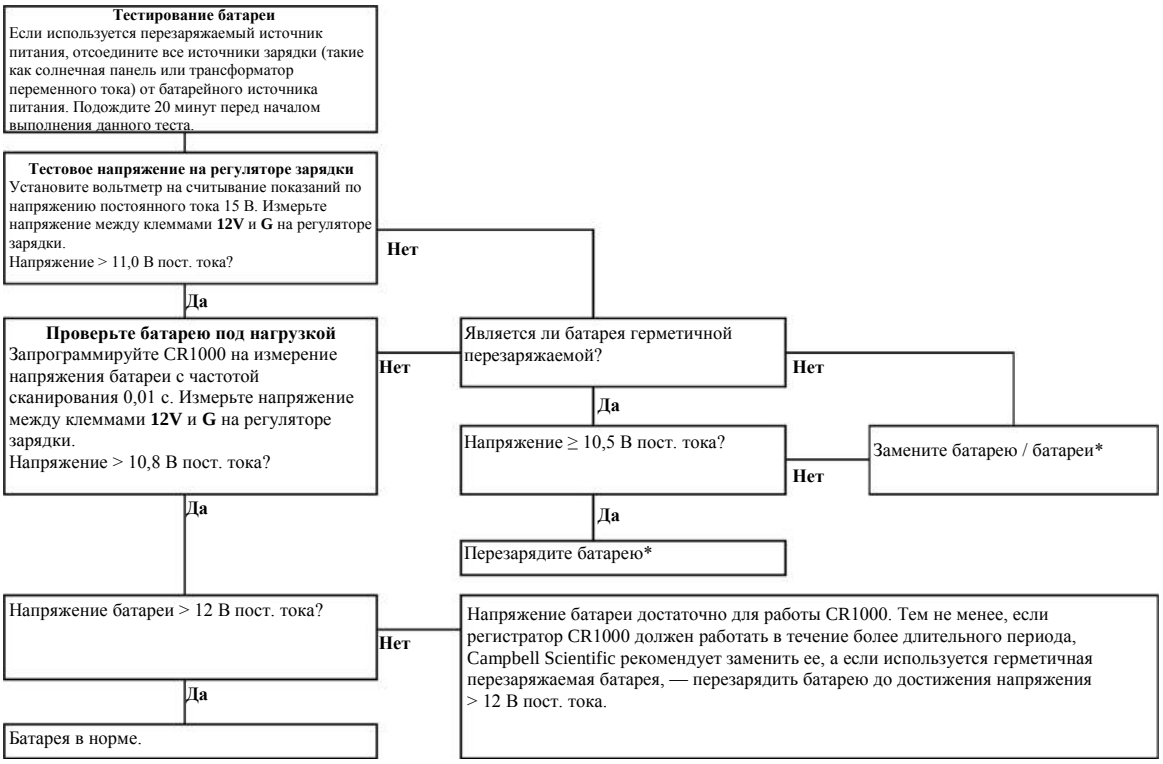
10.9.3. Диагностика источников электропитания — процедуры

Необходимое оборудование:

- вольтметр;
- резистор 5 кОм;
- резистор 50 Ом, 1 Вт для тестирования цепи зарядки и регулировки напряжения цепи зарядки.

10.9.3.1. Тестирование батареи

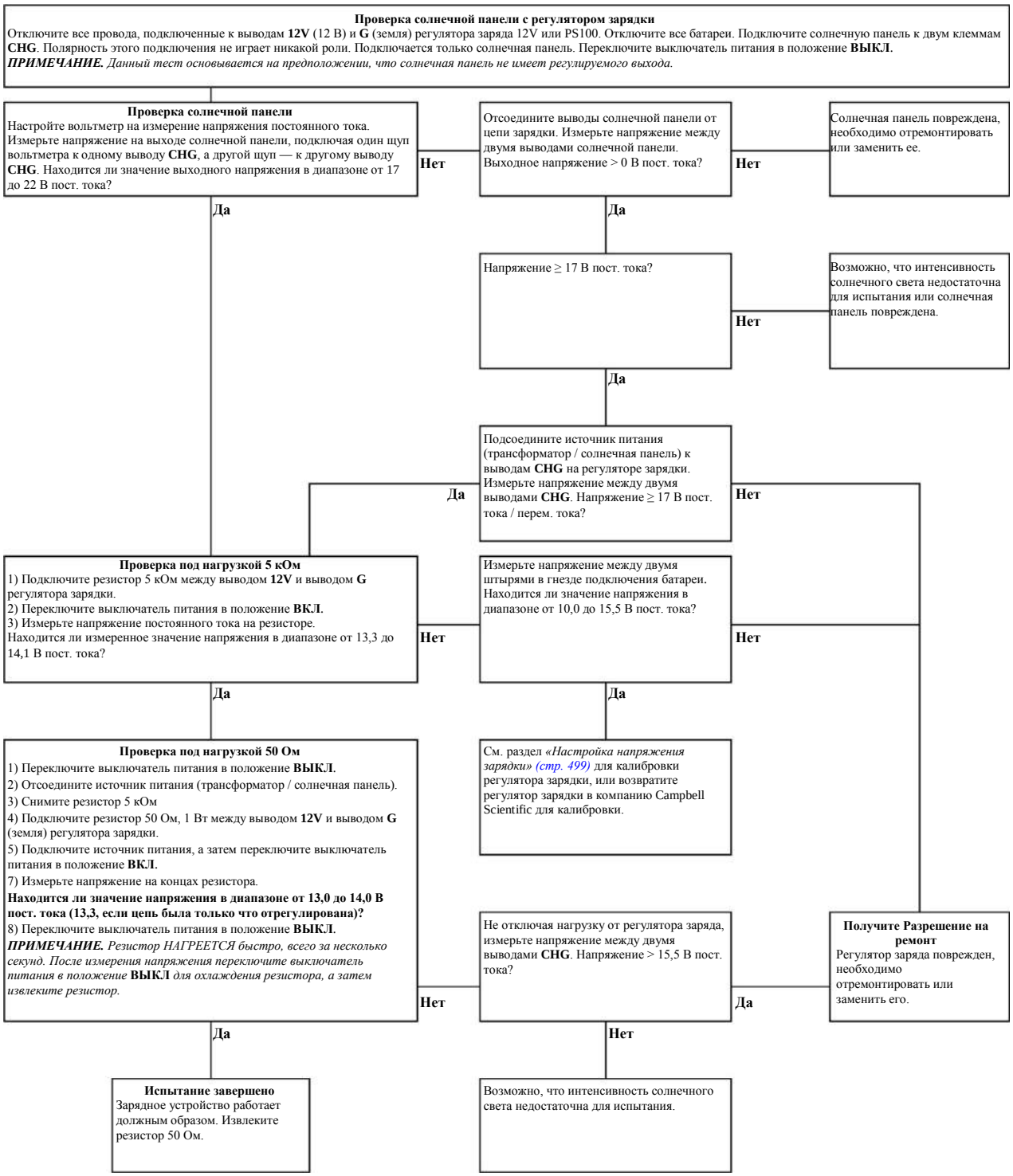
Рассмотренная в следующей блок-схеме процедура предназначена для тестирования герметичных перезаряжаемых или щелочных батарей, устанавливаемых в регуляторе зарядки PS100, или герметичных перезаряжаемых батарей, устанавливаемых в регуляторе зарядки CH100.
Если после выполнения процедуры требуется произвести ремонт, то см. раздел *«Гарантия и помощь»* *(стр. 3)* для получения информации о порядке отправки изделий в компанию Campbell Scientific.



*Если используется герметичная перезаряжаемая батарея, подзаряжаемая от основного источника питания, такого как солнечная панель или трансформатор переменного тока в переменный или переменного тока в постоянный, рекомендуется проверить регулятор зарядки. См. *Проверка солнечной панели с регулятором зарядки (стр. 496)* или *Проверка трансформатора с регулятором зарядки (стр. 498)*.

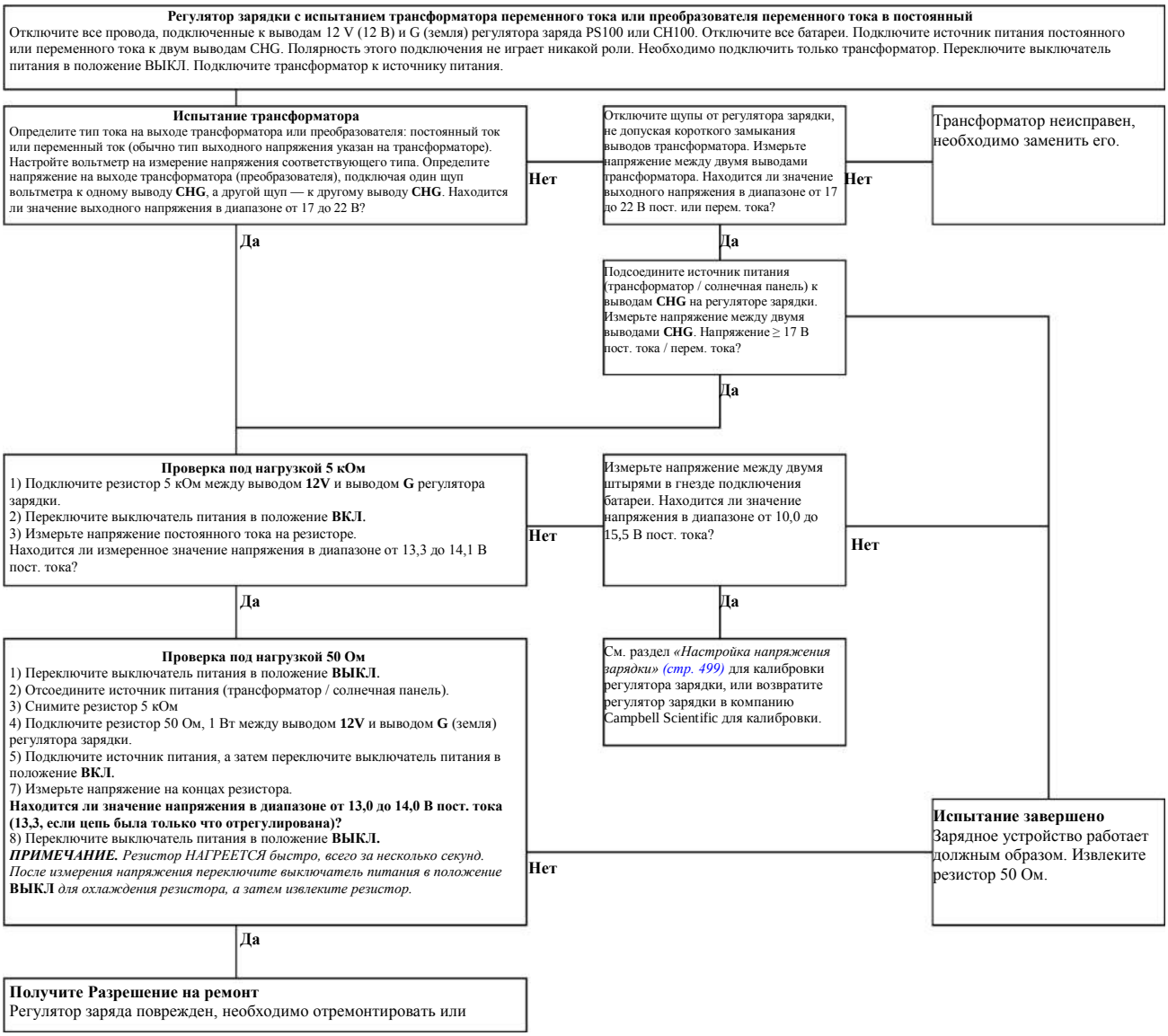
10.9.3.2. Проверка солнечной панели с регулятором зарядки

Описанная в этой схеме последовательности операций процедура предназначена для проверки регуляторов зарядки PS100 и CH100, в которых в качестве источника тока используется солнечная панель. Если после выполнения процедуры требуется произвести ремонт, то см. раздел *«Гарантия и помощь» (стр. 3)* для получения информации о порядке отправки изделий в компанию Campbell Scientific.


Проверка солнечной панели с регулятором зарядки
Отключите все провода, подключенные к выводам **12V** (12 В) и **G** (земля) регулятора заряда 12V или PS100. Отключите все батареи. Подключите солнечную панель к двум клеммам **CHG**. Полярность этого подключения не играет никакой роли. Подключается только солнечная панель. Переключите выключатель питания в положение **ВЫКЛ**.
ПРИМЕЧАНИЕ. *Данный тест основывается на предположении, что солнечная панель не имеет регулируемого выхода.*
Проверка солнечной панели
Настройте вольтметр на измерение напряжения постоянного тока. Измерьте напряжение на выходе солнечной панели, подключая один щуп вольтметра к одному выводу **CHG**, а другой щуп — к другому выводу **CHG**. Находится ли значение выходного напряжения в диапазоне от 17 до 22 В пост. тока?
Нет
Отсоедините выводы солнечной панели от цепи зарядки. Измерьте напряжение между двумя выводами солнечной панели. Выходное напряжение > 0 В пост. тока?
Нет
Солнечная панель повреждена, необходимо отремонтировать или заменить ее.
Да
Да
Напряжение ≥ 17 В пост. тока?
Нет
Возможно, что интенсивность солнечного света недостаточна для испытания или солнечная панель повреждена.
Да
Подсоедините источник питания (трансформатор / солнечная панель) к выводам **CHG** на регуляторе зарядки. Измерьте напряжение между двумя выводами **CHG**. Напряжение ≥ 17 В пост. тока / перем. тока?
Да
Нет
Проверка под нагрузкой 5 кОм
1) Подключите резистор 5 кОм между выводом **12V** и выводом **G** регулятора зарядки.
2) Переключите выключатель питания в положение **ВКЛ**.
3) Измерьте напряжение постоянного тока на резисторе.
Находится ли измеренное значение напряжения в диапазоне от 13,3 до 14,1 В пост. тока?
Нет
Измерьте напряжение между двумя штырями в гнезде подключения батареи. Находится ли значение напряжения в диапазоне от 10,0 до 15,5 В пост. тока?
Нет
Да
Да
Проверка под нагрузкой 50 Ом
1) Переключите выключатель питания в положение **ВЫКЛ**.
2) Отсоедините источник питания (трансформатор / солнечная панель).
3) Снимите резистор 5 кОм
4) Подключите резистор 50 Ом, 1 Вт между выводом **12V** и выводом **G** (земля) регулятора зарядки.
5) Подключите источник питания, а затем переключите выключатель питания в положение **ВКЛ**.
7) Измерьте напряжение на концах резистора.
Находится ли значение напряжения в диапазоне от 13,0 до 14,0 В пост. тока (13,3, если цепь была только что отрегулирована)?
8) Переключите выключатель питания в положение **ВЫКЛ**.
ПРИМЕЧАНИЕ. *Резистор НАГРЕЕТСЯ быстро, всего за несколько секунд. После измерения напряжения переключите выключатель питания в положение* **ВЫКЛ** *для охлаждения резистора, а затем извлеките резистор.*
См. раздел *«Настройка напряжения зарядки» (стр. 499)* для калибровки регулятора зарядки, или возвратите регулятор зарядки в компанию Campbell Scientific для калибровки.
Нет
Не отключая нагрузку от регулятора заряда, измерьте напряжение между двумя выводами **CHG**. Напряжение > 15,5 В пост. тока?
Да
Получите Разрешение на ремонт
Регулятор заряда поврежден, необходимо отремонтировать или заменить его.
Да
Нет
Испытание завершено
Зарядное устройство работает должным образом. Извлеките резистор 50 Ом.
Возможно, что интенсивность солнечного света недостаточна для испытания.

10.9.3.3. Регулятор зарядки с испытанием трансформатора

Описанная в этой схеме последовательности операций процедура предназначена для проверки регуляторов зарядки PS100 и CH100, в которых в качестве источника тока используется трансформатор переменного тока или преобразователь переменного тока в постоянный. Если после выполнения процедуры требуется произвести ремонт, то см. раздел *«Гарантия и помощь» (стр. 3)* для получения информации о порядке отправки изделий в компанию Campbell Scientific.



10.9.3.4. Настройка зарядного напряжения

Примечание. Компания Campbell Scientific рекомендует обеспечить, чтобы следующую процедуру выполнял квалифицированный электрик.

Описанная в этой схеме последовательности операций процедура предназначена для проверки и настройки регуляторов зарядки PS100 и CH100. Если после выполнения процедуры требуется произвести ремонт или калибровку, то см. раздел *«Гарантия и помощь» (стр. 3)* для получения информации о порядке отправки изделий в компанию Campbell Scientific.

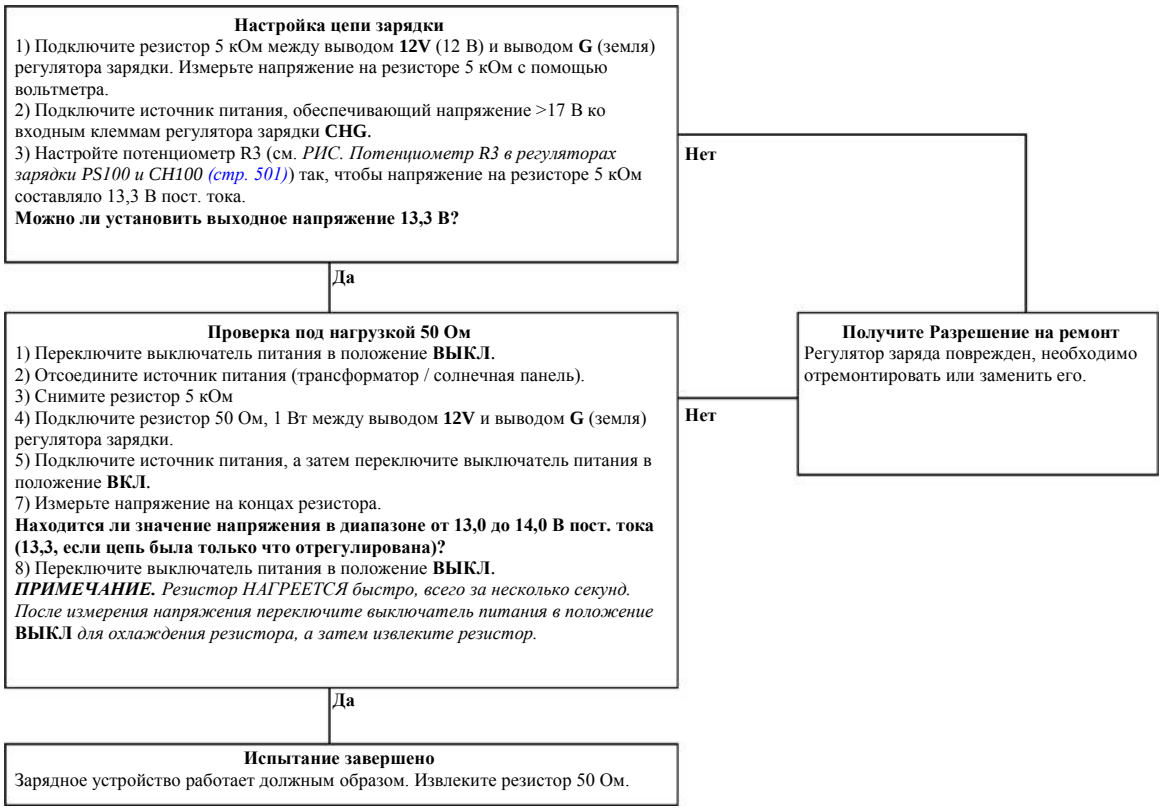
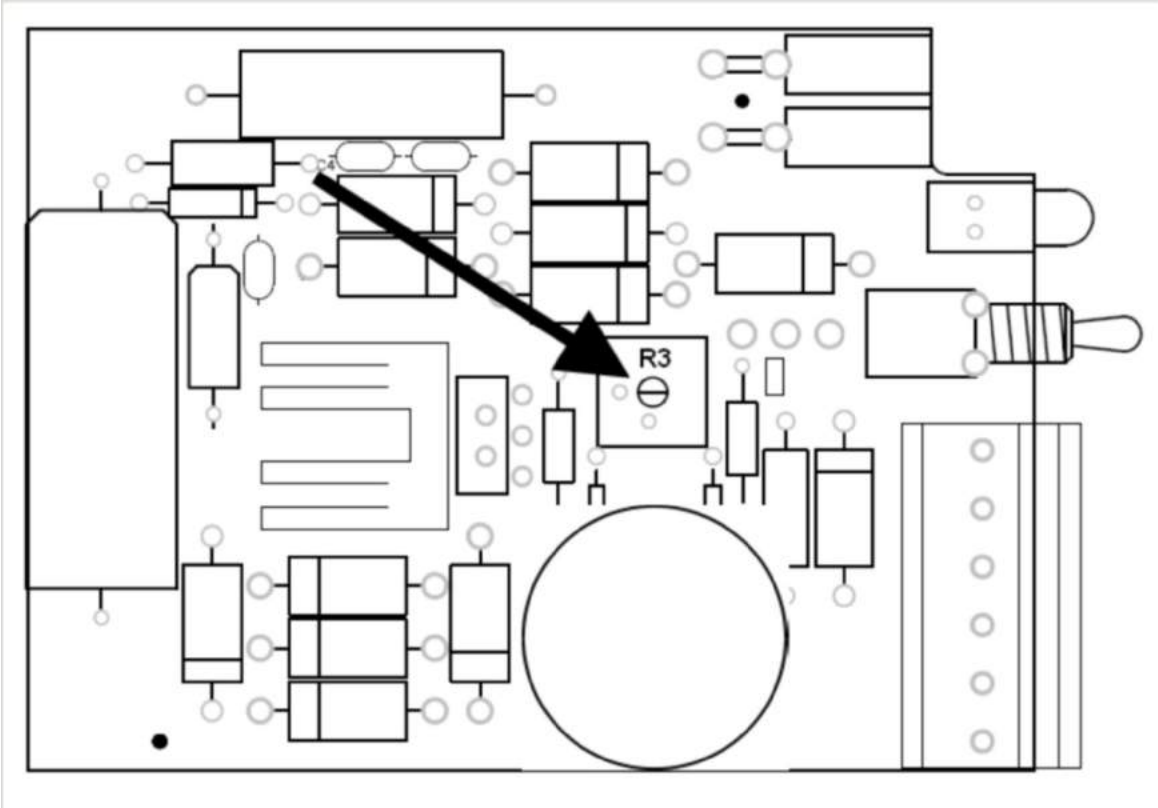

Настройка цепи зарядки
1) Подключите резистор 5 кОм между выводом 12V (12 В) и выводом G (земля) регулятора зарядки. Измерьте напряжение на резисторе 5 кОм с помощью вольтметра.
2) Подключите источник питания, обеспечивающий напряжение >17 В ко входным клеммам регулятора зарядки CHG.
3) Настройте потенциометр R3 (см. РИС. Потенциометр R3 в регуляторах зарядки PS100 и CH100 (стр. 501)) так, чтобы напряжение на резисторе 5 кОм составляло 13,3 В пост. тока.
Можно ли установить выходное напряжение 13,3 В?
Нет
Да
Проверка под нагрузкой 50 Ом
1) Переключите выключатель питания в положение ВЫКЛ.
2) Отсоедините источник питания (трансформатор / солнечная панель).
3) Снимите резистор 5 кОм
4) Подключите резистор 50 Ом, 1 Вт между выводом 12V и выводом G (земля) регулятора зарядки.
5) Подключите источник питания, а затем переключите выключатель питания в положение ВКЛ.
7) Измерьте напряжение на концах резистора.
Находится ли значение напряжения в диапазоне от 13,0 до 14,0 В пост. тока (13,3, если цепь была только что отрегулирована)?
8) Переключите выключатель питания в положение ВЫКЛ.
ПРИМЕЧАНИЕ. Резистор НАГРЕЕТСЯ быстро, всего за несколько секунд. После измерения напряжения переключите выключатель питания в положение ВЫКЛ для охлаждения резистора, а затем извлеките резистор.
Нет
Получите Разрешение на ремонт
Регулятор заряда поврежден, необходимо отремонтировать или заменить его.
Да
Испытание завершено
Зарядное устройство работает должным образом. Извлеките резистор 50 Ом.

Рисунок 131. Потенциометр R3 в зарядном устройстве/регуляторе PS100 или CH100



10.10. Режим терминала

В таблице *Команды терминала CR1000* *(стр. 502)* приведен список опций режима терминала. За исключением только команды C, для обычной работы с терминалом CR1000 опции терминала не требуются.

Для перехода в режим терминала подключите ПК к CR1000 с помощью последовательного кабеля, применявшегося в разделе *«Что потребуется» (стр. 46)*. Запустите программу-эмулятор. Доступны следующие программы-эмуляторы:

- Программное обеспечение регистратора данных Campbell Scientific *(стр. 95): окно Terminal Emulator (стр. 530)* ;
- *DevConfig* (утилита Campbell Scientific *Device Configuration Utility*): закладка **Terminal**;
- HyperTerminal (Гипертерминал). Начало работы в ОС Windows Vista. Необходимо приобрести и установить HyperTerminal (или другую программу эмуляции терминала).

Как показано на рис. *Закладка DevConfig Terminal* *(стр. 503)*, после входа в программу эмуляции терминала нажмите несколько раз **ВВОД** до получения приглашения **CR1000>**. Команда терминала состоит из одного символа, она завершается клавишей **ВВОД**. Если передать **H** и нажать **ВВОД**, то возвращается меню эмулятора терминала.

Для прерывания текущей команды нажмите клавишу ESC или дождитесь окончания тайм-аута (40 секунд). Одновременные сеансы не поддерживаются, попытка организации нескольких одновременных сеансов приводит к потере связи.

Таблица 133. Команды терминала CR1000

Опция	*Описание*	*Использование*
0	Время обработки сканирования; реальное время в секундах	Выводит технические данные циклов сканирования программы.
1	Дамп данных Serial Flash (флеш-памяти с доступом по последовательному интерфейсу)	Инструментальное средство Campbell Scientific.
2	Прочитать ИС часов	Вывести список двоичных данных, связанных с ИС часов CR1000.
3	Состояние	Вывести таблицу **Status** (Состояние) CR1000.
4	Состояние карты и ошибки компиляции	Выводит технические данные для установленной карты CF.
5	Информация сканирования	Технические данные по циклу сканирования CR1000.
6	Исходные данные для преобразования аналогового сигнала в цифровой	Технические данные по преобразованию аналоговых данных в цифровые.
7	VARS (Переменные)	Выводит список переменных таблицы **Public**.
8	Приостановить/начать вывод данных	Вывести все данные таблицы. Не рекомендуется применять эту команду для накопления данных, особенно при использовании связи. Формат дампа данных — ASCII без проверки ошибок.
9	Прочитать двоичные данные inloc	Выводит в виде двоичных данных таблицу **Public**.
A	Авторские права на операционную систему	Выводит сведения об авторских правах и версию операционной системы.
B	Коды операций устройства определения последовательности	Технические данные устройства определения последовательности выполнения заданий.
C	Внести изменения в таблицу констант	Редактировать константы, определенные в таблице **ConstTable / EndConstTable.** Работает только при наличии таблицы **ConstTable / EndConstTable** в активной программе.
D	Монитор задач MTdbg()	Инструментальное средство Campbell Scientific.
E	Ошибки компиляции	Выводит список ошибок компиляции для текущей попытки загрузки программы.
F	VARS без имен	Инструментальное средство Campbell Scientific.
G	Дамп памяти serial flash (флеш-памяти с доступом по последовательному интерфейсу) ЦП	Инструментальное средство Campbell Scientific.
H	Меню эмулятора терминала	Выводит главное меню.
I	Данные калибровки	Выводит коэффициенты усиления и смещения, полученные при внутренней калибровке схемы аналоговых измерений.
J	Вывести дамп файла.	Передает текст текущей программы, включая комментарии.
K	Не используется	
L	Читать шину периферийных устройств	Инструментальное средство Campbell Scientific.
M	Проверка памяти	Вывести результаты проверки памяти.
N	Информация о файловой системе	Выводит список файлов, хранящихся в памяти CR1000.
O	Данные по длине таблицы	Выводит технические данные по длине таблицы данных.
P	Serial talk through (Последовательная ретрансляция)	Генерирует с клавиатуры команды, которые передаются в подключенное устройство через последовательный порт устройства регистрации данных. Похожа на команду SDI12 Talk Through.
REBOOOT (ПЕРЕЗАГРУЗКА)	Повторная компиляция программы	После ввода последней буквы «T» команды “REBOOT” немедленно снова компилируется программа CR1000. Данные таблицы сохраняются в памяти. **ПРИМЕЧАНИЕ.** При вводе команды **REBOOT** эхо символов отключено (они не выводятся на экран терминала).

Таблица 133. Команды терминала CR1000

Опция	Описание	Использование
SDI12	Ретрансляция через порт SDI12	Генерирует с клавиатуры команды, которые передаются в подключенное устройство через порт SDI-12 CR1000. Похожа на команду Serial Talk Through.
T	Не используется	
U	Восстановление данных	Предоставляет средства восстановления данных, потерянных при загрузке новой программы. Детальное описание см. в разделе *«Поиск и устранение неисправностей — Восстановление данных»* *(стр. 504)*.
V	Дамп памяти низкого уровня	Инструментальное средство Campbell Scientific.
W	Comms watch (Просмотр трафика)	Ведет мониторинг трафика CR1000.
X	Идентификатор модуля шины периферийных устройств	Инструментальное средство Campbell Scientific

Рисунок 132. Закладка DevConfig **Terminal**

10.10.1. Последовательная ретрансляция и просмотр трафика

В режимах **P: Serial Talk Through** и **W: Comms Watch**: тайм-аут можно изменить на любое значение от 1 до 86 400 секунд (86 400 с = 1 сутки), значение по умолчанию составляет 40 с.

При использовании команд **P** и **W** в ходе сеанса учтите следующее:

- CR1000 не поддерживает одновременные сеансы.
- При запуске нового сеанса текущий сеанс завершится.
- CR1000 попытается запустить сеанс при получении символов, не входящих в PakBus, по девятипроводному порту **RS-232** или порту **CS I/O**, если только порт не был ранее открыт командой **SerialOpen()**.

Если CR1000 попытается начать сеанс по девятипроводному порту **RS-232** или порту **CS I/O** в результате приема символа, не входящего в PakBus, и этот порт не был открыт командой **SerialOpen()**, то немедленно прервутся все функции, выполняемые терминалом. Поэтому при использовании программ, часто открывающих и закрывающих последовательный порт, выше вероятность получения в закрытый последовательный порт символа, не соответствующего PakBus, что приведет к завершению текущих сеансов ретрансляции (talk-through) и контроля трафика (comms watch). В данном случае настройка **FileManager()**, позволяющая записать трафик из comms watch (контроля трафика) или sniffer в файл, не предоставляет возможность решить эту проблему.

10.11. Журналы

Журналы и метаданные обычно относятся к регистратору данных или работе ПО. Если журналы включены, то они доступны по адресам, приведенным ниже в таблице.

Таблица 134. Местоположения журналов

Пакет ПО	*Обычные местоположения журналов*
LoggerNet	C:\Campbellsci\LoggerNet\Logs
PC400	C:\Campbellsci\PC400\Logs
DevConfig	C:\Campbellsci\DevConfig\sys\cora\Logs

10.12. Поиск и устранение неисправностей — восстановление данных

Изредка могут потребоваться дополнительные действия для восстановления данных, которые в противном случае будут потеряны при использовании обычных методов сбора данных. Такие обстоятельства могут включать следующие:

- Ошибка управления программой
  - Программа на CRBasic была передана в CR1000 без указания того, что она должна запускаться после включения питания. Скорее всего это может случиться при использовании возможности **Compile, Save and Send** в старых версиях редактора *CRBasic Editor*.
  - Новая программа (возможно, что та же программа) была случайно передана в CR1000 с помощью клиента *Connect* или клиента *Set Up* в *LoggerNet*.
  - Программа была остановлена, используя ПО устройства регистрации **File Control** или ПО *LoggerLink*.
- Процессор: драйвер был случайно удален.
- Сетевое периферийное устройство (NL115, NL120, NL200 или NL240) было добавлено к CR1000 в случае отсутствия сетевого периферийного устройства, это привело к перераспределению памяти CR1000.
- Аппаратный отказ, например, нарушение целостности информации в памяти.
- Вставка и извлечение карты памяти обычно не приводит к потере данных в CR1000. Такие события изменяют определения таблиц, поскольку они могут изменить длину таблицы, но они не создают ситуации, требующей восстановления данных.

Обычно удается восстановить данные, используя мастера **Datalogger Data Recovery** в *DevConfig* *(стр. 111)*. Восстановление возможно, поскольку данные обычно не уничтожаются, а только теряются ссылки на них. При этом мастер восстанавливает «данные» во всей памяти независимо от того, производилась ли запись в эту память и производилась ли запись в эту память недавно.

При выполнении процедуры восстановления имейте в виду следующее:

Если накопитель CRD: (карта памяти) или накопитель USB: (запоминающее устройство Campbell Scientific) извлекался после первоначальной записи данных, то выполняется процедура **Datalogger Data Recovery**, поэтому скорее всего указатель памяти будет иметь неверное значение, и восстановленные данные будут повреждены. В этом случае возвратите накопитель CRD: или USB: в его место и снова запустите мастера **Datalogger Data Recovery**, а затем снова запустите программу на CRBasic.

В любом случае, даже если восстановление будет произведено правильно, в

полученном результате в правильных данных присутствуют пустые секции или секции со старыми данными. Поскольку все данные записаны в один файл, можно отсортировать правильные и неправильные данные.

11. Глоссарий

11.1. Термины

Термин ac (перем. ток)

См. Vac (стр. 532).

Термин: погрешность

Мера правильности измерения. Также см. приложение *Погрешность, точность и разрешение (стр. 533)*.

Термин: А/Ц-преобразование

Аналогово-цифровое преобразование. Процесс, преобразующий уровень аналогового напряжения в цифровое значение.

Термин: ампер (А)

Основная единица изменения силы тока. Она используется для количественного описания источника тока или требования к потребляющему ток устройству.

Термин: аналоговые

Данные, представленные в виде непрерывно изменяющихся электрических сигналов.

Термин: аргумент

Параметр (стр. 523): часть определения процедуры (или команды).

Аргумент (стр. 507): часть вызова процедуры (или выполнения программы). Аргумент вставляется в качестве параметра. Например, в команде CRBasic **Battery(*dest*) *dest*** = это часть параметра, определяющего, какой аргумент должен вставляться на его место в программе на CRBasic. Если переменная с именем ***BattV*** должна содержать результат измерения батареи функцией **Battery(), то *BattV*** — это аргумент, вставляемый вместо параметра ***dest***. В операторе

```
Battery(BattV)
```

BattV является аргументом.

Термин: ASCII / ANSI

Подробнее:
• *Термин: ASCII/ANSI (стр. 507)*
• *Таблица ASCII/ANSI (стр. 637)*

Аббревиатура, означающая «American Standard Code for Information Interchange / American National Standards Institute» (Американский стандартный код для обмена информацией, Американским национальный институт стандартов). Схема кодирования, в которой числа от 0 до 127 (ASCII) или от 0 до 255 (ANSI) используются для обозначения предопределенных алфавитно-цифровых символов. Обычно каждое число хранится и передается в виде 8 двоичных цифр (8 битов), то есть для хранения символа текста требуется 1 байт.

Термин: асинхронный

Передача данных между передающим и принимающим устройством происходит в виде последовательности нулей и единиц. Для корректного «считывания» данных принимающее устройство должно начинать считывание в правильной точке последовательности. При асинхронной связи для этого перед каждым символом и после него вставляется один или несколько стартовых и стоповых битов, которые означают начало и конец информации (см. *синхронная (стр. 530)*).

Указывает на отсутствие синхронизации передающего и принимающего устройств посредством тактового сигнала.

Термин: AWG

AWG («калибр») — общепринятый способ обозначения диаметра провода. Чем больше значение AWG, тем меньше диаметр провода. Чем меньше значение AWG, тем больше диаметр провода. Например, провод 14 AWG часто используют для заземления, потому что по нему могут протекать сильные токи. Провод 22 AWG часто используют как провод датчика, поскольку при измерениях используются только слабые токи.

Термин: скорость передачи в бодах

Скорость, с которой передаются данные.

Термин: маркер

Сигнал, передаваемый другим устройствам в сети PakBus®, для определения «соседних» устройств. В сети PakBus маркеры гарантируют, что все устройства в сети знают о жизнеспособности других устройств. Можно настроить передачи команды настройки часов вместе с маркером. Эту функцию можно использовать для синхронизации часов устройств в сети PakBus. См. также *PakBus (стр. 522)* и *Соседнее устройство (стр. 521)*.

Термин: двоичный

Означает данные, представленные последовательностями нулей и единиц. Также описывает состояние выключателя (Вкл. или Выкл.)

Термин: BOOL8

Тип данных длиной 1 бит, содержащих 8 битов (0 или 1) информации. Для типа данных BOOL8 требуется меньше места, чем для типа данных BOOLEAN, имеющего длину 32 бита.

Термин: двоичный

Часть имени функции, ее результатом может быть true (истина) или false (ложь).

Термин: тип данных boolean (логический)

Обычно используется для флага и представления состояния или оборудования, имеющего два состояния (true (истина) или false (ложь), например, флаг или управляющий порт.

Термин: пачка

Означает пачку измерений. Это аналог вспышки света, пачка измерений является интенсивной, в ней измерения быстро следуют одно за другим, но они не являются непрерывными.

Термин: мастер калибровки

Мастер калибровки упрощает использование команд калибровки в поле CRBasic **FieldCal()** и **FieldCalStrain()**. Он встроен в ПО *LoggerNet* (4.0 и выше) и *RTDAQ*.

Термин: обратный вызов

Это название процесса, в котором CR1000 инициирует связь с ПК, на котором работает соответствующее ПО *устройства регистрации компании Campbell Scientific* *(стр. 654)*. Другое название «Initiate Telecommunications» (Начать связь).

Термин: ПО CardConvert

Утилита, выбирающая окончательные данные из памяти CR1000 с карты Compact Flash (CF) и преобразующая эти данные в код ASCII или другой формат.

Термин: CD100

Необязательный дисплей с клавиатурой в корпусе, используемый совместно с устройствами регистрации CR1000. См. приложение *«Дисплей с клавиатурой — список»* *(стр. 651)*.

Термин: CDM/CPI

CPI — это фирменный интерфейс, используемый устройствами регистрации Campbell Scientific для связи с периферийными устройствами Campbell Scientific CDM. Он состоит из определения физического уровня и протокола обмена данными. Устройства CDM и устройства Campbell Scientific SDM близки по своим принципам, но шина CPI позволяет повысить скорость передачи данных и использовать более длинные кабели. Обычно устройства CDM потребляют при работе большую мощность, чем устройства SDM.

Термин: CF

См. *CompactFlash* *(стр. 510)*.

Термин: код

Программа на языке CRBasic или часть такой программы.

Термин: кнопка Collect / Collect Now

Кнопка или команда в ПО регистратора данных, обеспечивающая сбор по запросу окончательных данных из памяти. Эта возможность поддерживается в ПО *PC200W, PC400, LoggerNet* и *RTDAQ*.

Термин: COM-порт

COM — это общее название физических и виртуальных последовательных портов.

Термин: CompactFlash

CompactFlash® (CF) — это технология карты памяти, которая используется в некоторых модулях памяти Campbell Scientific. CompactFlash®— это зарегистрированный товарный знак CompactFlash® Association.

Термин: команды ввода-вывода

Обычно относится к команде CRBasic.

Термин: командная строка

Одна строка в программе на CRBasic. Максимальная длина строки составляет 512 символов, даже если эта строка содержит символы продолжения <пробел> <подчеркивание> (_). Обычно командная строка содержит один оператор программы, но она может содержать несколько операторов, разделенных <двоеточием> (:).

Термин: компилировать

Программный процесс преобразования человекочитаемого кода программы в двоичный машинный код. Программы пользователя CR1000 компилируются в операционной системе CR1000.

Термин: обработанные выходные данные

Выходные данные датчика с учетом коэффициентов масштабирования. См. *необработанные выходные данные* *(стр. 531)*.

Термин: разъем

Разъем — это устройство, которое обеспечивает совместное соединение и разъединение одного или нескольких проводников тока (проводов, кабелей, выводов и т. д.). Разъем состоит из двух частей — вилки и гнезда. Например, обычная розетка переменного тока в комнате, представляет собой гнездовую часть разъема. Вилка на конце провода лампы — это вилочная часть разъема. См. *вывод* *(стр. 530)*.

Термин: константа

Часть устройства памяти CR1000, которому присвоено алфавитно-цифровое имя и фиксированное число.

Термин: управление вводом-выводом

Выводы типа C, настроенные для управления устройством или контроля устройства.

Термин: CoraScript

CoraScript — это интерпретатор командной строки, используемый в ПО регистратора данных *LoggerNet*. Дальнейшая информация приведена в руководстве по *LoggerNet*, которое можно скачать по адресу *www.campbellsci.com*.

Термин: ЦП

Центральный процессор Мозг CR1000. Кроме того, он означает две следующие области памяти:

- CPU: накопитель
- Память, используемая ЦП для хранения данных таблицы

Термин: CR1000KD

Необязательный портативный дисплей с клавиатурой, используемый совместно с устройством регистрации CR1000. См. приложение *«Дисплей с клавиатурой — список»* *(стр. 651)*.

Термин: cr

Разрыв строки

Термин: Команда программы CRBasic Editor: Compile, Save and Send

Команда меню *CRBasic Editor*, которая компилирует программу, сохраняет ее и передает ее в устройство регистрации.

Термин: CRD

Необязательный накопитель, в котором используется карта памяти. См. *CompactFlash* *(стр. 510)*.

Термин: CS I/O

Фирменный порт ввода-вывода Campbell Scientific. Кроме того, это фирменный протокол последовательной передачи данных через порт **CS I/O**.

Термин: CVI

Интервал между сигналами подтверждения связи. Интервал времени, через который устройство PakBus® проверяет доступность соседей, включенных в его список соседей. Если сосед не отвечает за период времени, превышающий CVI в 2,5 раза, то устройство передает до 4 сигналов **Hello** (Алло). Если ответ не получен, то соответствующий сосед удаляется из списка соседей. Для получения более подробной информации см. раздел *«PakBus — общие сведения»* *(стр. 88)*.

Термин: кэш данных

Кэш данных — это набор двоичных файлов, хранящихся на жестком диске компьютера, в котором работает *ПО регистратора данных* *(стр. 512)*. Двоичный файл создается для каждой таблицы в каждом регистраторе данных. Эти файлы моделируют память регистратора данных; по умолчанию их длина вдвое превышает область хранения устройства регистрации. Когда ПО собирает данные из CR1000, эти данные записываются в двоичный файл для данного CR1000. Различные программные функции читают данные из кэша данных, а не прямо из CR1000. Это позволяет различным функциям ПО использовать эти данные одновременно.

Аналогично окончательным таблицам данных CR1000, двоичные файлы для кэша данных по умолчанию организованы как *кольцевая память* *(стр. 526)*.

Термин: ПО регистратора данных

ПО Campbell Scientific, содержащее как минимум следующие функции:

- Связь регистратора данных
- Загрузка программ
- Настройка часов
- Выборка данных измерений

Более полное описание приведено в разделе *«ПО регистратора данных — общие сведения»* *(стр. 95)* и в приложении *«ПО регистратора данных — список»* *(стр. 654)*.

Термин: точка данных

Значение данных, переданное в *память окончательных данных* *(стр. 515)* в результате выполнения *команды обработки выходных данных* *(стр. 512)*. Одновременно строки выхода точек данных образуют запись в таблице данных.

Термин: таблица данных

Способ организация данных в памяти CR1000 или в файлах, получаемых при сборе данных из памяти CR1000. Программа на CRBasic создает фундаментальную таблицу данных в результате выполнения команды **DataTable()**, эта таблица находится в двоичном виде в основной памяти в статическом ОЗУ. См. таблицу *Выделение памяти в CR1000* *(стр. 371)*. Кроме того, структура таблиц данных также содержится в *кэше данных* *(стр. 511)*, в отдельных файлах данных в ЦП (CPU:), в накопителях USR:, CRD: и USB:, и в двоичных или ASCII-файлах, полученных после сбора памяти окончательных данных *ПО устройства регистрации* *(стр. 512)*.

Термин: интервал вывода данных

Альтернативное название: интервал вывода

Интервал между последовательными *записями* *(стр. 525)* в окончательной таблице данных в памяти.

Термин: команды обработки выходных данных

Команды языка CRBasic, которые обрабатывают значения данных для окончательного вывода в память окончательных данных. Примеры команд обработки вывода: **Totalize(), Maximize(), Minimize()** и **Average().** Источники данных для этих команд — это значения или строки, хранящиеся в памяти переменных. Результаты промежуточных расчетов хранятся в *памяти для обработки выходных данных* *(стр. 512)*, они ожидают срабатывания выходного триггера. Окончательное расположение данных, сформированных командами обработки вывода, обычно представляет собой память окончательных данных, но программа на CRBasic может направить их в память переменных для дальнейшей обработки. Передача в память с обработанными окончательными данными производится, когда аргумент ***Trigger*** в команде **DataTable()** установлен равным **True** (Истина).

Термин: память обработанных выходных данных

Память статического ОЗУ автоматически выделяется для промежуточных расчетов, выполняемых командами обработки выходных данных CRBasic. Невозможно осуществлять мониторинг память для обработанных выходных данных. Список команд, использующих память для обработанных выходных данных, приведен в разделе *«Обработка для вывода в память окончательных данных»* *(стр. 542)* .

Термин: dc (пост. ток)

См. *Vdc (стр. 532)*.

Термин: DCE

Data **C**ommunication **E**quipment (Аппаратура передачи данных). Этот термин имеет более широкое значение, но в ограниченном контексте практического использования CR1000 он означает, назначение штырьков, тип электрического разъема и функции в порту RS-232. Порт RS-232 в CR1000 представляет собой DCE. Для соединения устройства DCE с устройством DCE используется нуль-модемный кабель. См. *Термин: DTE (стр. 514)*.

Термин: осушитель

Гигроскопический материал, поглощающий влагу из окружающего воздуха. Если поместить его внутри изолированного объема, например, внутри корпуса регистратора данных, то он предотвращает конденсацию влаги.

Термин: ПО DevConfig

Утилита конфигурации устройства (стр. 111), включенная в ПО *LoggerNet, RTDAQ, PC400, ее также можно загрузить с адреса www.campbellsci.com/downloads* (*http://www.campbellsci.com/downloads*).

Термин: DHCP

Dynamic Host Configuration Protocol (Протокол динамической настройки хостов). Протокол прикладного уровня в семействе протоколов TCP/IP.

Термин: дифференциальный

Вывод датчика или измерительный вывод, если напряжение аналогового сигнала передается по двум выводам. Результат измерения пропорционален разности потенциалов на двух выводах.

Термин: Dim

Команда CRBasic, используемая для объявления переменных и их размерности. Переменные, объявленные командой **Dim**, являются скрытыми в течение команд устройства регистрации.

Термин: dimension

Команда. Используется для написания программы на CRBasic для массива переменной длины, как показано в следующих примерах:

- Команда **DIM *example(3)*** создает три переменные ***example(1), example(2),*** и ***example(3).***
- Команда **DIM *example(3,3)*** создает девять переменных.
- Команда **DIM *example(3,3,3)*** создает 27 переменных.

Термин: DNS

Domain name system (Система имен доменов). Протокол прикладного уровня в семействе протоколов TCP/IP.

Термин: DTE

Data **T**erminal **E**quipment (Оконечное оборудование данных). Этот термин имеет более широкое значение, но в ограниченном контексте практического использования CR1000 он означает назначение штырьков, тип электрического разъема и функции в порту RS-232. Порт RS-232 в CR1000 представляет собой DCE. Если подключить устройство DCE с помощью нуль-модемного кабеля, то фактически получается устройство DTE. См. *Термин: DCE (стр. 513)*.

Термин: дуплексный

Протокол последовательной передачи данных. Последовательная связь может быть симплексной, полудуплексной или дуплексной.

Подробнее: *симплексная (стр. 528)*, *дуплексная (стр. 248)*, *полудуплексная (стр. 517)* и *полностью дуплексная (стр. 516)*.

Термин: коэффициент использования

Процент времени работы, когда устройство находится в рабочем состоянии. Например, если CR1000 запрограммировано на интервал сканирования 1 секунда, а программа завершает работу всего через 100 миллисекунд, то говорят, что коэффициент использования составляет 10 %.

Термин: заземление

Заземлитель или другое подходящее устройство, обеспечивающее подключение системы или устройства к земле. Заземление — это сток для переходных электрических сигналов и потенциально повреждающих потенциалов, создаваемых, например, близким ударом молнии. Заземление является предпочтительным опорным потенциалом для аналоговых измерений напряжения. Обратите внимание на то, что большинство объектов имеют «электрический потенциал», причем потенциал в различных точках земли, даже удаленных друг от друга всего на несколько метров, может быть различным.

Термин: технические единицы измерения

Единицы измерения, которые явно описывают явления, в противоположность, например, основной единице аналогового измерения CR1000 — милливольту.

Термин: устойчивость к электростатическим разрядам

Разряд статического электричества

Термин: ESS

Станция датчиков параметров окружающей среды

Термин: возбуждение

Приложение точного напряжения, обычно к мосту сопротивления.

Термин: интервал выполнения

См. *Интервал сканирования (стр. 526)*.

Термин: время выполнения

Время, необходимое для выполнения команды или группы команд. Если время выполнения программы превышает **Scan()** ***Interval***, то программа выполняется реже, чем запрограммировано, при этом в таблице состояния **Status** будет увеличено значение регистра **SkippedScan** *(стр. 487)* .

Термин: выражение

Последовательность слов, операторов и чисел, которая дает значение или результат.

Термин: FFT

Fast **F**ourier **T**ransform (Быстрое преобразование Фурье). Способ анализа частотного спектра

Термин: File Control (Управление файлами)

File Control — это возможность ПО регистратора данных *LoggerNet, PC400 и RTDAQ* *(стр. 95)*. Она обеспечивает представление файловой системы CR1000, а также меню команд управления файлами:

Команда Delete вызывает удаление указанного файла.

Команда Send вызывает перенос файла (обычно файла программы на языке CRBasic) из памяти ПК в память CR1000.

Команда Retrieve вызывает выгрузку совокупности файлов, выведенных командой **File Control**. *При выгрузке файла данных из карты CF командой* **Retrieve** *необходимо предварительно остановить программу CR1000, иначе возможно повреждение целостности данных.*

Команда Format форматирует выбранное запоминающее устройство CR1000. Будут удалены все файлы в этом устройстве, включая файлы с данными.

Термин: закладка File Retrieval (Выборка файла)

Возможность *LoggerNet Setup Screen*. В карте сети *Setup Screen* (Entire Network (Вся сеть)) щелкните узел регистратора данных CR1000. Вкладка **File Retrieval** должна быть одной из нескольких вкладок, открытых в правой части экрана.

Термин: заполнить память и остановиться (fill and stop memory)

Конфигурация памяти для таблиц данных, вызывающая прекращение приема данных после заполнения таблицы.

Термин: память окончательных данных (final-data memory)

Часть статического ОЗУ CR1000, выделенная для хранения таблиц данных с выходными массивами. Если данные уже записаны в окончательную память, то их невозможно изменить, их можно только перезаписать, если они стали самыми старыми данными. По умолчанию память окончательных данных сконфигурирована как *кольцевая память* *(стр. 526)*, при этом новые данные перезаписывают самые старые данные.

Термин: окончательные данные в памяти (final-memory data)

Данные, находящиеся в памяти окончательных данных.

Термин: флеш-память

Тип памяти, не требующий питания от батареи. Однако флеш-память имеет время жизни, зависящее от числа операций записи в нее. Чем чаще производится запись данных, тем меньше ожидаемый срок службы.

Термин: FLOAT

Тип данных — четырехбайтовые, с плавающей запятой. По умолчанию в CR1000 этот тип данных используется для переменных типа **Public** и **Dim**. Тот же формат, что и IEEE4.

Термин: fN1

fN1 или F_{notch}. Первая частота режекции. При цифровой обработке сигналов (DSP) провалом называют область частотного спектра, в которой входной сигнал фильтра сильно подавлен. Вводимые в фильтр на частоте fN1 сигналы подавляются полностью, а частоты, близкие к частоте режекции, сильно ослабляются, но они не подавляются полностью. Более правильный технический термин — это *нулевой уровень передачи*, он означает, что на заданной частоте сигнал полностью подавляется фильтром.

Термин: FP2

Тип данных с плавающей запятой длиной два байта. В CR1000 этот тип данных используется по умолчанию для запоминаемых данных. Для переменных и целочисленных вычислений используется тип данных IEEE с плавающей запятой длиной четыре байта. Тип данных FP2 пригоден для большинства сохраняемых данных. Тип данных FP2 обеспечивает разрешение, составляющее три или четыре значащих цифры, для него требуется вдвое меньше памяти, чем для типа данных IEEE4.

Термин: FTP

File Transfer Protocol (Протокол передачи файлов). Протокол прикладного уровня в семействе протоколов TCP/IP.

Термин: дуплексный

Протокол последовательной передачи данных. Одновременная двусторонняя связь. Обычно для взаимодействия между последовательным портом CR1000 и ПК используют дуплексную связь.

Подробнее: *симплексная (стр. 528)*, *дуплексная (стр. 248)*, *полудуплексная (стр. 517)* и *полностью дуплексная (стр. 516)*.

Термин: диапазон частот

Диапазон частот описывает график зависимости Y от X, где X — частота. Данные, полученные от струнного датчика *VSPECT (стр. 532)*, относятся к диапазону частот.

Термин: амплитудно-частотная характеристика (АЧХ)

Частота выборки показывает, насколько часто прибор передает результат на свой выход; амплитудно-частотная характеристика показывает, как прибор отвечает на быстрые флуктуации на его входе. Например, при частоте выборки 1 кГц большая термопара обеспечивает высокую частоту выборки, однако при этом не гарантируется, что измерение имеет хорошую АЧХ. Весьма вероятно, что термопара из тонкой проволоки, выходной сигнал которой быстро изменяется при скачке температуры, имеет хорошую АЧХ.

Термин: «мусор»

В мире передачи данных так называют отходы. Если данные некорректно передаются или принимаются (что может быть вызвано многими причинами), то часто получается строка неправильных символов, не имеющая смысла («мусор»). Обычными причинами получения «мусора» являются: 1) различие частоты передачи данных и частоты приема данных и 2) передача синхронных данных в асинхронное устройство или передача асинхронных данных в синхронное устройство.

Термин: глобальная переменная

Переменная, используемая во всей программе на CRBasic. Этот термин обычно используют для подпрограмм. Следует различать глобальные переменные (объявленные в командах **Public** или **Dim**) и локальные переменные, объявленные в командах **Sub()** и **Function()**.

Термин: земля

Это электрический потенциал 0 вольт.

Термин: полудуплексный

Протокол последовательной передачи данных. Двусторонняя связь, которая не является одновременной. SDI-12 — это протокол полудуплексной связи.

Подробнее: *симплексная (стр. 528), дуплексная (стр. 248), полудуплексная (стр. 517)* и *полностью дуплексная (стр. 516)*.

Термин: квитирование связи

Обмен заранее определенной информацией между двумя устройствами, который гарантирует их соединение. Если линия CLK/HS (штырек 7) порта устройства регистрации **CS I/O** не используется для синхронизации, то ее обычно используют для контроля наличия или отсутствия периферийного оборудования.

Термин: обмен Hello-сообщениями

Процесс проверки соседнего узла. См. раздел *«PakBus — общие сведения» (стр. 88)*.

Термин: герц (Гц)

Единица частоты в системе единиц СИ. Количество циклов или импульсов в секунду.

Термин: HTML

Hyper**t**ext **M**arkup **L**anguage (Язык разметки гипертекста). Язык программирования, используемый для создания веб-страниц.

Термин: HTTP

Hypertext Transfer Protocol (Протокол передачи гипертекста). Протокол прикладного уровня в семействе протоколов TCP/IP.

Термин: IEEE4

Тип данных с плавающей запятой длиной четыре бита. Стандарт IEEE 754. Этот тот же формат, что и **Float**.

Термин: включаемый файл

Файл, содержащий код на CRBasic, который должен включаться в конце текущей программы на CRBasic, либо этот код может выполняться в качестве программы по умолчанию. См. *параметр* **Include File Name (Имя включаемого файла)** *(стр. 603)*.

Термин: INF

Слово данных, которое означает, что результат функции является бесконечным или неопределенным.

Термин: инициировать связь

Имя процесса, которым CR1000 инициирует связь с ПК, в котором запущено ПО *LoggerNet*. Другое название — **CAllback (Обратный вызов)** *(стр. 509)*.

Термин: команды ввода-вывода

Используется для запуска измерений и записи результатов во входную память и для настройки портов управления или логических портов или чтения из них.

Термин: команды ввода-вывода

Обычно относится к команде CRBasic.

Термин: целое

Число, не имеющее дробной части. 15 и 7956 — целые числа; числа 1,5 и 79,56 не являются целыми.

Термин: промежуточная память

См. *Память для обработки выходных данных (стр. 512)*.

Термин: IP

Internet Protocol (Интернет-протокол). Протокол межсетевого взаимодействия в семействе протоколов TCP/IP.

Термин: IP-адрес

Уникальный адрес устройства в сети Интернет.

Термин: IP trace (трассировка)

Функция, используемая для передачи данных по IP. Раньше для доступа к информации IP trace использовалась команда CRBasic **IPTrace()** *(стр. 289)*; ее результат сохранялся в строковой переменной. Теперь возможности *параметра* **Files Manager (Диспетчер файлов)** *(стр. 603)* изменились, и он позволяет создавать файл в накопителе CR1000 (например, в USR:) для хранения информации в кольцевой памяти.

Термин: изоляция

Соединенные кабелями устройства связи и сами кабели могут формировать альтернативные цепи заземления и точки ввода электромагнитных шумов в CR1000. Альтернативные цепи заземления и электромагнитные шумы могут быть источниками ошибок при измерениях. При использовании оптических развязок в разъемах сигналы связи пропускаются, однако разрываются альтернативные цепи заземления, также удается отфильтровать некоторые электромагнитные шумы. Компания Campbell Scientific предлагает интерфейсы RS-232 с вводом-выводом CS с оптической развязкой в качестве дополнительной возможности CR1000, используемой в порту ввода-вывода **CS I/O**. См. приложение *«Список модулей ввода-вывода для последовательной связи» (стр. 646)*.

Термин: JSON

Java Script **O**bject Notation (Объектная нотация в Java-скрипте). Формат файла данных, доступный в CR1000 и *LoggerNet*.

Термин: память KEEP

Энергонезависимая память, в которой хранятся некоторые *регистры* *(стр. 603)* используемые при перезагрузке CR1000 после включения питания и запуска программы. Например, адрес PakBus, имя станции, интервалы между маркерами, списки соседей, таблица маршрутизации и тайм-ауты, используемые при связи.

Термин: дисплей с клавиатурой

Дополнительный дисплей с клавиатурой CR1000KD, используемый в качестве периферийного оборудования для регистратора данных CR1000. Список других совместимых дисплей с клавиатурой приведен в Приложении *Дисплей с клавиатурой — список* *(стр. 651)*.

Термин: оконечный узел

Узел PakBus, расположенный в конце ветви. В этом режиме CR1000 не может передавать пакеты из одного своего коммуникационного порта в другой свой коммуникационный порт. В этом режиме список соседей не поддерживается, но возможно взаимодействие с другими устройствами регистрации PakBus и беспроводными датчиками. Этот режим невозможно использовать для установления связи с другими устройствами регистрации и маршрутизации данных на них.

Термин: lf

Перевод строки (от «line **f**eed»). Часто ассоциируется с разрывом строки (<cr>). **<cr><lf>.**

Термин: локальная переменная

Переменная, которую можно использовать только в той подпрограмме, где она объявлена. Этот термин используется для различения локальных переменных, которые объявлены в командах **Sub()** и **Function()**, и глобальных переменных, объявленных в командах **Public** и **Dim**.

Термин: LONG

Тип данных, используемый для объявления целых чисел.

Термин: цикл

Последовательность команд в программе на CRBasic, которая повторяется программируемое число раз. Цикл завершается командой **end**.

Термин: счетчик циклов

Его значение увеличивается на единицу при каждом прохождении цикла.

Термин: питание от сети

Национальная электрическая сеть

Термин: инициируемый вручную

Инициируемый пользователем, обычно с *Дисплея с клавиатурой CR1000KD (стр. 651)*, в отличии от вызванного программой.

Термин: запоминающее устройство

USB: флеш-накопитель. См. приложение*«Устройства для хранения данных (стр. 653)*.

Термин: MD5-дайджест

16-байтовая контрольная сумма конфигурации VTP TCP/IP.

Термин: милли

Префикс, означающий 1/1000 базовой единицы измерений в системе измерений СИ.

Термин: Modbus

Коммуникационный протокол, опубликованный компанией Modicon в 1979 г. для использования в программируемых логических контроллерах (ПЛК). См. раздел *Modbus (стр. 91)*.

Термин: модем/терминал

Любое устройство, имеющее:

- Способность организовать кольцевую линию CR1000 или использоваться с интерфейсом с оптической развязкой (см. приложение *Интерфейсы последовательного ввода-вывода CS (стр. 652)*) для организации кольцевой линии и переключения CR1000 в режим ожидания команд связи.
- Асинхроный последовательный порт, который можно настроить на взаимодействие с CR1000.

Термин: деление по модулю

Математическая операция. Результат равен остатку после деления.

Термин: MSB

Наиболее значащий **бит** (самый старший бит). См. приложение *Порядок следования байтов (стр. 643)*.

Термин: мультиметр

Недорогое широко используемое устройство, часто применяемое для поиска неисправностей в системах сбора данных.

Термин: множитель

Этот термин часто означает параметр, используемый в команде измерения CRBasic, который означает коэффициент наклона для линейной функции (другое название — «масштабный коэффициент»). Например, при пересчете температуры из °C в °F используется уравнение °F = °C*1,8 + 32. Коэффициент **1,8** представляет собой множитель. См. *Термин: смещение (стр. 521)*.

Термин: мВ

Сокращенное обозначение милливольта в системе измерений СИ.

Термин: NAN

Not **a** **n**umber (Не число). Слово данных, означающее ошибку измерения или обработки. Выход за пределы по напряжению, ошибка датчика SDI-12 и неопределенный математический результат может дать NAN. См. раздел *NAN и ±INF* *(стр. 482)*.

Термин: соседнее устройство

Устройство в сети PakBus, непосредственно взаимодействующее с данным устройством без маршрутизации через промежуточное устройство. См. *PakBus* *(стр. 522)*.

Термин: NIST

National Institute of Standards and Technology (Национальный институт стандартов и технологии)

Термин: узел

Устройства в сети (обычно в сети PakBus). Коммуникационный сервер организует коммутируемый доступ к узлу по телефонной линии или взаимодействует с узлом. Узлы образуют иерархию, при этом доступ ко всем узлам обеспечивается тем же устройством (родительским узлом), введенным как дочерние узлы. Узел может одновременно быть родительским узлом и дочерним узлом. См. *PakBus — общие сведения* *(стр. 88)*.

Термин: NSEC

Тип данных длиной восемь битов, состоящий из четырех байтов секунд времени, протекшего начиная с 1990 года, и четырех байтов наносекунд в секунде. См. таблицы *Типы данных* *(стр. 131, стр. 130)*.

Термин: нуль-модем

Устройство (обычно многожильный кабель), преобразующее порт RS-232 из DCE в DTE или из DTE в DCE.

Термин: цифровой монитор

Цифровой монитор в *ПО регистратора данных* *(стр. 654)* или в дисплее с клавиатурой.

Термин: смещение

Член, часто представляющий собой параметр в команде измерения CRBasic, означающий точку пересечения линейной функцией оси Y (иначе называемый фактором сдвига или значением в нуле). Например, для пересчета температуры из °C в °F используют уравнение °F = °C*1,8 + 32. Число **32** представляет собой смещение. См. *Термин: множитель* *(стр. 520)*.

Термин: Ом

Единица измерения сопротивления. Для обозначения Ома используется греческая буква «омега» (Ω). 1,0 Ом равен отношению 1,0 В к 1,0 А.

Термин: закон Ома

Этот закон устанавливает связь между током, сопротивлением и напряжением. Напряжение равно произведению тока на сопротивление (V = I • R).

Термин: онлайновая передача данных

Обычная передача данных в установленное на объекте периферийное оборудование. Такая передача производится под управлением программы, введенной в устройство регистрации.

Термин: операционная система

Операционная система (которую также называют «встроенное ПО») — это набор команд, реализующих основные функции CR1000, которые пользователь может применять при написании программ на CRBasic. Операционная система предварительно загружается в CR1000 на заводе, но можно снова загрузить или обновить ее, используя ПО *Утилиты конфигурации устройств* *(стр. 111)* . Объектный модуль .obj последней доступной операционной системы CR1000 можно скачать по адресу *www.campbellsci.com/downloads* (*http://www.campbellsci.com/downloads*).

Термин: вывод

Широкий термин. Он означает: а) информацию, передаваемую электронным датчиком, б) передачу данных из памяти переменных в память окончательных данных, или в) передачу электрической мощности из CR1000 или из периферийного устройства в другое устройство.

Термин: выходной массив

Строка значений выходных данных в памяти окончательных данных. Вывод производится тогда, когда триггер таблицы выходных данных равен **True** (Истина).

Термин: интервал выходных данных

См. *интервал выходных данных* *(стр. 512)*.

Термин: команды обработки выходных данных

См. *команды обработки выходных данных* *(стр. 512)*.

Термин: память обработанных выходных данных

См. *память для обработки выходных данных* *(стр. 512)*.

Термин: PakBus

Фирменный коммуникационный протокол, аналогичный протоколу *IP* *(стр. 518)* , разработанный компанией Campbell Scientific для обеспечения связи между измерительными приборами Campbell Scientific. Дополнительная информация приведена в разделе *PakBus — общие сведения* *(стр. 88)*.

Термин: ПО PakBusGraph

Показывает взаимодействие различных узлов в сети PakBus, позволяет вести мониторинг и настраивать некоторые *регистры* *(стр. 525)* в каждом узле. Обычно узлом PakBus является регистратор данных Campbell Scientific, ПК или телекоммуникационное устройство. См. раздел *ПО регистратора данных* *(стр. 450)*.

Термин: параметр

Параметр (стр. 523): часть определения процедуры (или команды).

Аргумент (стр. 507): часть вызова процедуры (или выполнения программы). Аргумент вставляется в качестве параметра. Например, в команде CRBasic **Battery(*dest*) *dest*** = это часть параметра, определяющего, какой аргумент должен вставляться на его место в программе на CRBasic. Если переменная с именем ***BattV*** должна содержать результат измерения батареи функцией **Battery(), то *BattV*** — это аргумент, вставляемый вместо параметра ***dest***. В операторе

```
Battery(BattV)
```

BattV является аргументом.

Термин: среднее за период

Метод измерений, в котором высокочастотные цифровые часы используются для измерения различия по времени между сигнальными переходами. Датчики, для которых обычно используется среднее за период, включают рефлектометры, определяющие содержание воды.

Термин: периферийный

Любое устройство, предназначенное для использования совместно с CR1000 (или другим регистратором данных Campbell Scientific). Для периферийного устройства требуется работающее устройство CR1000. Периферийные устройства включают модули *измерения и контроля (стр. 85)*, а также модули *выборки данных и связи (стр. 651)*.

Термин: ping

Утилита, которая пытается взаимодействовать с другим устройством в сети. См. раздел *PakBus — общие сведения (стр. 88)* и разделы *Ping (PakBus) (стр. 398)* и *Ping (IP) (стр. 295)*.

Термин: ping

Режим выполнения программы на CRBasic, в котором команды оцениваются как группы похожих команд, имеющие заданный приоритет группы. Дополнительная информация приведена в разделе *Конвейерный режим (стр. 152)*. См. *Термин: последовательный режим (стр. 527)*.

Термин: коэффициент Пуассона

Коэффициент, используемый при измерении механического напряжения. Он равен отношению поперечной деформации к деформация растяжения:

$\nu = -(\varepsilon_{trans} / \varepsilon_{axial})$.

Термин: точность

Мера воспроизводимости измерений. Также см. приложение *Погрешность, точность и разрешение (стр. 533)*.

Термин: PreserveVariables

Команда CRBasic, защищающая переменные типа Public от стирания при рекомпиляции программы.

Термин: печатающее устройство

Любое устройство, способное получать выходной сигнал по штырьку 6 (линия PE) в режиме «только чтение». Принтеры. Непрограммируемые терминалы и компьютеры в терминальном режиме, относятся к этой группе.

Термин: периферийное печатающее устройство

См. *печатающее устройство (стр. 524)*.

Термин: команды обработки

Команды CRBasic, используемые для дальнейшей обработки значений входных данных, которые возвращают результат в виде переменной, которая доступна для обработки выходных данных.

Эти команды включают арифметические функции и трансцендентные функции. См. Приложение *Команды обработки и математические функции (стр. 563)*.

Термин: команды управления программой

Они изменяют порядок выполнения команд CRBasic. Кроме того, они используются для установки флажков и их удаления. См. раздел *Управление ПЛК — общие сведения (стр. 74)*.

Термин: оператор программы

Конструкция команды программы ограничена одной строкой, либо несколько строк команды соединяются символами продолжения строки <пробел><подчеркивание> (_). Длина командной строки, включая символы продолжения строки, не может превышать 512 символов.

Термин: команда Program Send

Program Send — это возможность *ПО регистратора данных (стр. 95)*. Определение команды зависит от ПО, см. приведенную ниже таблицу:

Таблица 135. Команда Program Send

Программное обеспечение	*Команда*	*Расположение команды*
LoggerNet	**Send New... (Отправить новый...)**	*Экран Connect*
PC400	**Send Program (Отправка программы)**	**Вкладка Clock/Program**
RTDAQ	**Send Program (Отправка программы)**	**Вкладка Clock/Program**
PC200W	**Send Program (Отправка программы)**	**Вкладка Clock/Program**

Термин: Public

Команда CRBasic, используемая для объявления переменных и их размерности. При работе регистратора данных можно вести мониторинг переменных, объявленных в команде **Public**. См. *Термин: Dim (стр. 513)*.

Термин: импульс

Электрический сигнал, для которого характерно быстрое увеличение напряжения, короткое плато и быстрое уменьшение напряжения.

Термин: запись

Запись — это полная строка данных в таблице данных или файле данных. Все данные в записи имеют общую метку времени.

Термин: регулятор (regulator)

Настройка, элемент таблицы Status или элемент таблицы DataTableInformation.

Термин: регулятор (regulator)

Устройство для стабилизации источника питания. Обычно регуляторы компании Campbell Scientific преобразуют напряжение переменного тока или напряжение пост. тока, превышающее 16 В пост. тока, в напряжение , приблизительно составляющее 14 В пост. тока.

Термин: команда Reset Tables

Команда **Reset Tables** перезагружает таблицы данных, настроенные на заполнение и остановку. Эта команда располагается в различных местах ПО регистратора данных согласно следующему:

LoggerNet — экран Connect | Вкладка **Station Status** | Вкладка **Table Fill Times** Вкладка | **Reset Tables**

PC400 — последовательность команд: **Datalogger | Station Status | Table Fill Times | Reset Tables**

RTDAQ — последовательность команд: **Datalogger | Station Status | Table Fill Times | Reset Tables**

PC200W — последовательность команд: **Datalogger | Station Status | Table Fill Times | Reset Tables**

Термин: сопротивление

Свойство электрической цепи замедлять поток электронов или перенаправлять поток электронов в цепи.

Термин: резистор

Устройство, которое обеспечивает определенное сопротивление.

Термин: разрешение

Мера точности измерений. Также см. приложение *Погрешность, точность и разрешение* *(стр. 533)*.

Термин: кольцевая линия

Кольцевая линия запускается внешним устройством, которое передает CR1000 команду организовать связь по порту RS-232. Кольцевая линия соответствует штырьку 3 *DCE* *(стр. 513)* порта RS-232.

Термин: кольцевая память

Конфигурация памяти, при которой самые новые данные перезаписывают самые старые данные. Эта настройка используется по умолчанию для таблиц окончательных данных.

Термин: колебания (ringing)

Колебания напряжения или тока сигнала на выходе датчика, при которых возбуждение датчика приводит к резонансу паразитных емкостей и индуктивностей.

Термин: RMS

Среднеквадратическое значение. Мера колебаний амплитуды волны или других изменяющихся величин вокруг нуля.

Термин: маршрутизатор

Устройство, сконфигурированное как маршрутизатор, может пересылать пакеты PakBus из одного порта в другой порт. Для выполнения своих функций маршрутизации CR1000 настроен как маршрутизатор, который поддерживает собственный список соседей и передает этот список другим маршрутизаторам в сети PakBus. Кроме того, это устройство получает списки соседей от других маршрутизаторов.

Термин: RS-232

Рекомендованный стандарт **RS-232**. Нестрогий стандарт, определяющий взаимодействие двух вычислительных устройств. Реализация протокола RS-232 для связи регистраторов данных Campbell Scientific с ПК является весьма жесткой, но прозрачной для большинства пользователей. CR1000 имеет гибкие возможности реализации протокола связи RS-232 с интеллектуальными датчиками.

Термин: частота выборки

Скорость измерения устройством CR1000. Частота выборки измерений представляет интерес при рассмотрении влияния временного сдвига или того, насколько близко по времени производится серия измерений, или насколько близка метка времени измерения к истинному времени явления. Максимальная частота выборки — это частота, с которой можно повторять измерения, производимые одной командой CRBasic.

Частота выборки показывает, насколько часто прибор передает результат на свой выход; амплитудно-частотная характеристика показывает, как прибор отвечает на быстрые флуктуации на его входе. Например, при частоте выборки 1 кГц большая термопара обеспечивает высокую частоту выборки, однако при этом не гарантируется, что измерение имеет хорошую АЧХ. Весьма вероятно, что термопара из тонкой проволоки, выходной сигнал которой быстро изменяется при скачке температуры, имеет хорошую АЧХ.

Термин: интервал сканирования (scan interval)

Интервал времени между инициированием каждого выполнения данной команды **Scan()** в программе на CRBasic. Если 24 часа (86 400 секунд) делятся на **Scan()** ***Interval*** (интервал сканирования) без остатка, то интервал сканирования синхронизируется с 24-часовыми часами, причем эта программа запускается в полночь, а затем — через каждый интервал сканирования **Scan()** ***Interval***. Программа впервые запускается после первого появления **Scan()** ***Interval*** после компиляции. Если 24 часа не делятся на интервал сканирования **Scan()** ***Interval*** без остатка, то выполнение начнется в первую секунду после компиляции.

Термин: время сканирования

Если функции времени запускаются внутри конструкции **Scan()/NextScan**, то временные отметки основаны на моменте начала сканирования по часам CR1000. Разрешение времени сканирования равно продолжительности сканирования. См. *системное время (стр. 530)*.

Термин: SDI-12

Serial **D**ata **I**nterface (Последовательный интерфейс для передачи данных), скорость передачи составляет **1200** бод. Протокол связи для обмена данными между CR1000 и совместимых с SDI-12 интеллектуальными датчиками.

Термин: SDM

Synchronous **D**evice for **M**easurement (Синхронное измерительное устройство). Микропроцессорное периферийное устройство или датчик, взаимодействующий по короткому кабелю с CR1000 по фирменному протоколу Campbell Scientific.

Термин: прямой термоэлектрический эффект

Индуцирует термо-ЭДС, имеющее напряжение порядка одного мкВ, на стыках различных металлов, контакты между которыми находятся при различных температурах. На этом принципе основано измерение температуры с помощью термопары. Кроме того, этот эффект вызывает малые корректируемые смещения в измерительных цепях CR1000.

Термин: ping

Режим выполнения программы на CRBasic, при котором каждый оператор выполняется в том порядке, в котором он включен в программу. Дополнительная информация приведена в разделе *Последовательный режим (стр. 153)*. См. *Термин: конвейерный режим (стр. 523)*.

Термин: семафор (семафор измерения)

При выполнении основного сканирования в последовательном режиме семафор блокирует ресурсы, связанные с измерениями. Иначе говоря, он получает семафор измерений. Это происходит на уровне сканирования, поэтому все частичные сканирования внутри сканирования (независимо от того, производят ли они измерения), будут блокировать измерения с медленной последовательностью (включая фоновую калибровку системы). Блокировка ресурсов измерения на уровне сканирования обеспечивает непрерывное выполнение основного сканирования.

Термин: send (передавать)

Кнопка **Send** в *ПО средства регистрации (стр. 95)*. Пересылает программу на CRBasic или операционную систему в CR1000.

Термин: последовательный

Широкий термин, означающий вывод последовательности буквенно-цифровых символов в электронном виде.

Термин: ПО Short Cut

Мастер программы на CRBasic, пригодный для многих приложений CR1000. Для использования мастера *Short Cut* не требуется знание языка CRBasic. Этого мастера можно бесплатно скачать с сайта *www.campbellsci.com*.

Термин: SI (Système Internationale)

Международная система метрических единиц измерения. В ней определены общепринятые единицы измерений.

Термин: сигнатура

Число, зависящее от данных и последовательности данных в памяти. Для ее расчета используется алгоритм, который с вероятностью 99,998 % гарантирует, что сигнатура изменяется при изменении данных или последовательности данных. См. разделы *Безопасность — общие сведения* *(стр. 92)* и *Сигнатуры* *(стр. 472)*.

Термин: симплексная

Протокол последовательной передачи данных. Данные передаются только в одном направлении. Последовательная связь между последовательным датчиком и CR1000 может быть симплексной.

Подробнее: *симплексная* *(стр. 528)*, *дуплексная* *(стр. 248)*, *полудуплексная* *(стр. 517)* и *полностью дуплексная* *(стр. 516)*.

Термин: симплексная

Означает датчик или измерительный терминал, для которого аналоговый сигнал напряжения передается по одному проводу и измеряется по отношению к земле (0 В).

Термин: пропущенные сканирования

Они образуются, когда программа на CRBasic выполняется слишком долго для данного интервала сканирования. Пропущенные сканирования могут привести к ошибками при измерении импульсов.

Термин: медленная последовательность

Обычно этот термин означает более медленное вторичное сканирование в программе на CRBasic. Основное сканирование имеет приоритет по сравнению с медленной последовательностью.

Термин: SMTP

Simple Mail Transfer Protocol (Простой протокол электронной почты). Протокол прикладного уровня в семействе протоколов TCP/IP.

Термин: SNP

Файл снимка

Термин: SP

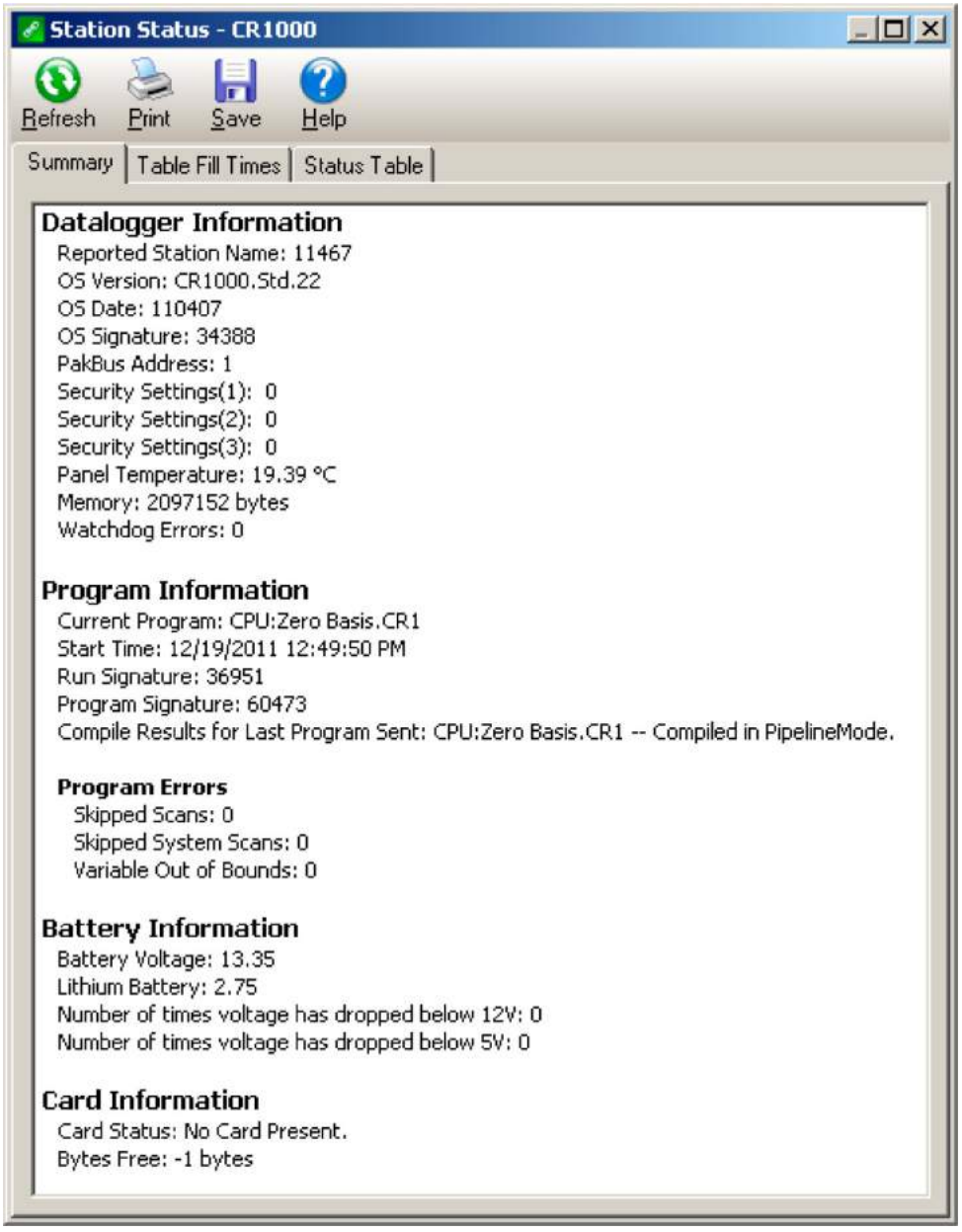
Пауза

Термин: состояние

Показывает, включено или выключено устройство.

Термин: команда Station Status (Состояние станции)

Команда, включенная в большую часть *ПО регистратора данных* *(стр. 95)*. На приведенном ниже рисунке показан пример вывода команды *station status*.



Термин: строка

Данные или переменная, состоящие из алфавитно-цифровых символов.

Термин: ПО (программное обеспечение)

См. *Термин: ПО регистратора данных* *(стр. 512)*.

Термин: сканирование частоты

Последовательность частот от самой низкой частоты до самой высокой частоты, используемая при возбуждении струны при измерениях *VSPECT (стр. 532)*.

Термин: синхронный

Передача данных между передающим и принимающим устройством происходит в виде последовательности нулей и единиц. Для корректного «считывания» данных принимающее устройство должно начинать считывание в правильной точке последовательности. При синхронной связи такая координация осуществляется путем синхронизации передающего и принимающего устройства с помощью общего тактового сигнала (см. *Асинхронный (стр. 247)*).

Термин: системное время

Если функции времени выполняются вне конструкции **Scan() / NextScan**, то регистрируемое этой командой время будет браться из системных часов, разрешение которых составляет 10 мс. См. *время сканирования (стр. 527)*.

Термин: задание

Используются два определения:

- Автоматическая группировка операторов программы на CRBasic компилятором CR1000. Задание включает SDM или цифровое измерение и обработку. Заданиям присваиваются приоритеты, если программа на CRBacis выполняется в конвейерном режиме.
- Настраиваемая пользователем функция, определенная с помощью *Мастера заданий LoggerNet*.

Термин: TCP/IP

Transmission Control Protocol / Internet Protocol (Протокол управления передачей / Межсетевой протокол).

Термин: Telnet

Программная утилита, которая пытается установить связь с другим конкретным устройством в сети и опросить его. Telnet — это резидентная программа в ОС Windows.

Термин: вывод

Точка, в которой провод (или провода) соединяется с коммутационной панелью или разъемом. Как правило, провод крепится в выводе небольшим винтовым или пружинным зажимом и закрывается крышкой, крепящейся с помощью винта или подпружиненного рычага. См. *разъем (стр. 510)*.

Термин: эмулятор терминала

Командная оболочка, которая облегчает формирование команд низкого уровня для регистрирующего устройства или какого-либо другого совместимого устройства. Эмулятор терминала включен в подавляющую часть версий *ПО регистратора данных (стр. 95)* компании Campbell Scientific.

Термин: термистор

Термистор — это устройство для измерения температуры, имеющее резистивный элемент, сопротивление которого изменяется при изменении температуры. Такое изменение является сильным, стабильным, причем оно хорошо известно. Как правило, выход термистора является нелинейным, поэтому при измерении требуется произвести линеаризацию с помощью уравнения Стейнхарта-Харта или полиномиального уравнения. В командах CRBasic **Therm107(),**

Therm108() и **Therm109()** используются уравнения Стейнхарта-Харта.

Термин: временная область

Временная область описывает данные, нанесенные на график X-Y, причем по оси X отложено время. Данные временных рядов находятся во временной области.

Термин: пропускная способность

Скорость, на которой можно производить измерения в инженерных единицах, а также сохранять их в таблице окончательных данных в памяти. CR1000 может сканировать датчики со скоростью, превышающую пропускную способность. Основным фактором, определяющим пропускную способность, является обработка, запрограммированная в программе на CRBasic. В последовательном режиме вся требуемая командой обработка должна быть завершена перед переходом к следующей команде.

Термин: TTL

Transistor-to-**T**ransistor **L**ogic (Транзисторно-транзисторная логика). Последовательный протокол, использующий уровни логического сигнала 0 В пост. тока и 5 В пост. тока.

Термин: TLS

Transport **L**ayer **S**ecurity (Безопасность на транспортном уровне). Протокол защищенной передачи по сети Интернет.

Термин: переключение [между двумя состояниями]

Изменение текущего состояния питания.

Термин: UINT2

Тип данных, используемый для эффективного хранения отсчетов счетчика импульсов, состояния порта (например, состояние 16 портов хранится в одной переменной) или целочисленных значений, в которых хранятся двоичные флаги.

Термин: безусловный выход

Основной выход датчика, либо выход датчика до применения коэффициентов масштабирования. См. *условный вывод* *(стр. 510)*.

Термин: ИБП (UPS)

Источник **Б**есперебойного **П**итания. ИБП можно применять для большинства приложений регистратора данных, использующих питание от сети переменного тока, трансформатор перем. тока с вилкой или адаптер переменного тока в постоянный с вилкой, устройство зарядки и аккумулятор. Для CR1000 требуется также внешнее устройство зарядки.

Термин: программа пользователя

Программа на CRBasic, написанная вами в мастере программы *Short Cut*.

Термин: USR: накопитель

Часть памяти CR1000, выделенная для хранения файлов изображений или других файлов.

Термин: URI

Универсальный идентификатор ресурса

Термин: URL

Универсальный локатор ресурса

Термин: переменная

Пакет статического ОЗУ, которому присвоено буквенно-цифровое имя. Переменные находятся в памяти переменных.

Термин: память переменных

Часть статического ОЗУ, зарезервированная для хранения переменных. Память переменных может перезаписываться и регулярно перезаписывается новыми значениями или строками согласно программе на CRBasic. Если переменные объявлены **As Public**, то возможен визуальный мониторинг памяти.

Термин: В перем. тока

Volts **a**lternating **c**urrent (вольт перем. тока). Другое обозначение: VAC. Используются два определения:

- Питание от сети — это питание от источника с высоким напряжением перем. тока, обычно 110 В перем. тока или 220 В перем. тока и фиксированной частотой 50 Гц или 60 Гц. Источник с высоким Vac можно использовать как основной источник питания для блоков питания Campbell Scientific. Не подавайте высокое напряжение питания непосредственно на CR1000.
- CR1000 измеряет изменяющуюся частоту низкого напряжения переменного тока в диапазоне ±20 В перем. тока. Например, некоторые анемометры формируют низковольтный сигнал перем. тока.

Термин: В пост. тока

Volts **d**irect **c**urrent (вольт пост. тока). Другое обозначение — VDC. Используются два определения:

- CR1000 работает от источника с номинальным напряжением питания 12 В пост. тока. CR1000 предоставляет номинальное напряжение 12 В пост. тока, регулируемое напряжение 5 В пост. тока, регулируемое напряжение 3,3 В пост. тока и переменное возбуждение в диапазоне ±2,5 В пост. тока.
- CR1000 измеряет аналоговое напряжение в диапазоне ±5,0 В пост. тока и импульсное напряжение в диапазоне ±20 В пост. тока.

Термин: вольтметр

См. *Термин: мультиметр (стр. 520)*.

Термин: Вольт

Единица измерения электрического потенциала в системе SI

Термин: VSPECT

Товарный знак запатентованного Campbell Scientific метода спектрального анализа, способа измерения области частот и способа измерения с помощью струнного датчика.

Термин: сторожевой таймер

Система исправления ошибок, которая отслеживает состояние процессора, программные таймеры и программные счетчики при выполнении программы на CRBasic. См. раздел *Ошибки сторожевой схемы (стр. 488)*. Следующие события будут вызывать сброс сторожевого таймера, при этом перезагружается процессор и восстанавливается выполнение программы на CRBasic.

- Отказ процессора
- Процессор проигнорировал стандартное обновление системы
- Содержание датчика лежит вне заданных пределов
- Скачок напряжения
- Переходное напряжение

При перезагрузке увеличивается содержимое счетчика в записи **WatchdogTimer** таблицы **Status** *(стр. 603)*. Небольшое число перезагрузок сторожевого таймера (от 1 до 10) может вызывать беспокойство, но обычно это указывает на то, что требуется только контроль ситуации.

Большое количество ошибок (> 10), накопленное за короткий период времени, указывает на наличие аппаратной или программной проблемы. Проконсультируйтесь с инженером по приложениям Campbell Scientific.

Термин: защищенный от атмосферных воздействий

Означает, что корпус прибора является непроницаемым при обычном состоянии окружающей среды. Однако при чрезвычайных погодных явлениях герметичность корпуса может быть нарушена.

Термин: веб-API

Application Programming Interface (прикладной программный интерфейс) (для получения дополнительной информации см. раздел *Web Service API (стр. 423)*).

Термин: подстановочный

Символ или выражение, вместо которого подставляется любой другой символ или выражение.

Термин: XML

Extensible markup language (Расширяемый язык разметки).

Термин: программа пользователя

Программа на CRBasic, написанная вами с помощью мастера *Short Cut* или *CRBasic Editor*.

11.2. Концепции

11.2.1. Погрешность, точность и разрешение

Часто путают три следующих термина: погрешность, точность и разрешение. Погрешность — это мера правильности одного измерения или группы измерений в совокупности. Точность — это мера воспроизводимости группы измерений. Разрешение — это мера точности отсчета. В совокупности эти три термина описывают точность работы системы регистратора. Для объяснения связи этих трех терминов рассмотрим в качестве аналога учебную стрельбу по мишеням. В таблице *Погрешность, точность и разрешение (стр. 533)* показаны четыре мишени. Яблочко каждой мишени представляет собой абсолютно правильное измерение. Каждый выстрел является аналогом попытки проведения измерения. Диаметр пули является аналогом разрешения. Цель системы сбора данных: обеспечить низкую погрешность, высокую точность, а также получить данные с разрешением, пригодным для данного приложения.

Рисунок 133. Взаимосвязь погрешности, точности и разрешения

12. Заявления о принадлежности

Использование перечисленных ниже товарных знаков в *Руководстве оператора* CR1000 не означает получение разрешения компанией Campbell Scientific от соответствующих компаний-владельцев, которые перечислены ниже:

- Crydom
- Newark
- Mouser
- MicroSoft
- WordPad
- HyperTerminal
- LI-COR

Приложение A. Команды программирования на CRBasic

Связанные темы:
• *Программирование CRBasic — общие сведения* *(стр. 86)*
• *Программирование CRBasic — подробная информация* *(стр. 122)*
• *Программирование CRBasic — команды* *(стр. 537)*
• *Библиотека программных ресурсов* *(стр. 169)*
• *Справка для редактора CRBasic*

В этом приложение приведен полный список *команд CRBasic CR1000* *(стр. 518)*.

- Алфавитный список команд приведен в разделе *Указатель* в статье *Команда*.
- Примеры кодов приведены во всем этом руководстве, их можно найти, используя список *Таблица примеров CRBasic*.
- Списки параметров, информация о приложениях и примеры кодов приведены в Справочной информации *CRBasic Editor* *(стр. 125)*.

A.1 Объявления программы

Команды, используемые при объявлении программы, обычно содержатся в программе перед командой **BeginProg()**.

AngleDegrees

Определяет, что в математических функциях в качестве аргументов используются градусы, а не радианы.

Синтаксис

```
AngleDegrees
```

EncryptExempt

Определяет один или несколько адресов PakBus, на которые регистратор данных не будет посылать зашифрованные сообщения PakBus, даже если шифрование PakBus включено.

Синтаксис

```
EncryptExempt(Begi nPakBusAddr, EndPakBusAddr)
```

PipelineMode

Настраивает CR1000 на выполнение заданий измерений отдельно от заданий обработки, но одновременно с ними.

Синтаксис

```
PipelineMode
```

SequentialMode

Настраивает регистратор данных на последовательное выполнение заданий.

Синтаксис

```
SequentialMode
```

SetSecurity

Задает цифровые пароли для уровней защиты регистратора данных 1, 2 и 3. Выполняет программу во время компиляции.

Синтаксис

```
SetSecurity(security[1], security[2], security[3])
```

StationName

Задает имя станции, используемое внутри CR1000. Не оказывает влияния на файлы данных, созданные ПО регистратора данных. См. разделы *Прочие возможности* *(стр. 174)* и *Условный вывод* *(стр. 170)*.

Синтаксис

```
StationName(name of station)
```

Sub / ExitSub / EndSub

Объявляет имя, переменные и код, которые образуют подпрограмму. Список аргументов не обязателен. Команда **Exit Sub** является необязательной.

Синтаксис

```
Sub subname (argument list)
   [statement block]
Exit Sub
   [statement block]
End Sub
```

WebPageBegin / WebPageEnd

См. *TCP/IP — подробная информация* *(стр. 289)*.

A.1.1 Объявления и модификаторы переменных

Псевдоним

Присваивает переменной второе имя.

Синтаксис

```
Alias [variable] = [alias name]; Alias [array(4)] = [alias
   name], [alias name(2)], [alias name]
```

As

Определяет тип данных для переменных **Dim** или **Public**.

Синтаксис

```
Dim [variable] AS [data type]
```

Dim

Объявляет и определяет размерность переменных типа Private. Размерность — это необязательный элемент. Переменные **Dim** невозможно вывести на *цифровой монитор* *(стр. 521)*.

Синтаксис

```
Dim [variable name (x,y,z)]
```

ESSVariables

Автоматически определяет переменные, запрашиваемые приложением **E**nvironmental **S**ensor **S**tation. Используется совместно с командой **ESSInitialize.**

Синтаксис

```
ESSVariables
```

NewFieldNames

Присваивает новое имя родовой переменной или массиву обобщенного типа. Эта команда предназначена для использования с сетями беспроводных датчиков Campbell Scientific.

Синтаксис

```
NewFieldNames(GenericName, NewNames)
```

PreserveVariables

Сохраняет значения в переменных типа **Dim** или **Public** при перезапуске программы на CRBasic после отказа питания, остановки вручную или других операций, вызывающих перекомпиляцию программы.

Синтаксис

```
PreserveVariables
```

Public

Объявляет переменные типа Public и их размерности. Размерность — это необязательный элемент.

Синтаксис

```
Public [variable name (x,y,z)]
```

ReadOnly

Помечает разделенный запятыми список переменных (**Public** или с именем **Alias**) как список переменных, используемых только для чтения.

Синтаксис

```
ReadOnly [variable1, variable2, ...]
```

Единица измерения

Присваивает имя блока полю, связанному с переменной.

Синтаксис

```
Units [variable] = [unit name]
```

A.1.2 Объявление констант

Const

Объявляет символьные константы, используемые вместо числовых элементов.

Синтаксис

```
Const [constant name] = [value or expression]
```

ConstTable / EndConstTable

Объявляет константы, значения которых можно изменить, используя дисплей с клавиатурой CR1000KD или опцию **C** терминала. При изменении значений программа перекомпилируется с новыми значениями. См. *Константы (стр. 137)*.

Синтаксис

```
ConstTable
   [constant a] = [value]
   [constant b] = [value]
   [constant c] = [value]
EndConstTable
```

A.2 Объявления таблиц данных

DataTable / EndTable

Это метки начала и конца таблицы данных.

Синтаксис

```
DataTable(Name, TrigVar, Size)
   [data table modifiers]
   [on-line storage destinations]
   [output processing instructions]
EndTable
```

DataTime

Объявление в таблице данных, которое позволяет использовать системное время в метках времени.

Синтаксис

```
DataTime(Option)
```

A.2.1 Модификаторы таблицы данных

DataEvent

Устанавливает запуск и остановку триггеров при записи данных в таблицу. Одно из возможных применений — использование совместно с командой **WorstCase().**

Синтаксис

```
DataEvent(RecsBefore, StartTrig, StopTrig, RecsAfter)
```

DataInterval

Определяет интервал времени для выходной таблицы.

Синтаксис

```
DataInterval(TintoInt, Interval, Units, Lapses)
```

FillStop

Определяет остановку таблицы данных после ее заполнения. По умолчанию таблица данных организована в виде *кольцевой памяти (стр. 526)*.

Синтаксис

```
Fill Stop
```

Примечание. Для перезагрузки таблицы после ее заполнения и остановки используйте команду **ResetTable()** в программе на CRBasic или команду ПО регистратора данных *Reset Tables* *(стр. 525)* .

OpenInterval

Определяет, что обработка временных рядов должна включать все измерения, произведенные после последней записи данных.

Синтаксис

```
OpenInterval
```

TableHide

Подавляет отображение и сбор данных таблицы данных в памяти CR1000.

Синтаксис

```
TableHide
```

A.2.2 Адресаты данных

Примечание. В большинстве приложений **TableFile()** с *опцией 64* является предпочтительной командой для записи данных в запоминающее устройство Campbell Scientific или на карту данных.
Дополнительная информация приведена в разделе **TableFile()** *с опцией 64* *(стр. 206)*.

CardFlush

Немедленно записывает все буферизованные данные из внутренней памяти CR1000 и файловой системы в запоминающее устройство Campbell Scientific или на карту данных. Часто команда **TableFile()** с *опцией 64* является предпочтительным вариантом по сравнению с этой командой.

Синтаксис

```
CardFlush
```

CardOut

Передает выходные данные на карту данных. Часто команда **TableFile()** с *опцией 64* является предпочтительным вариантом по сравнению с этой командой.

Синтаксис

```
CardOut(StopRing, Size)
```

DSP4

Передать данные в дисплей DSP4. Производство монтируемого сверху дисплея DSP4 прекращено.

Синтаксис

```
DSP4(FlagVar, Rate)
```

TableFile

Записывает файл из таблицы данных в флеш-память CR1000.

Синтаксис

```
TableFileC'FileName", Options, MaxFiles, NumRecs /
   TimeIntoInterval, Interval, Units, OutStat, LastFileName)
```

A.2.3 Обработка выхода для записи в апамять окончательных данных

Читать далее. См. *Команды обработки выходных данных (стр. 145)*.

FieldNames

Непоследственно следует за командой обработки вывода для изменения используемых по умолчанию полей имен.

Синтаксис

```
FieldNames(”Fieldname1 : Description1, Fieldname2 :
   Description2...")
```

A.2.3.1 Один источник данных

Среднее

Сохраняет среднее значение по интервалу выходных данных для исходной переменной или каждого элемента указанного массива.

Синтаксис

```
Average(Reps, Source, DataType, DisableVar)
```

Ковариантa

Вычисляет ковариацию значений в массиве по времени.

Синтаксис

```
Covariance(NumVals, Source, DataType, DisableVar, NumCov)
```

FFT

Выполняет быстрое преобразование Фурье (БПФ) для сохраненных в массиве временных рядов измерений.

Синтаксис

```
FFT(Source, DataType, N, Tau, Units, Option)
```

Максимум

Сохраняет максимальное значение в интервале выходных данных.

Синтаксис

```
Maximum(Reps, Source, DataType, DisableVar, Time)
```

Медиана

Сохраняет медианное значение зависимой переменной в интервале выходных данных.

Синтаксис

```
Median(Reps, Source, MaxN, DataType, DisableVar)
```

Минимум

Сохраняет минимальное значение в интервале выходных данных.

Синтаксис

```
Minimum(Reps, Source, DataType, DisableVar, Time)
```

Момент

Сохраняет математическое ожидание в интервале выходных данных.

Синтаксис

```
Moment(Reps, Source, Order, DataType, DisableVar)
```

PeakValley

Определяет максимальное и минимальное значения сигнала.

Синтаксис

```
PeakValley(DestPV, DestChange, Reps, Source, Hysteresis)
```

Образец

Сохраняет значения тока во время вывода.

Синтаксис

```
Sample(Reps, Source, DataType)
```

SampleFieldCal

Записывает в таблицу данные колибровки в рабочих условиях. См. *Функции калибровки* *(стр. 598)*.

SampleMaxMin

Производит измерение переменной в тот момент, когда другая переменная достигает максимального или минимального значения за определенный период.

Синтаксис

```
SampleMaxMin(Reps, Source, DataType, DisableVar)
```

StdDev

Вычисляет среднеквадратичное отклонение по интервалу выходных данных.

Синтаксис

```
StdDev(Reps, Source, DataType, DisableVar)
```

Суммировать

Вычисляет сумма по интервалу выходных данных.

Синтаксис

```
Totalize(Reps, Source, DataType, DisableVar)
```

A.2.3.2 Несколько источников данных

ETsz

Сохраняет суммарное испарение (ETsz) и солнечное излучение (RSo).

Синтаксис

```
ETsz(Temp, RH, uZ, Rs, Longitude, Latitude, Altitude, Zw, Sz,
   DataType, DisableVar)
```

RainFlowSample

Сохраняет замер CDM_VW300RainFlow в таблице данных.

Синтаксис

```
RainFlowSampe(Source, DataType)
```

Вектор ветра

Обрабатывает скорость ветра, полученную от полярных или ортогональных датчиков. Для сокращения времени обработки производятся только те вычисления, которые требуются для определения запрошенных данных.

Синтаксис

```
WindVector(Repetitions, Speed/East, Direction/North,
   DataType, DisableVar, Subinterval, SensorType, OutputOpt)
```

Читать далее. См. *Вектор ветра (стр. 296)*.

A.3 Однократное выполнение при компиляции

Следующие команды расположены между командами **BeginProg** и **Scan()**.

ESSInitialize

Инициализирует значения ESS во время компиляции. Используется совместно с **ESSVariables.**

Синтаксис

```
ESSInitialize
```

MovePrecise

Используется совместно с AddPrecise. Перемещает переменную с высокой точностью в другую переменную.

Синтаксис

```
MovePrecise(PrecisionVariable, X)
```

PulseCountReset

Сбрасывает счетчики импульсов и средние значения по времени выполнения, используемые в команде счетчика импульсов. Обычно это команда считается устаревшей. Она используется только в очень специальном коде.

Синтаксис

```
PulseCountReset
```

A.4 Команды управления программой

A.4.1 Обычные команды управления программой

BeginProg / EndProg

Обозначает начало и конец программы.

Синтаксис

```
BeginProg
   [program code]
EndProg
```

Call

Передает управление программой из главной программы в подпрограмму.

Синтаксис

```
Call subname (list of variables)
```

CallTable

Вызывает таблицу данных, обычно для обработки вывода.

Синтаксис

```
CallTable(TableName)
```

Пауза

Приостанавливает выполнение программы.

Синтаксис

```
Delay(Option, Delay, Units)
```

Do / While / Until / Exit Do / Loop

Повторяет блок операторов, если условие является истинным, или до тех пор, пока условие не станет истинным.

Синтаксис

```
Do [{While | Until} condition]
   [statementblock]
[ExitDo]
   [statementblock]
Loop

-or-

Do
   [statementblock]
[ExitDo]
   [statementblock]
Loop [{While | Until} condition]
```

EndSequence

Завершает последовательность, начатую командой **BeginProg** или **SlowSequence**. Во многих приложениях эта команда является необязательной.

Синтаксис

```
EndSequence
```

Exit

Завершает выполнение программы.

Синтаксис

```
Exit
```

For / To / Step / ExitFor / Next

Повторяет группу команд определенное число раз.

Синтаксис

```
For counter = start To end [ Step increment ]
   [statement block]
[ExitFor]
   [statement block]
Next [counter [, counter][, ...]]
```

If / Then / Else / Elself / EndIf

Вызывает выполнение или обход сегмента кода после вычисления выражения. **Else** — это необязательная часть команды. **ElseIf** — это необязательная часть команды. Обратите внимание на то, что операторы **EndSelect** и **EndIf** вызывают одну и ту же функцию.

Синтаксис

```
If [condition] Then [thenstatements] Else [elsestatements]

-or-

If [condition 1] Then
   [then statements]
ElseIf [condition 2] Then
   [elseif then statements]
Else
   [else statements]
EndIf
```

Scan / ExitScan / ContinueScan / NextScan

Устанавливает скорость сканирования программы. **ExitScan** и **ContinueScan** — это необязательные части команды. См. раздел *Измерение: более высокие аналоговые скорости* *(стр. 229)*, где приведена информация по использованию команд **Scan() / NextScan** при измерении импульсов.

Синтаксис

```
Scan(Interval, Units, Option, Count)
   [statement block]
ExitScan
   [statement block]
ContinueScan
   [statement block]
NextScan
```

Select Case / Case / Case Is / Case Else / EndSelect

Выполняет один блок операторов или несколько блоков операторов в зависимости от значения выражения. **CaseElse** — это необязательная часть команды. Обратите внимание на то, что операторы **EndSelect** и **EndIf** вызывают одну и ту же функцию.

Синтаксис

```
Select Case testexpression
Case [expression 1]
   [statement block 1]
Case [expression 2]
   [statement block 2]
Case Is [expression fragment]
Case Else
   [statement block 3]
EndSelect
```

SlowSequence

Отмечает начало участка кода, который будет выполняться одновременно с главной программой.

Синтаксис

```
SlowSequence
```

SubScan / NextSubScan

Управляет мультиплексором или измеряет некоторые аналоговые входы на большей скорости, чем при сканировании программой. См. *Измерение: более высокие аналоговые скорости (стр. 229)* , где приведена информация по использованию команд **SubScan / NextSubScan**.

Синтаксис

```
SubScan(SubInterval, Units, Count)
   [measurements and processing]
NextSubScan
```

TriggerSequence

Используется совместно с **WaitTriggerSequence** для управления выполнением кода в медленной последовательности.

Синтаксис

```
TriggerSequence(SequenceNum, Timeout)
```

WaitTriggerSequence

Используется совместно с **TriggerSequence** для управления выполнения кода в медленной последовательности.

Синтаксис

```
WaitTriggerSequence
```

WaitDigTrig

Запускает сканирование измерений от внешнего цифрового триггера.

Синтаксис

```
WaitDigTrig(ControlPort, Option)
```

While / Wend

Выполняет последовательность операторов в цикле, пока заданное условие является истинным.

Синтаксис

```
While [condition]
   [StatementBlock]
Wend
```

A.4.2 Дополнительные команды управления программой

Data / Read / Restore

Определяет список читаемых констант типа FLOAT (с помощью команды **Read**) в массив переменных далее в программе.

Синтаксис

```
Data [list of constants]
   Read [VarExpr]
Restore
```

DataLong / Read / Restore

Определяет список читаемых констант типа LONG (с помощью команды ***Read***) в массив переменных далее в программе.

Синтаксис

```
DataLong [list of constants]
   Read [Variable Expression]
Restore
```

IfTime

Возвращает число, означающее **True (-1)** или **False (0)**, используя часы реального времени CR1000.

Синтаксис

```
If (IfTime(TintoInt, Interval, Units)) Then

-or-

Variable = IfTime(TintoInt, Interval, Units)
```

Read

Читает в массив переменных константы из списка, определенного в команде **Data** или **DataLong**.

Синтаксис

```
Read [Variable Expression]
```

Restore

Возвращает положение указателя **Read** на первое положение в списке, определенном в команде **Data** или **DataLong**.

Синтаксис

```
Restore
```

SemaphoreGet

Запускает *семафор* *(стр. 527)* от 1 до 3 для предотвращения конфликтов ресурсов.

Синтаксис

```
SemaphoreGet()
```

SemaphoreRelease

Отменяет *семафор* *(стр. 527)*, ранее запущенный командой **SemaphoreGet().**

Синтаксис

```
SemaphoreRelease()
```

ShutDownBegin

Запускает выполнение кода в случае аварийного завершения, например, при передаче новой программы.

Синтаксис

```
ShutDownBegin
```

ShutDownEnd

Останавливае выполнение кода в случае нормального завершения, например, при передаче новой программы.

Синтаксис

```
ShutDownEnd
```

TimeIntoInterval

Возвращает число, означающее **True (-1)** или **False (0)** на основании показаний часов реального времени устройства регистрации.

Синтаксис

```
Variable = TimeIntoInterval(TintoInt, Interval, Units)
-or-
```

```
If TimeIntoInterva1(TintoInt, Interval, Units)
```

TimeIsBetween

Определяет, находится ли значение часов реального времени CR1000 внутри временного интервала.

Синтаксис

```
TimeIsBetween(BeginTime, EndTime, Interval, Units)
```

A.5 Команды измерения

Читать далее. Дополнительная информация по записи данных, полученных по каналу RS-232 и TTL с выходов датчиков, приведена в разделах *Последовательный ввод-вывод* *(стр. 583)* и *Последовательный ввод-вывод (I/O)**(стр. 245)*.

A.5.1 Диагностика

Battery

Измеряет напряжение питания.

Синтаксис

```
Battery(Dest)
```

ComPortIsActive

Возвращает логическое значение, означающее выявление активности в COM-порту.

Синтаксис

```
variable = ComPortlsActive(ComPort)
```

InstructionTimes

Возвращает время выполнения каждой команды в программе.

Синтаксис

```
InstructionTimes(Dest)
```

PanelTemp

Измеряет температуру панели в °C.

Синтаксис

```
PanelTemp(Dest, Integ)
```

Signature

Возвращает сигнатуру кода программы в программу устройства регистрации.

Синтаксис

```
variable = Signature
```

A.5.2 Напряжение

VoltDiff

Измеряет разность потенциалов между входом высокого уровня и входом низкого уровня дифференциальных аналоговых входов.

Синтаксис

```
VoltDiff(Dest, Reps, Range, DiffChan, RevDiff, SettlingTime,
   Integ, Mult, Offset)
```

VoltSe

Измеряет напряжение на несимметричном входе относительно земли.

Синтаксис

```
VoltSe(Dest, Reps, Range, SEChan, MeasOfs, SettlingTime, Integ,
   Mult, Offset)
```

A.5.3 Термопары

Связанные темы:
• Измерения термопар — подробная информация
• Измерения термопар — команды

TCDiff

Производит измерение с помощью дифференциальной термопары.

Синтаксис

```
TCDiff(Dest, Reps, Range, DiffChan, TCType, TRef, RevDiff,
   SettlingTime, Integ, Mult, Offset)
```

TCSe

Производит измерение с помощью одноэлементной термопары.

Синтаксис

```
TCSe(Dest, Reps, Range, SEChan, TCType, TRef, MeasOfs,
   SettlingTime, Integ, Mult, Offset)
```

A.5.4 Измерения с помощью моста сопротивлений

Связанные темы:
• Измерения сопротивления — технические характеристики
• *Измерение сопротивления — общие сведения* *(стр. 67)*
• *Измерение сопротивления — подробная информация* *(стр. 337)*
• *Измерение сопротивления — команды* *(стр. 551)*

BrFull

Измеряет отношение V_{diff} / V_x для четырехпроводного полного моста. Передает значение 1000 • (V_{diff} / V_x).

Синтаксис

```
BrFull(Dest, Reps, Range, DiffChan, Vx/ExChan, MeasPEx, ExmV,
   RevEx, RevDiff, SettlingTime, Integ, Mult, Offset)
```

BrFull6W

Измеряет отношение V_{diff2} / V_{diff1} для шестипроводного полного моста. Передает значение 1000 • (V_{diff2} / V_{diff1}).

Синтаксис

```
BrFull6W(Dest, Reps, Range1, Range2, DiffChan, Vx/ExChan,
   MeasPEx, ExmV, RevEx, RevDiff, SettlingTime, Integ, Mult,
   Offset)
```

BrHalf

Измеряет напряжение на несимметричном выходе трехпроводной полумостовой схемы. Delay (задержка) является необязательным элементом.

Синтаксис

```
BrHalf(Dest, Reps, Range, SEChan, Vx/ExChan, MeasPEx, ExmV,
   RevEx, SettlingTime, Integ, Mult, Offset)
```

BrHalf3W

Измеряет отношение R_s / R_f для трехпроводной полумостовой схемы.

Синтаксис

```
BrHalf3W(Dest, Reps, Range, SEChan, Vx/ExChan, MeasPEx, ExmV,
   RevEx, SettlingTime, Integ, Mult, Offset)
```

BrHalf4W

Измеряет отношение R_s / R_f для четырехпроводной полумостовой схемы.

Синтаксис

```
BrHalf4W(Dest, Reps, Range1, Range2, DiffChan, Vx/ExChan,
   MeasPEx, ExmV, RevEx, RevDiff, SettlingTime, Integ, Mult,
   Offset)
```

A.5.5 Возбуждение

ExciteV

Эта команда устанавливает указанный потенциал в заданном канале возбуждения с переключаемым напряжением.

Синтаксис

```
ExciteV(Vx/ExChan, ExmV, XDelay)
```

SW12

Устанавливает переключаемый вывод 12 В перем. тока **SW12** в состояние высокого или низкого напряжения.

Синтаксис

```
SW12(Port)
```

A.5.6 Импульс и частота

Связанные темы:
• Импульсные измерения — технические характеристики
• *Импульсные измерения — общие сведения* *(стр. 68)*
• *Импульсные измерения — подробная информация* *(стр. 349)*
• *Импульсные измерения — команды* *(стр. 553)*

Примечание. Для измерения импульсов на выводах типа **C** может потребоваться резистор установки рабочей точки или подтягивающий к земле резистор. См. раздел *Выводы для измерения импульсов* *(стр. 352)***.**

PeriodAvg

Измеряет период сигнала на выводе типа **H/L]**, настроенном на несимметричный вход напряжения.

Синтаксис

```
PeriodAvg(Dest, Reps, Range, Terminal, Threshold, PAOption,
   Cycles, Timeout, Mult, Offset)
```

PulseCount

Измеряет число или частоту импульсов напряжения на клемме типа P или C, настроенной на ввод импульсов.

Синтаксис

```
PulseCount(Dest, Reps, Terminal, PConfig, POption, Mult,
   Offset)
```

VibratingWire

Производит измерения с помощью струнного счётчика. Эта команда считается устаревшей. Вместо нее используются модуль AVW200 и команда AVW200().

Синтаксис

```
VibratingWire(Dest, Reps, Range, SEChan, Vx/ExChan, StartFreq,
   EndFreq, TSweep, Steps, DelMeas, NumCycles, DelReps,
   Multiplier, Offset)
```

A.5.7 Цифровой ввод-вывод

CheckPort

Возвращает состояние настроенной на управление клеммы типа **C**.

Синтаксис

```
X = CheckPort(Port)
```

PortGet

Читает состояние настроенной на управление клеммы типа **C**.

Синтаксис

```
PortGet(Dest, Port)
```

PortsConfig

Конфигурирует выводы типа **C** для ввода или вывода.

Синтаксис

```
PortsConfig(Mask, Function)
```

ReadIO

Читает состояние клемм типа **C**.

Синтаксис

```
ReadIO(Dest, Mask)
```

TimerIO

Измеряет время между фронтами (переходами между состояниями) или частоту на выводах типа **C**.

Синтаксис

```
TimerIO(Dest, Edges, Function, Timeout, Units)
```

A.5.7.1 Управление

PortSet

Устанавливает высокий или низкий потенциал в указанном выводе типа **C**.

Синтаксис

```
PortSet(Terminal, State)
```

PulsePort

Изменяет состояние вывода типа **C**, по истечении тайм-аута изменяет состояние этого вывода и вводит вторую задержку.

Синтаксис

```
PulsePort(Terminal, Delay)
```

WriteIO

Задает состояние клемм типа **C**.

Синтаксис

```
WriteIO(Mask, Source)
```

A.5.7.2 Измерение

PWM

Реализует широтно-импульсную модуляцию на выводе типа **C**.

Синтаксис

```
PWM(Source, Terminal, Period, Units)
```

TimerIO

Измеряет интервал частот или частоту на выводе типа **C**.

Синтаксис

```
TimerIO(Dest, Edges, Function, Timeout, Units)
```

A.5.8 Поддержка датчика SDI-12 — команды

Связанные темы:
• *Поддержка датчика SDI-12 — общие сведения* *(стр. 72)*
• *Поддержка датчика SDI-12 — подробная информация* *(стр. 363)*
• *Последовательный интерфейс ввода/вывода: поддержка датчика SDI-12 — программные ресурсы* *(стр. 267)*
• *Поддержка датчика SDI-12 — команды* *(стр. 555)*

SDI12Recorder

Передает команды датчику SDI-12 и получает результаты от него.

Синтаксис

```
SDI12Recorder(Dest, Terminal, SDIAddress, SDICommand,
   Multiplier, Offset)
```

SDI12SensorSetup

Настраивает CR1000 на работу в качестве датчика SDI-12.

```
SDI12SensorSetup(Repetitions, SDIPort, SDIAddress,
   ResponseTime)
```

SDI12SensorResponse

Управляет передачей данных из CR1000 в устройство регистрации SDI-12.

Синтаксис

```
SDI12SensorResponse(SDI12Source)
```

A.5.9 Специализированные датчики

ACPower

Измеряет параметры питания и его параметры качества для однофазного питания, питания с расщепленной фазой и трехфазного питания по схеме «звезда». НЕ ВКЛЮЧАЙТЕ УСТРОЙСТВО CR1000 ПРЯМО В ЭЛЕКТРОСЕТЬ ПЕРЕМЕННОГО ТОКА.

Синтаксис

```
ACPower(DestAC, ConfigAC, LineFrq, ChanV, VMult, MaxVrms,
   ChanI, IMult, MaxIrms, Reps)
```

ОПАСНО! При подключении к электросети переменного тока возможен летальный исход. Вы несете ответственность за обеспечение того, что подключения к сети переменного тока соответствует применяемым электротехническим правилам и нормам. Для получения информации о доступных разделяющих трансформаторах обратитесь к инженеру по приложению в компанию Campbell Scientific.

CS110

Определяет напряженность электрического поля, используя измеритель напряженности электрического поля CS110.

Синтаксис

```
CS110(Dest, Leakage, Status, Integ, Mult, Offset)
```

CS110Shutter

Управляет затвором измерителя напряженности электрического поля CS110.

Синтаксис

```
CS110Shutter(Status, Move)
```

CS616

Включает рефлектометр для измерения содержания воды CS616 и производит измерения.

Синтаксис

```
CS616(Dest, Reps, SEChan, Port, MeasPerPort, Mult, Offset)
```

CS7500

Организует связь с датчиком содержания CO_2 and H_2O в открытой атмосфере CS7500. CS7500 и LI-7500 компании LI-COR — это один и тот же продукт.

Синтаксис

```
CS7500(Dest, Reps, SDMAddress, Command)
```

CSAT3

Обеспечивает связь с трехмерным акустическим анемометром CSAT3.

Синтаксис

```
CSAT3(Dest, Reps, SDMAddress, CSAT3Cmd, CSAT3Opt)
```

EC100

Обеспечивает связь с газовым анализатором для открытой атмосферы EC150 и канальным ИК газоанализатором EC155, используя SDM (мультиплексирование с пространственным разделением).

Синтаксис

```
EC100(Dest, SDMAddress, EC100Cmd)
```

EC100Configure

Настраивает газовый анализатор EC150 для открытой атмосферы и канальный ИК газоанализатор EC155.

Синтаксис

```
EC100Configure(Result, SDMAddress, ConfigCmd, DestSource)
```

GPS

Используется для оборудования GPS с целью обеспечения правильной работы часов CR1000 или предоставления другой информации GPS, например, местоположения и скорости. Для правильного выполнения этой команды возможно потребуется заменить на заводе встроенный аккумулятор.

Синтаксис

```
GPS(GPS_Array, ComPort, TimeOffsetSec, MaxErrorMsec,
   NMEA_Sentences)
```

Примечание. Для изменения используемой по умолчанию в GPS скорости передачи 38400 бод задайте новую скорость передачи, используя команду **SerialOpen()**.

HydraProbe

Считывает данные с датчика SDI-12 Hydra Probe компании Stevens Vitel.

Синтаксис

```
HydraProbe(Dest, SourceVolts, ProbeType, SoilType)
```

LI7200

Организует связь с газовым анализатором LI-7200 компании LI-COR для открытой атмосферы с датчиком CO_2 и H_2O.

Синтаксис

```
LI7200(Dest, Reps, SDMAddress, Command)
```

LI7700

Организует связь с газовым анализатором LI-7700 компании LI-COR для открытой атмосферы с датчиком CO_2 и H_2O.

Синтаксис

```
LI7200(Dest, Reps, SDMAddress, Command)
```

TGA

Производит измерения с помощью системы анализатора следов газов TGA100A.

Синтаксис

```
TGA(Dest, SDMAddress, DataList, ScanMode)
```

Therm107

Производит измерения с помощью термистора модели 107 компании Campbell Scientific.

Синтаксис

```
Therm107(Dest, Reps, SEChan, Vx/ExChan, SettlingTime, Integ,
   Mult, Offset)
```

Therm108

Производит измерения с помощью термистора модели 108 компании Campbell Scientific.

Синтаксис

```
Therm108(Dest, Reps, SEChan, Vx/ExChan, SettlingTime, Integ,
   Mult, Offset)
```

Therm109

Производит измерения с помощью термистора модели 109 компании Campbell Scientific.

Синтаксис

```
Therm109(Dest, Reps, SEChan, Vx/ExChan, SettlingTime, Integ,
   Mult, Offset)
```

A.5.9.1 Сеть беспроводных датчиков

ArrayIndex

Возвращает индекс именованного элемента массива.

Синтаксис

```
ArrayIndex(Name)
```

CWB100

Настраивает CR1000 на запрос и получение измерений от сети беспроводных датчиков CWB100.

Синтаксис

```
CWB100(ComPort, CWSDest, CWSConfig)
```

CWB100Diagnostics

Настраивает CR1000 на запрос и получение измерений от сети беспроводных датчиков CWB100.

Синтаксис

```
CWB100(ComPort, CWSDest, CWSConfig)
```

CWB100Routes

Возвращает измерения из сети беспроводных датчиков.

Синтаксис

```
CWB100Diagnostics(CWBPort, CWSDiag)
```

CWB100RSSI

Опрашивает амплитуду радиосигналов беспроводных датчиков в сети беспроводных датчиков.

Синтаксис

```
CWB100RSSI(CWBPort)
```

A.5.10 Поддержка периферийных устройств

В одной программе можно использовать несколько команд SDM.

AM25T

Управляет мультиплексором аналогового входного напряжения AM25T.

Синтаксис

```
AM25T(Dest, Reps, Range, AM25TChan, DiffChan, TCType, Tref,
   ClkPort, ResPort, VxChan, RevDiff, SettlingTime, Integ,
   Mult, Offset)
```

AVW200

Управляет измерительным прибором со струнным датчиком AVW200 *VSPECT* *(стр. 532)* и собирает данные с него.

Синтаксис

```
AVW200(Result, ComPort, NeighborAddr, PakBusAddr, Dest,
   AVWChan, MuxChan, Reps, BeginFreq, EndFreq, ExVolt,
   Therm50_60Hz, Multiplier, Offset)
```

CDM_VW300Config

Настраивает анализатор динамического спектра для струнного датчика CDM-VW300.

Синтаксис

```
CDM_VW300Config(DeviceType, CPIAddress, SysOptions, ChanEnable,
   ResonAmp, LowFreq, HighFreq, ChanOptions, Mult, Offset,
   SteinA, SteinB, SteinC, RF_MeanBins, RF_AmpBins, RF_LowLim,
   RF_HighLim, RF_Hyst, RF_Form)
```

CDM_VW300Dynamic

Собирает данные динамических измерений *VSPECT* *(стр. 532)* из анализатора динамического спектра для струнного датчика CDM-VW300.

Синтаксис

```
CDM_VW300Dynamic(CPIAddress, DestFreq, DestDiag)
```

CDM_VW300Rainflow

Собирает данные по дождю в виде гистограммы от измерительного периферийного оборудования со струнным датчиком CDM-VW300.

Синтаксис

```
CDM_VW300Rainflow(CPIAddress, RF1, RF2, RF3, RF4, RF5, RF6,
   RF7, RF8)
```

CDM_VW300Static

Собирает статические измерения *VSPECT* *(стр. 532)* от измерительного прибора со струнным датчиком CDM-VW300.

Синтаксис

```
CDM_VW300Static(CPIAddress, DestFreq, DestTherm, DestStdDev)
```

CPISpeed

Управляет скоростью передачи данных по шине CPI.

Синтаксис

```
CPISpeed(BitRate))
```

MuxSelect

Выбирает указанный канал мультиплексора.

Синтаксис

```
MuxSelect(ClkPort, ResPort, ClkPulseWidth, MuxChan, Mode)
```

SDMAO4

Устанавливает уровни выходного напряжения в устройстве с непрерывным аналоговым выходом SDM-AO4.

Синтаксис

```
SDMAO4(Source, Reps, SDMAdress)
```

SDMAO4A

Устанавливает уровни выходного напряжения в устройстве с непрерывным аналоговым выходом SDM-AO4A.

Синтаксис

```
SDMAO4A(Source, Reps, SDMAdress)
```

SDMCAN

Управляет интерфейсом SDM-CAN и читает данные из него.

Синтаксис

```
SDMCAN(Dest, SDMAddress, TimeQuanta, TSEG1, TSEG2, ID,
   DataType,
```

SDMCD16AC

Управляет управляющим устройством SDM-CD16AC, SDM-CD16 или SDM-CD16D.

Синтаксис

```
SDMCD16AC(Source, Reps, SDMAddress)
```

SDMCD16Mask

Управляет управляющим устройством SDM-CD16AC, SDM-CD16 или SDM-CD16D. В отличие от SDMCD16AC, эта команда позволяет CR1000 выбирать активацию портов с использованием маски. Часто используется с командой **TimedControl().**

Синтаксис

```
SDMCD16Mask(Source, Mask, SDMAddress)
```

SDMCVO4

Управляет четырехканальным устройством SDM-CVO4 с выходным током/напряжением.

Синтаксис

```
SDMCVO4(CVO4Source, CVO4Reps, SDMAddress, CVO4Mode)
```

SDMGeneric

Передает команды в устройство с SDM, которое не поддерживается операционной системой. См. приложение *Порядок следования байтов (стр. 643)*.

Синтаксис

```
SDMGeneric(Dest, SDMAddress, CmdByte, NumvaluesOut, Source,
   NumValuesIn, BytesPerValue, BigEndian, DelayByte)
```

SDMINT8

Управляет интервальным таймером SDM-INT8 и читает данные из него.

Синтаксис

```
SDMINT8(Dest, Address, Config8_5, Config4_1, Funct8_5,
   Funct4_1, OutputOpt, CaptureTrig, Mult, Offset)
```

SDMIO16

Настраивает модуль расширения ввода-вывода SDM-IO16 и производит измерения.

Синтаксис

```
SDMIO16(Dest, Status, Address, Command, Mode Ports 16 to 13,
   Mode Ports 12 to 9, Mode Ports 8 to 5, Mode Ports 4 to 1,
   Mult, Offset)
```

SDMSIO4

Управляет модулем расширения ввода-вывода SDM-SIO4, передачей данных из него и приемом данных этим модулем.

Синтаксис

```
SDMSIO4(Dest, Reps, SDMAddress, Mode, Command, Param1, Param2,
   ValuesPerRep, Multiplier, Offset)
```

SDMSpeed

Изменяет скорость, используемую CR1000 для синхронизации SDM (мультиплексирования с пространственным разделением).

Синтаксис

```
SDMSpeed(BitPeriod)
```

SDMSW8A

Управляет модулем расширения замыкания ключа SDM-SW8A и читает данные из него.

Синтаксис

```
SDMSW8A(Dest, Reps, SDMAddress, FunctOp, SW8AStartChan, Mult,
   Offset)
```

SDMTrigger

Обеспечивает синхронизацию при проведений измерений SDM для всех

устройств SDM.

Синтаксис

```
SDMTrigger
```

SDMX50

Управляет коаксиальным мультиплексором SDM-X50 без использования команды **TDR100()**.

Синтаксис

```
SDMX50(SDMAddress, Channel)
```

TDR100

Проводит измерения с помощью датчиков TDR, подключенных к рефлектометру во временной области TDR100 напрямую или через коаксиальный мультиплексор SDMX50.

Синтаксис

```
TDR100(Dest, SDMAddress, Option, Mux/ProbeSelect, WaveAvg, Vp,
   Points, CableLength, WindowLength, ProbeLength, ProbeOffset,
   Mult, Offset)
```

TimedControl

Позволяет устройству определения последовательности выполнения SDM управлять последовательностью фиксированных значений и длительностей. Это позволяет SDM-CD16x управлять событиями, используя точное время. См. приложение *Возбудители реле — список (стр. 649)*.

Синтаксис

```
TimedControl(Size, SyncInterval, IntervalUnits, DefaultValue,
   CurrentIndex, Source, ClockOption
```

A.6 Управление ПЛК — команды

Связанные темы:
- *Управление ПЛК — общие сведения (стр. 74)*
- *Управление ПЛК — подробная информация(стр. 244)*
- *Модули управления ПЛК — общие сведения (стр. 368)*
- *Модули управления ПЛК — списки (стр. 648)*
- *Контроль ПЛК — команды (стр. 562)*
- Переключаемое выходное напряжение — технические характеристики
- Переключаемое выходное напряжение — общие сведения
- *Переключаемое выходное напряжение — подробная информация (стр. 103)*

См. описание приведенных ниже команд в другой части этого приложения.

PortGet()
PortSet()
PulsePort()
ReadIO()
SDMAO4()

SDMAO4A()
SDMCD16AC()
SDMCD16Mask()
SDMCV04()
SDMIO16()
TimedControl()
ProcHiPri/EndProcHiPri
DNP()
DNPUpdate()
DNPVariable()
ModbusMaster()
ModbusSlave()

A.7 Команды обработки и математические команды

A.7.1 Математические операторы

Примечание. Объявление программы AngleDegrees() (см. *Объявления программы (стр. 537)*) приводит к тому, что в математических функциях аргументы задаются в градусах, а не в радианах.

A.7.2 Арифметические операторы

Таблица 136. Арифметические операторы

Символ	*Название*	*Комментарии*
^	Возведение в степень	Результат всегда предоставляется как тип данных *FLOAT (стр. 161)* , чтобы исключить проблемы, которые могут возникать при возведении целого числа в отрицательню степень. Однако если длина результата превышает 24 бита, то происходит потеря точности. Например, **(46340 ^ 2)** равно **2 147 395 584** (не совсем точно), в то время как **(46340 * 46340)** равно **2 147 395 600** (абсолютно точно) Лучше используйте последовательно операцию умножения вместо ^, если требуется 32-битовое предствление. Имеет те же возможности, что команда **PWR()** *(стр. 568)*.
*	Умножение	
/	Деление	Используйте команду **INTDV()** *(стр. 568)* для обеспечения точности 32-битового представления
+	Сложение	
-	Вычитание	
=	Равно	
<>	Не равно	
>	Больше	
<	Меньше	
>=	Больше или равно	
<=	Меньше или равно	

A.7.3 Побитовые операции

Побитовые операции сдвига (<< и >>) позволяют CRBasic работать с положениями битов в переменной, объявленной **As Long** (как целое). Ниже приведены примеры выражений и ожидаемые результаты:

- **&B00000001 << 1** дает **&B00000010** (десятичное число **2**)
- **&B00000010 << 1** дает **&B00000100** (десятичное число **4**)
- **&B11000011 << 1** дает **&B10000110** (десятичное число **134**)
- **&B00000011 << 2** дает **&B00001100** (десятичное число **12**)
- **&B00001100 >> 2** дает **&B00000011** (десятичное число **3**)

Результатом выполнения этих операторов является значение левого операнда, биты которого сдвинуты на указанное число позиций. Образующиеся «дырки» заполняются нулями.

Данные, передаваемые интеллектуальным датчиком или коммуникационным протоколом, могут представлять собой выходные данные, сжатые в целые числа, которые состоят из «упакованных» полей. Сжатие данных этого типа используется для экономии используемой памяти и повышения пропускной способности. Ниже приведен пример сжатия данных в восьмибитовое целое число.

> Упакованное целое, записанное в переменную ***input_val***, будет распаковано в три целых числа, которые будут сохранены как три отдельных числа: ***value_1, value_2*** и ***value_3***. В упакованном целом числе информация, которая распаковывается в число ***value_1***, хранится в битах 7 и 6, число ***value_2*** распаковывается из битов 5 и 4, а число ***value_3*** — из битов 3, 2, 1 и 0. Код на CRBasic для такой подпрограммы распаковки показан в примере CRBasic *Использование операторов битового сдвига* *(стр. 564)*.

Для целого числа без знака сдвиг влево эквивалентен умножению на два. Сдвиг вправо эквивалентен делению на два.

Ниже приведены операторы:

<<

Побитовый сдвиг влево

Синтаксис

```
Variable = Numeric Expression << Amount
```

>>

Побитовый сдвиг вправо

Синтаксис

```
Variable = Numeric Expression >> Amount
```

&

Побитовое И — побитово выполняет операцию И для переменной в выражении и возвращает это значение в переменную.

CRBasic: пример 70. Использование операторов побитового сдвига

```
'Пример программы показывает распаковку упакованного целого числа. Двоичное значение в
'переменой input_val распаковывается с получением трех целых значений, записываемых в отдельные
'переменные value(1), value(2) и value(3). Информация, распакованная в переменную value(1),
'хранится в битах 7 и 6 переменной input_val, переменная value(2) распаковывается из битов 5 и 4,
'а переменная value(3) — из битов с номерами от 3 до 0, причем нулевой бит — это LSB или самый
'младший бит.

Public input_val As Long = &B01100011
Public value(3) As Long

BeginProg

  'Распакуйте переменную input_val, маскируя все биты, кроме представляющей интерес группу
  'битов, используя
  'функцию И, а затем сдвигайте группу битов вправо до тех пор, пока самый правый бит займет
  'положение самого младшего бита. Результат представляет собой распакованное число в десятичном
  'виде.
  value(1) = (input_val AND &B11000000) >> 6
  value(2) = (input_val AND &B00110000) >> 4
  value(3) = (input_val AND &B00001111) 'Сдвиг не требуется, поскольку бит младшего разряда уже
                                        'находится в положении
                                        'самого младшего бита.

EndProg
```

A.7.4 Составные операторы присваивания

Таблица 137. Составные операторы присваивания

Символ	*Название*	*Функция*
^=	Присваивание экспоненты	Возводит значение переменной в заданную выражением степень и возвращает результат в эту переменную.
*=	Присвоение умножения	Умножает значение переменной на значение выражения и возвращает результат в эту переменную.
+=	Присвоение сложения	Добавляет значение переменной к значению выражения и возвращает результат в эту переменную. Кроме того, соединяет строковое выражение с переменной, объявленной как тип данных STRING. Присваивает результат переменной. См. пример в CRBasic *Конкатенация чисел и строк* *(стр. 284)*.
-=	Вычитание и присваивание	Вычитает значение выражения из значения переменной и присваивает результат переменной.
/=	Деление и присваивание	Делит значение переменной на значение выражения и присваивает результат переменной.
\=	Целочисленное деление и присваивание	Делит значение переменной на значение выражения и присваивает целочисленный результат переменной.

A.7.5 Логические операторы

AND (И)

Выполняет логическое объединение двух выражений.

Синтаксис

```
result = expr1 AND expr2
```

EQV

Выполняет проверку логической эквивалентности двух выражений.

Синтаксис

```
result = expr1 EQV expr2
```

NOT (НЕ)

Выполняет операцию логического отрицания выражения.

Синтаксис

```
result = NOT expression
```

OR (ИЛИ)

Выполняет логическое сложение двух выражений.

Синтаксис

```
result = expr1 OR expr2
```

XOR (исключающее ИЛИ)

Выполняет операцию логического исключения для двух выражений.

Синтаксис

```
result = expr1 XOR expr2
```

IIF

Вычисляет переменную или выражение и возвращает один из двух результатов в зависимости от результата расчета.

Синтаксис

```
Result = IIF (Expression, TrueValue, FalseValue)
```

IMP

Выполняет логическую импликацию для двух выражений.

Синтаксис

```
result = expression1 IMP expression2
```

A.7.6 Тригонометрические функции

A.7.6.1 Встроенные тригонометрические функции

ACOS

Возвращает арккосинус числа.

Синтаксис

```
x = ACOS(source)
```

ASIN

Возвращает арксинус числа.

Синтаксис

```
x = ASIN(source)
```

ATN

Возвращает арктангенс числа.

Синтаксис

```
x = ATN(source)
```

ATN2

Возвращает арктангенс y/x.

Синтаксис

```
x = ATN(y, x)
```

COS

Возвращает косинус угла, заданного в радианах.

Синтаксис

```
x = COS(source)
```

COSH

Возвращает гиперболический косинус выражения или величины.

Синтаксис

```
x = COSH(source)
```

SIN

Возвращает синус угла.

Синтаксис

```
x = SIN(source)
```

SINH

Возвращает гиперболический синус выражения или величины.

Синтаксис

```
x = SINH(Expr)
```

TAN

Возвращает тангенс угла.

Синтаксис

```
x = TAN(source)
```

TANH

Возвращает гиперболический тангенс выражения или величины.

Синтаксис

```
x = TANH(Source)
```

A.7.6.2 Производные тригонометрические функции

В таблице *Производные тригонометрические функции* *(стр. 568)* перечислены тригонометрические функции, которые могут быть получены из встроенных тригонометрических функций.

Таблица 138. Производные тригонометрические функции

Функция	*Эквивалент в CRBasic*
Секанс	Sec = 1 / Cos(X)
Косеканс	Cosec = 1 / Sin(X)
Котангенс	Cotan = 1 / Tan(X)
Арксеканс	Arcsec = Atn(X / Sqr(X * X - 1)) + Sgn(Sgn(X) - 1) * 1.5708
Арккосеканс	Arccosec = Atn(X / Sqr(X * X - 1)) + (Sgn(X) - 1) * 1.5708
Арккотангенс	Arccotan = Atn(X) + 1.5708
Гиперболический секанс	HSec = 2 / (Exp(X) + Exp(-X))
Гиперболический косеканс	HCosec = 2 / (Exp(X) - Exp(-X))
Гиперболический котангенс	HCotan = (Exp(X) + Exp(-X)) / (Exp(X) - Exp(-X))
Обратный гиперболический синус	HArcsin = Log(X + Sqr(X * X + 1))
Обратный гиперболический косинус	HArccos = Log(X + Sqr(X * X - 1))
Обратный гиперболический тангенс	HArctan = Log((1 + X) / (1 - X)) / 2
Обратный гиперболический секанс	HArcsec = Log((Sqr(-X * X + 1) + 1) / X)
Обратный гиперболический косеканс	HArccosec = Log((Sgn(X) * Sqr(X * X + 1) + 1) / X)
Обратный гиперболический котангенс	HArccotan = Log((X + 1) / (X - 1)) / 2

A.7.7 Арифметические функции

ABS

Возвращает модуль числа. Возвращает значение для типа данных Long (длинный), когда выражение имеет тип Long.

Синтаксис

```
x = ABS(source)
```

Ceiling

Округляет значение до верхнего целочисленного значения.

Синтаксис

```
variable = Ceiling(Number)
```

EXP

Возвращает значение e (основание натурального логарифма), возведенное в степень.

Синтаксис

```
x = EXP(source)
```

Floor

Округляет значение до нижнего целочисленного значения.

Синтаксис

```
variable = Floor(Number)
```

FRAC

Возвращает дробную часть числа.

Синтаксис

```
x = FRAC(source)
```

INT или FIX

Возвращает целую часть числа.

Синтаксис

```
x = INT(source)
x = Fix(source)
```

INTDV

Выполняет целочисленное деление двух чисел.

Синтаксис

```
X INTDV Y
```

LN или LOG

Возвращает значение натурального логарифма числа. Ln и Log выполняют одинаковую функцию.

Синтаксис

```
x = LOG(source)
x = LN(source)
```

Примечание. LOGN = LOG(X) / LOG(N)

LOG10

Функция L0G10 возвращает значение десятичного логарифма числа.

Синтаксис

```
x = LOG10 (number)
```

MOD

Деление по модулю. Делит одно число на другое и возвращает только остаток.

Синтаксис

```
result = operand1 MOD operand2
```

PWR

Выполняет возведение в степень переменной. Такая же функция, как у оператора. См. раздел *Арифметические операторы* *(стр. 563)*.

Синтаксис

```
PWR(X, Y)
```

RectPolar

Преобразует прямоугольные координаты в полярные.

Синтаксис

```
RectPolar(Dest, Source)
```

Round

Округляет величину до верхнего или нижнего значения.

Синтаксис

```
variable = Round (Number, Decimal)
```

SGN

Определяет знак числа.

Синтаксис

```
x = SGN(source)
```

Sqr

Возвращает квадратный корень из числа.

Синтаксис

```
x = SQR(number)
```

A.7.8 Интегрированная обработка данных

DewPoint

Вычисляет точку росы (°C) по температуре сухого термометра и относительной влажности.

Синтаксис

```
DewPoint(Dest, Temp, RH)
```

PRT

Вычисляет температуру (°C) на основе сопротивления резистивного датчика температуры. Настоящая команда в большинстве приложений заменена на **PRTCalc()**.

Синтаксис

```
PRT(Dest, Reps, Source, Mult)
```

PRTCalc

Вычисляет температуру на основе сопротивления резистивного датчика температуры в соответствии с серией альтернативных стандартов, в том числе Международной электротехнической комиссии МЭК (IEC). Заменяет функцию **PRT()** в большинстве приложений.

Синтаксис

```
PRTCalc(Dest, Reps, Source, PRTType, Mult, Offset)
```

SolarPosition

Вычисляет положение солнца.

Синтаксис

```
SolarPosition(Dest, Time, UTC_OFFSET, Lat_c, Lon_c, Alt_c,
   Pressure, AirTemp)
```

SatVP

Вычисляет давление насыщенного пара (кПа) по температуре.

Синтаксис

```
SatVP(Dest, Temp)
```

StrainCalc

Преобразует результат команды выполнения мостовых измерений в микродеформацию.

Синтаксис

```
StrainCalc(Dest, Reps, Source, BrZero, BrConfig, GF, v)
```

VaporPressure

Вычисляет давление пара по температуре и относительной влажности.

Синтаксис

```
VaporPressure(Dest, Temp, RH)
```

WetDryBulb

Вычисляет давление пара (кПа) по температурам влажного и сухого термометров и барометрическое давление.

Синтаксис

```
WetDryBulb(Dest, DryTemp, WetTemp, Pressure)
```

A.7.9 Пространственная обработка данных

AvgSpa

Вычисляет пространственное среднее значение среди величин исходного массива.

Синтаксис

```
AvgSpa(Dest, Swath, Source)
```

CovSpa

Вычисляет пространственную ковариацию данные статистических рядов.

Синтаксис

```
CovSpa(Dest, NumOfCov, SizeOfSets, CoreArray, DatArray)
```

FFTSpa

Выполняет **F**ast **F**ourier **T**ransform (быстрое преобразование Фурье**)** с временным рядом измерений.

Синтаксис

```
FFTSpa(Dest, N, Source, Tau, Units, Option)
```

MaxSpa

Находит максимальное значение в массиве.

Синтаксис

```
MaxSpa(Dest, Swath, Source)
```

MinSpa

Находит минимальное значение в массиве.

Синтаксис

```
MinSpa(Dest, Swath, Source)
```

RMSSpa

Вычисляет среднеквадратическое значение для массива.

Синтаксис

```
RMSSpa(Dest, Swath, Source)
```

SortSpa

Сортирует элементы массива по возрастанию.

Синтаксис

```
SortSpa(Dest, Swath, Source)
```

StdDevSpa

Определяет среднеквадратическое отклонение для массива.

Синтаксис

```
StdDevSpa(Dest, Swath, Source)
```

A.7.10 Другие функции

AddPrecise

Позволяет выполнять высокоточное суммирование переменных или обрабатывать высокоточные переменные вместе с функцией **MovePrecise.**

Синтаксис

```
AddPrecise(PrecisionVariable, X)
```

AvgRun

Сохраняет скользящее среднее значение измерения.

Синтаксис

```
AvgRun(Dest, Reps, Source, Number)
```

Примечание. Функцию **AvgRun()** не следует вводить в логическую структуру **For / Next** с индексированными параметрами *Source* и *Dest* и параметром *Reps*, который задан равным 1. По сути, если это выполнить, то скользящее среднее будет определяться, используя значения различных элементов массива, вместо расчета независимого скользящего среднего значения по каждому элементу массива. Результаты будут представлять собой скользящее среднее пространственного среднего значения по различным элементам исходного массива.

Randomize

Инициализирует генератор случайных чисел.

Синтаксис

```
Randomize(source)
```

RND

Генерирует случайное число.

Синтаксис

```
RND(source)
```

TotalRun

Выдает промежуточную сумму для измерения.

Синтаксис

```
TotalRun(Dest, Reps, Source, Number, RunReset)
```

A.7.10.1 Гистограммы

Histogram

Представляет входные данные в виде стандартной гистограммы (частотное распределение) или в виде гистограммы взвешенных величин.

Синтаксис

```
Histogram(BinSelect, DataType, DisableVar, Bins, Form, WtVal,
   LoLim, UpLim)
```

Histogram4D

Обрабатывает входные данные в виде стандартной гистограммы (частотное распределение) или в виде гистограммы взвешенных величин с количеством измерений до четырех.

Синтаксис

```
Histogram4D(BinSelect, Source, DataType, DisableVar, Bins1,
   Bins2, Bins3, Bins4, Form, WtVal, LoLim1, UpLim1, LoLim2,
   UpLim2, LoLim3, UpLim3, LoLim4, UpLim4)
```

LevelCrossing

Обрабатывает данные в виде одномерной или двумерной гистограммы, используя алгоритм расчета с пересечением уровней.

Синтаксис

```
Leve1Crossing(Source, DataType, DisableVar, NumLevels, 2ndDim,
   CrossingArray, 2ndArray, Hysteresis, Option)
```

RainFlow

Обрабатывает данные с помощью алгоритма расчета на основе метода дождевого потока, необходимого для оценки накопленного усталостного повреждения компонентов, подвергающихся воздействию нагрузки или циклической деформации. См. статью Downing S. D., Socie D. F. (1982) Simple Rainflow Counting Algorithms. International Journal of Fatigue Volume 4, Issue 1.

Синтаксис

```
RainFlow(Source, DataType, DisableVar, MeanBins, AmpBins,
   Lowlimit, Highlimit, MinAmp, Form)
```

A.8 Строковые функции

Связанные темы
• *Строковые операции* *(стр. 282)*

& Объединяет строковые переменные.
+ Объединяет строковые и числовые переменные.
– Сравнивает две строки, возвращает ноль, если они идентичны.

A.8.1 Строковые операции

Таблица 139. Строковые операции

Эксплуатация	*Комментарии*
Строковые константы	Постоянные строки могут вставляться в выражения, используя кавычки. Например: `FirstName = "Mike"`
Сложение строк	Строки могут объединяться, используя оператор «+». Например: `FullName = FirstName + " " + MiddleName + " " + LastName`
Вычитание строк	**String1-String2** дает целочисленный результат в диапазоне от **–255** до **+255.**
Преобразование строк в числа и наоборот	Преобразование строк в числа и чисел в строки выполняется автоматически, когда при необходимости строке присваивают числовое значение или числовой переменной присваивают строку.

Таблица 139. Строковые операции	
Операторы сравнения строк	Со строками работают операторы сравнения =, >, <, <>, >= и <=.
Процесс вывода выходных строковых данных	Команда **Sample()** преобразует типы данных, если исходный тип данных отличается от типа данных в **Sample()**. Использование строк запрещено во всех командах процесса вывода, за исключением **Sample().**

A.8.2 Строковые команды

ArrayLength

Возвращает длину массива переменных.

Синтаксис

```
ArrayLength(Variable)
```

ASCII

Возвращает код ASCII/ANSI (Американский стандартный код для обмена информацией / Американский национальный институт стандартов) для символа в строке.

Синтаксис

```
Variable = ASCII (ASCIIString(1,1,X))
```

CheckSum

Возвращает контрольную сумму для символов в строке.

Синтаксис

```
Variable = CheckSum(ChkSumString, ChkSumType, ChkSumSize)
```

CHR

Вводит символ ANSI в строку.

Синтаксис

```
CHR(Code)
```

Erase

Стирает все байты в переменной или в массиве переменных.

Синтаксис

```
Erase(EraseVar)
```

FormatFloat

Преобразует величину с плавающей точкой в строку. Заменено на **SPrintF().**

Синтаксис

```
String = FormatFloat(Float, FormatString)
```

FormatLong

Преобразует величину в формате LONG в строку. Заменено на **SPrintF().**

Синтаксис

```
String = FormatLong(Long, FormatString)
```

FormatLongLong

Преобразует целое 64-битное число в формате LONG в десятичное значение в формате строковой переменной.

Синтаксис

```
FormatLongLong(LongLongVar(1))
```

HEX

Возвращает выражение в виде шестнадцатиричной строки.

Синтаксис

```
Variable = HEX(Expression)
```

HexToDec

Преобразует шестнадцатиричную строку в целое число или число с плавающей точкой.

Синтаксис

```
Variable = HexToDec(Expression)
```

InStr

Определяет положение строковой переменной в строке.

Синтаксис

```
Variable = InStr(Start, SearchString, FilterString,
   SearchOption)
```

LTrim

Возвращает копию строки без начальных пробелов.

Синтаксис

```
variable = LTrim(TrimString)
```

Left

Возвращает подстроку, которая представляет собой определенный ряд символов с левой стороны исходной строки.

Синтаксис

```
variable = Left(SearchString, NumChars)
```

Len

Возвращает количество байтов в строке.

Синтаксис

```
Variable = Len(StringVar)
```

LowerCase

Преобразует строку в символы нижнего регистра.

Синтаксис

```
String = LowerCase(SourceString)
```

Mid

Возвращает подстроку, входящую в строку.

Синтаксис

```
String = Mid(SearchString, Start, Length)
```

Replace

Находит строку для подстроки и заменяет эту подстроку на другую строку.

Синтаксис

```
variable = Replace(SearchString, SubString, ReplaceString)
```

Right

Возвращает подстроку, которая представляет собой определенный ряд символов с правой стороны исходной строки.

Синтаксис

```
variable = Right(SearchString, NumChars)
```

RTrim

Возвращает копию строки без концевых пробелов.

Синтаксис

```
variable = RTrim(TrimString)
```

StrComp

Сравнивает две строки путем вычитания символов в одной строке из символов в другой строке.

Синтаксис

```
Variable = StrComp(String1, String2)
```

SplitStr

Разбивает существующую строку на одну или несколько строк или числовых переменных.

Синтаксис

```
SplitStr(SplitResult, SearchString, FilterString, NumSplit,
   SplitOption)
```

SPrintF

Преобразует данные в отформатированные строки. Возвращает длину отформатированной строки. Заменяет функции **FormatFloat()** и **FormatLong().**

Синтаксис

```
length = SPrintF(Destination, format,...)
```

Trim

Возвращает копию строки без начальных или концевых пробелов.

Синтаксис

```
variable = Trim(TrimString)
```

UpperCase

Преобразует строку в символы верхнего регистра.

Синтаксис

```
String = UpperCase(SourceString)
```

A.9 Регистрация времени — команды

Связанные темы:
• *Хронометраж — общие сведения (стр. 75)*
• *Хронометраж — команды (стр. 578)*

В CR1000 время сохраняется в виде целых чисел в секундах и наносекундах с полуночи 1 января 1990 г.

ClockChange

Возвращает миллисекунды в изменении времени, обусловленном любой настройкой часов, выполненной с момента последнего выполнения функции **ClockChange.**

Синтаксис

```
variable = ClockChange
```

ClockReport

Отправляет значение времени CR1000 в удаленный регистратор данных в сети PakBus.

Синтаксис

```
ClockReport(ComPort, RouterAddr, PakBusAddr)
```

ClockSet

Устанавливает часы CR1000 на основе значений из массива.

Синтаксис

```
ClockSet(Source)
```

DaylightSaving

Задает летнее время. Определяет начало и конец летнего времени. На выбор передвигает вперед или возвращает назад на один час время в CR1000.

Синтаксис

```
variable = DaylightSaving(DSTSet, DSTnStart, DSTDayStart,
   DSTMonthStart, DSTnEnd, DSTDayEnd, DSTMonthEnd, DSTHour)
```

DaylightSavingUS

Определяет начало и конец летнего времени в США. На выбор передвигает вперед или возвращает назад на один час время в CR1000.

Синтаксис

```
variable = DaylightSavingUS(DSTSet)
```

IfTime

Возвращает число, означающее **True (-1)** или **False (0)**, используя часы реального времени CR1000.

Синтаксис

```
If (IfTime(TintoInt, Interval, Units)) Then

-or-

Variable = IfTime(TintoInt, Interval, Units)
```

PakBusClock

Устанавливает на часах CR1000 время, заданное в устройстве сети PakBus.

Синтаксис

```
PakBusClock(PakBusAddr)
```

RealTime

Определяет год, месяц, день, час, минуты, секунды, микросекунды, день недели и (или) день в году по часам CR1000.

Синтаксис

```
RealTime(Dest)
```

SecsSince1990

Возвращает секунды, прошедшие с 1990 г. Тип данных — LONG. Используют с **GetRecord().**

Синтаксис

```
SecsSince1990(date, option)
```

TimeIntoInterval

Возвращает число, означающее **True (-1)** или **False (0)** на основании показаний часов реального времени устройства регистрации.

Синтаксис

```
Variable = TimeIntoInterva1(TintoInt, Interval, Units)
-or-
```

```
If TimeIntoInterval(TintoInt, Interval, Units)
```

TimeIsBetween

Определяет, находится ли значение часов реального времени CR1000 внутри временного интервала.

Синтаксис

```
TimeIsBetween(BeginTime, EndTime, Interval, Units)
```

Timer

Возвращает значение таймера.

Синтаксис

```
variable = Timer(TimNo, Units, TimOpt)
```

A.10 Команды модема с поддержкой голосовых сообщений

Примечание. Обратитесь к руководствам для модема с поддержкой голосовых сообщений компании Campbell Scientific, которые доступны на сайте *www.campbellsci.com/manuals* (*http://www.campbellsci.com/manuals*).

DialVoice

Определяет строку набора для модема COM310 с поддержкой голосовых сообщений.

Синтаксис

```
DialVoice(DialString)
```

VoiceBeg / EndVoice

Отмечает начало и конец звукового кода, исполняемого, когда CR1000 обнаруживает вызов от модема с поддержкой голосовых сообщений.

Синтаксис

```
VoiceBeg
   [voice code to be executed]
EndVoice
```

VoiceHangup

Отсоединяет модем с поддержкой голосовых сообщений

Синтаксис

```
VoiceHangup
```

VoiceKey

Распознает возврат символов от **1** до **9**, * или #. Зачастую используется для добавления задержки, которая предоставляет время для произнесения сообщения, в последовательность **VoiceBegin/EndVoice**.

Синтаксис

```
VoiceKey(TimeOut*IDH_Popup_VoiceKey_Timeout)
```

VoiceNumber

Возвращает одно или несколько чисел (от 1 до 9), оканчивающихся ключом # или *.

Синтаксис

```
VoiceNumber(TimeOut*IDH_POPUP_VoiceKey_Timeout)
```

VoicePhrases

Выдает перечень фраз для функции **VoiceSpeak().**

Синтаксис

```
VoicePhrases(PhraseArray, Phrases)
```

VoiceSetup

Управляет отсоединением модема COM310 с поддержкой голосовых сообщений.

Синтаксис

```
VoiceSetup(HangUpKey, ExitSubKey, ContinueKey, SecsOnLine,
   UseTimeout, CallOut)
```

VoiceSpeak

Задает звуковую строку, которую необходимо произнести с помощью модема с поддержкой голосовых сообщений.

Синтаксис

```
VoiceSpeak("String" + Variable + "String"..., Precision)
```

A.11 Пользовательские меню — команды

Связанные темы:
• *Пользовательские меню — общие сведения* *(стр. 84, стр. 581)*
• *Экран данных: пользовательские меню — подробная информация* *(стр. 182).*
• *Пользовательские меню — набор команд* *(стр. 581).*
• *Экранная клавиатура — общие сведения* *(стр. 83)*
• *Справка к редактору CRBasic* для **DisplayMenu()**

Пользовательские меню разработаны с использованием следующего синтаксиса, используемого перед командой **BeginProg**.

```
DisplayMenu("MenuName", AddToSystem)
   MenuItem("MenuItemName", Variable)
   MenuPick(Item1, Item2, Item3...)
   DisplayValue("MenuItemName", tablename.fieldname)
   SubMenu(MenuName)
     MenuItem("MenuItemName", Variable)
   EndSubMenu
EndMenu

BeginProg
   [program body]
EndProg
```

DisplayLine

Отображает полную линейку текста, предназначенного только для чтения, в пользовательском меню.

Синтаксис:

```
DisplayLine(Value)
```

DisplayMenu / EndMenu

Отмечает начало и конец пользовательского меню.

Синтаксис:

```
DisplayMenu("MenuName", AddToSystem)
   [menu definition]
EndMenu
```

DisplayValue

Определяет имя и соответствующее табличное значение или переменную для отдельного пункта в пользовательском меню.

Синтаксис:

```
DisplayValue("MenuItemName", Expression)
```

MenuItem

Определяет имя и соответствующее значение измерения для отдельного пункта в пользовательском меню.

Синтаксис:

```
MenuItem("MenuItemName", Variable)
```

MenuPick

Создает перечень выбираемых опций, которые могут использоваться при редактировании значения функции **MenuItem()**.

Синтаксис:

```
MenuPick(Item1, Item2, Item3...)
```

MenuRecompile

Создает пункт пользовательского меню для перекомпилирования программы после внесения изменений в одно или несколько значений функции **ConstTable()**.

Синтаксис

```
MenuRecompile("CompileString", CompileVar)
```

SubMenu / EndSubMenu

Устанавливает начало и конец меню второго уровня для пользовательского меню.

Синтаксис:

```
DisplayMenu("MenuName", 100)
   SubMenu("MenuName")
     [menu definition]
   EndSubMenu
EndMenu
```

A.12 Последовательный ввод/вывод

Читать далее. См. *Последовательный ввод/вывод (стр. 245)*.

MoveBytes

Перемещает двоичные байты данных в другое место памяти при переводе данных из формата «от старшего к младшему» в «от младшего к старшему». См. приложение *Порядок следования байтов (стр. 643)*.

Синтаксис

```
MoveBytes(Destination, DestOffset, Source, SourceOffset,
   NumBytes)
```

SerialBrk

Отправляет сигнал размыкания заданной длительности в последовательный порт CR1000.

Синтаксис

```
SerialBrk(Port, Duration)
```

SerialClose

Закрывает коммуникационный порт, предварительно открытый с помощью функции **SerialOpen().**

Синтаксис

```
SerialClose(ComPort)
```

SerialFlush

Удаляет любые символы из буфера последовательного ввода.

Синтаксис

```
SerialFlush(ComPort)
```

SerialIn

Настраивает коммуникационный порт для приема входящих последовательных данных.

Синтаксис

```
SerialIn(Dest, ComPort, TimeOut, TerminationChar, MaxNumChars)
```

SerialInBlock

Сохраняет входящие последовательные данные. Эта функция возвращает количество полученных байтов.

Синтаксис

```
SerialInBlock(ComPort, Dest, MaxNumberBytes)
```

SerialInChk

Возвращает количество символов доступных в последовательном буфере регистратора данных.

Синтаксис

```
SerialInChk(ComPort)
```

SerialInRecord

Читает входящие через COM-порт последовательные данные и сохраняет данные в целевую переменную.

Синтаксис

```
SerialInRecord(COMPort, Dest, BeginWord, NBytes, EndWord,
   NBytesReturned, LoadNAN)
```

SerialOpen

Настраивает порт регистратора данных для связи с устройством, не входящим в сеть PakBus.

Синтаксис

```
SerialOpen(ComPort, BaudRate, Format, TXDelay, BufferSize)
```

SerialOut

Передает строку через коммуникационный порт регистратора данных.

Синтаксис

```
SerialOut(ComPort, OutString, WaitString, NumberTries, TimeOut)
```

SerialOutBlock

Отправляет двоичные данные по коммуникационному порту. Поддерживает прозрачную последовательную передачу.

Синтаксис

```
SerialOutBlock(ComPort, Expression, NumberBytes)
```

A.13 Одноранговая сеть PakBus® Система связи

Связанные темы:
• *Связь PakBus® — общие сведения* *(стр. 88)*
• *Связь PakBus® — подробная информация* *(стр. 393)*
• *Связь PakBus® — команды* *(стр. 584)*
• *Руководство по работе в сети PakBus (*доступно на сайте *www.campbellsci.com/manuals (http://www.campbellsci.com/manuals))*

PakBus является собственным сетевым протоколом, разработанным для оптимизации системы связи между регистраторами данных и периферийным оборудованием компании Campbell Scientific. Сеть PakBus отличается свойствами автоматического обнаружения и самовосстановления. Далее приведен перечень команд CRBasic, управляющих процессом работы сети PakBus. Некоторые команды PakBus задают адрес сети PakBus (***PakBusAddr***) или COM-порт (***ComPort***). ***PakBusAddr*** может быть переменной CRBasic. ***ComPort*** является константой. ***ComPort*** имеет следующие аргументы:

- **ComRS232**
- **ComME**
- **Com310**
- **ComSDC7**
- **ComSDC8**
- **ComSDC10**

- **ComSDC11**
- **Com1** (C1,C2)
- **Com2** (C3,C4)
- **Com3** (C5,C6)
- **Com4** (C7,C8)
- **Com32–Com46** (доступны при использовании модуля расширения ввода/вывода с одним каналом. См. приложение*Перечень модулей последовательного ввода/вывода* *(стр. 646)*)

Скорость передачи данных в бодах по асинхронным портам (ComRS232, ComME, Com1, Com2, Com3, Com4 и Com32-Com46) по умолчанию установлена 9600, если она не задана другим способом с помощью команды **SerialOpen()** или если порт не открывается по факту получения входящего пакета сети PakBus, имеющего иную скорость передачи данных. См. таблицу *Скорости передачи данных по асинхронным портам* *(стр. 588)*.

В общем случае команды PakBus записывают результирующий код в переменную, указывающую на успешность или сбой передачи. Успешность соответствует результирующему коду **0**. В противном случае результирующий код возрастает. Если коммуникационное взаимодействие успешно, но обнаружена ошибка, устанавливается отрицательный результирующий код. См. *Справку к редактору CRBasic* для получения разъяснений по кодам ошибок. Для команд, возвращающих результирующий код, повторные передачи могут быть запрограммированы с помощью логики CRBasic, как показано в следующем фрагменте кода:

```
For I = 1 to 3
   GetVariable (ResultCode ,...)
   If ResultCode = 0 Exit For
Next
```

Аргумент ***Timeout*** задается в сотых долях секунды (0,01). Если ввести **0**, тогда будет использоваться тайм-аут, заданный по умолчанию и определяющийся временем передачи по наилучшему маршруту. Используйте *PakBusGraph* *(стр. 654)* **Параметры повторной пересылки** для расчета этого времени. Поскольку эти команды коммуникационного взаимодействия ожидают реакции или тайм-аута перед тем, как программа перейдет к следующей команде, они должны использоваться в *медленной последовательности* *(стр. 157)*. Медленная последовательность не будет препятствовать выполнению основного сканирования или другим медленным последовательностям. Опционально перед параметром ***ComPort*** можно поставить тире, например, ***-ComME***, которое позволит команде не ждать реакции или тайм-аута. Это ускорит выполнение команды; однако любые запрашиваемые ею данные и результирующий код будут записаны только после завершения коммуникационного взаимодействия.

AcceptDataRecords

Настраивает CR1000 на прием и сохранение записей от удаленного регистратора PakBus.

Синтаксис

```
AcceptDataRecords(PakBusAddr, TableNo, DestTableName)
```

Broadcast

Отправляет широковещательное сообщение в сеть PakBus.

Синтаксис

```
Broadcast(ComPort, Message)
```

ClockReport

Отправляет значение времени регистратора данных в удаленный регистратор данных в сети PakBus.

Синтаксис

```
ClockReport(ComPort, RouterAddr, PakBusAddr)
```

DataGram

Инициализирует приложение SerialServer / DataGram / PakBus в регистраторе данных, когда компилируется программа.

Синтаксис

```
DataGram(ComPort, BaudRate, PakBusAddr, DestAppID, SrcAppID)
```

DialSequence / EndDialSequence

Задает код необходимый для передачи пакетов в устройство сети PakBus.

Синтаксис

```
DialSequence(PakBusAddr)
   DialSuccess = DialModem(ComPort, DialString, ResponseString)
EndDialSequence(DialSuccess)
```

EncryptExempt

Определяет один или несколько адресов PakBus, на которые регистратор данных не будет посылать зашифрованные сообщения PakBus, даже если шифрование PakBus включено.

Синтаксис

```
EncryptExempt(BeginPakBusAddr, EndPakBusAddr)
```

GetDataRecord

Извлекает самую последнюю запись из таблицы данных удаленного регистратора данных сети PakBus и сохраняет запись в CR1000.

Синтаксис

```
GetDataRecord(ResultCode, ComPort, NeighborAddr, PakBusAddr,
   Security, Timeout, Tries, TableNo, DestTableName)
```

Примечание. Регистраторы данных CR200, CR510PB, CR10XPB и CR23XPB не отвечают на запрос GetDataRecord, отправленный от других регистраторов данных сети PakBus.

GetFile

Получает файл от другого регистратора данных сети PakBus.

Синтаксис

```
GetFile(ResultCode, ComPort, NeighborAddr, PakBusAddr,
   Security, TimeOut, "LocalFile", "RemoteFile")
```

GetVariables

Извлекает значения переменных или массива переменных из таблицы данных регистратора данных сети PakBus.

Синтаксис

```
GetVariables(ResultCode, ComPort, NeighborAddr, PakBusAddr,
   Security, TimeOut, "TableName", "FieldName", Variable,
   Swath)
```

Network

Вместе с командой **SendGetVariables** конфигурирует необходимые регистраторы данных в сети PakBus для отправки и приема данных от главного узла.

Синтаксис

```
Network(ResultCode, Reps, BeginAddr, TimeIntoInterval,
   Interval, Gap, GetSwath, GetVariable, SendSwath,
   SendVariable)
```

PakBusClock

Устанавливает на часах регистратора данных время, заданное в устройстве сети PakBus.

Синтаксис

```
PakBusClock(PakBusAddr)
```

Route

Возвращает адрес (или маршрут) соседнего регистратора данных сети PakBus.

Синтаксис

```
variable = Route(PakBusAddr)
```

RoutersNeighbors

Возвращает в CR1000 перечень всех маршрутизаторов сети PakBus и всех известных соседних узлов.

Синтаксис

```
RoutersNeighbors(DestArray(MaxRouters, MaxNeighbors+1))
```

Routes

Возвращает перечень известных динамических маршрутизаторов для регистратора данных PakBus, которые сконфигурированы как маршрутизаторы в сети PakBus.

Синтаксис

```
Routes(Dest)
```

SendData

Отправляет самую последнюю запись из таблицы данных в удаленное устройство сети PakBus.

Синтаксис

```
SendData(ComPort, RouterAddr, PakBusAddr, DataTable)
```

SendFile

Отправляет файл в другой регистратор данных сети PakBus.

Синтаксис

```
SendFile(ResultCode, ComPort, NeighborAddr, PakBusAddr,
   Security, TimeOut, "LocalFile", "RemoteFile")
```

SendGetVariables

Отправляет массив значений в главный регистратор данных сети PakBus и извлекает массив данных из главного регистратора данных.

Синтаксис

```
SendGetVariables(ResultCode, ComPort, RouterAddr, PakBusAddr,
   Security, TimeOut, SendVariable, SendSwath, GetVariable,
   GetSwath)
```

SendTableDef

Отправляет табличные описания из таблицы данных в удаленное устройство сети PakBus.

Синтаксис

```
SendTableDef(ComPort, RouterAddr, PakBusAddr, DataTable)
```

SendVariables

Отправляет значение или значения переменной или массива переменных в таблицу данных удаленного регистратора данных.

Синтаксис

```
SendVariables(ResultCode, ComPort, RouterAddr, PakBusAddr,
   Security, TimeOut, "TableName", "FieldName", Variable,
   Swath)
```

StaticRoute

Задает статический маршрут для регистратора данных сети PakBus.

Синтаксис

```
StaticRoute(ComPort, NeighborAddr, PakBusAddr)
```

TimeUntilTransmit

Возвращает оставшееся время в секундах до коммуникационного взаимодействия с главным регистратором данных.

Синтаксис

```
variable = TimeUntilTransmit
```

Таблица 140. Скорости передачи данных для асинхронных портов

Скорость	*Комментарии*
-nnnn (начало автоматического обнаружения передачи[1] при nnnn)	начало автоматического обнаружения передачи[1] при nnnn
0	начало автоматического обнаружения передачи при 9600
300 1200 4800	
9600	по умолчанию
19200 38400 57600 115200	

Таблица 140. Скорости передачи данных для асинхронных портов
[1] Автоматическое обнаружение передачи: измерения выполняются по факту наличия сигнала в канале связи, и скорость передачи данных устанавливается CR1000.

A.14 Управление переменными

Arraylndex

Возвращает индекс именованного элемента массива.

Синтаксис

```
variable = Arraylndex(Name)
```

ArrayLength

Возвращает длину массива переменных. В случае переменных типа STRING (строка) возвращается общее количество символов, которые может содержать массив строк.

Синтаксис

```
ArrayLength(Variable)
```

Erase

Стирает все байты в переменной или в массиве переменных.

Синтаксис

```
Erase(EraseVar)
```

FindSpa

Выполняет поиск исходного массива для значения и возвращает позицию значения в массиве.

Синтаксис

```
FindSpa(SoughtLow, SoughtHigh, Step, Source)
```

Move

Перемещает значения в диапазоне изменения переменных для различных переменных или заполняет диапазон изменения переменных константой.

Синтаксис

```
Move(Dest, DestReps, Source, SourceReps)
```

A.15 Управление файлами

Команды для доступа и управления файлами сохранены в памяти CR1000.

CalFile

Сохраняет переменные данные, такие как данные калибровки датчика, из программы в файл, записанный в энергонезависимой памяти CR1000. Функция **CalFile()** датирована ранним числом и не используется с функцией **FieldCal()**.

Синтаксис

```
CalFile(Source/Dest, NumVals, "Deviceifilename", Option)
```

FileCopy

Копирует файл с одного накопителя на другой.

Синтаксис

```
FileCopy(FromFileName, ToFileName)
```

FileClose

Закрывает описатель файла, созданного с помощью функции **FileOpen().**

Синтаксис

```
FileClose(FileHandle)
```

FileEncrypt

Выполняет алгоритм шифрования файла. Разрешает рассылку CRBasic файлов без предоставления исходного кода.

Синтаксис

```
Boolean Variable = FileEncrypt(FileName)
```

FileList

Возвращает список файлов, имеющихся на определенном носителе.

Синтаксис

```
FileList (Drive, DestinationArray)
```

FileManage

Управляет программными файлами из активной программы регистратора данных.

Синтаксис

```
FileManage("Device: FileName", Attribute)
```

FileOpen

Открывает текстовый файл в кодировке ASCII или двоичный файл для записи или чтения.

Синтаксис

```
FileHandle = FileOpen("FileName", "Mode", SeekPoint)
```

FileRead

Считывает файл, указанный командой FileHandle, и сохраняет результаты в переменную или массив переменных.

Синтаксис

```
FileRead(FileHandle, Destination, Length)
```

FileReadLine

Считывает строку файла, указанную командой ***FileHandle***, и сохраняет результат в переменную или массив переменных.

Синтаксис

```
FileReadLine(FileHandle, Destination, Length)
```

FileRename

Изменяет имя файла на накопителе CR1000.

Синтаксис

```
FileRename(drive:OldFileName, drive:NewFileName)
```

FileSize

Возвращает размер файла, хранящегося в памяти CR1000.

Синтаксис

```
FileSize(FileHandle)
```

FileTime

Возвращает время создания файла, указанного командой ***FileHandle*.**

Синтаксис

```
Variable = FileTime(FileHandle)
```

FileWrite

Записывает данные в кодировке ASCII или двоичные данные в файл, на который внутри программы ссылается команда ***FileHandle*.**

Синтаксис

```
FileWrite(FileHandle, Source, Length)
```

Include

Вставляет код из файла (***Filename***) в месте команды **Include()** при компилировании. Команда **Include()** не может быть вложенной.

Синтаксис

```
Include("Device: Filename")
```

NewFile

Определяет, был ли обновлен файл, хранящийся в CR1000, с момента последнего запуска команды. Обычно используется для графических файлов.

Синтаксис

```
NewFile (NewFileVar, "FileName")
```

RunProgram

Вызывает вторичный программный файл CRBasic из текущей активной программы.

Синтаксис

```
RunProgram("Device:FileName", Attrib)
```

A.16 Доступ и управление таблицами данных

Команды для доступа и управления данными, сохраняемыми в таблицах, включая таблицы **Public (общего доступа)** и Status (состояния).

FileMark

Вставляет маркер файла в таблицу данных.

Синтаксис

```
FileMark(TableName)
```

GetRecord

Извлекает одну запись из таблицы данных и сохраняет результаты в массив. Может использоваться вместе с функцией **SecsSince1990().**

Синтаксис

```
GetRecord(Dest, TableName, RecsBack)
```

ResetTable

Используется для возврата в исходное состояние при контроле программы.

Синтаксис

```
ResetTable(TableName)
```

SetSetting

Изменяет значение для настройки или табличное поле **Status** (состояние).

Синтаксис

```
SetSetting("FieldName", Value)
```

SetStatus

Изменяет значение для настройки или табличное поле **Status** (состояние).

Синтаксис

```
SetStatus("FieldName", Value)
```

TableName.EventCount

Возвращает количество событий, хранящихся в памяти, которые были зафиксированы, для таблицы данных, управляемой событиями.

Синтаксис

```
TableName.EventCount(1,1)
```

TableName.FieldName

Обращается к определенному полю из записи в таблице.

Синтаксис

```
TableName. FieldName(FieldNameIndex, RecordsBack)
```

TableName.Output

Определяет, были ли записаны данные в указанную таблицу данных при последнем обращении к таблице данных.

Синтаксис

```
TableName.Output(1,1)
```

TableName.Record

Устанавливает номер определенной записи таблицы данных.

Синтаксис

```
TableName.Record(1,n)
```

TableName.TableFull

Указывает на то, что таблица с ограниченным заполнением заполнена, или на то, что начата перезапись старых данных в таблице, заполняемой непрерывно по кругу.

Синтаксис

```
TableName.TableFull (1,1)
```

TableName.TableSize

Возвращает количество записей, помещенных в таблицу данных.

Синтаксис

```
TableName.TableSize(1,1)
```

TableName.TimeStamp

Возвращает время в виде интервала или временную отметку для записи в указанной таблице данных.

Синтаксис

```
TableName.TimeStamp(m,n)
```

WorstCase

Сохраняет одно или несколько неблагоприятных событий, хранящихся в памяти, в отдельные таблицы. Используется вместе с функцией **DataEvent().**

Синтаксис

```
WorstCase(TableName, NumCases, MaxMin, Change, RankVar)
```

A.17 Протокол управления передачей / протокол Интернет (TCP/IP) — команды

Связанные темы:
• *TCP/IP — общие сведения (стр. 91)*
• *TCP/IP — подробная информация (стр. 423)*
• *TCP/IP — команды (стр. 593)*
• *Подключения TCP/IP — список(стр. 652)*

Эти команды предназначены для пользования электронной почтой, службой мобильных сообщений (CMC), веб-страницами и другими службами протокола Интернет. Эти службы доступны только тогда, когда CR1000 используется вместе с сетевым устройством связи, которое имеет разрешенный ключ протокола для подключения к Интернету с удаленным

доступом или протокола Интернет (PPP/IP), например, при использовании набора протоколов IP для CR1000.

DHCPRenew

Выполняет повторный запуск протокола динамического конфигурирования узла (DHCP) по интерфейсу локальной сети Ethernet.

Синтаксис

```
DHCPRenew
```

EMailRecv

Запрашивает сообщения электронной почты с SMTP-сервера (работающего на основе простого протокола передачи электронной почты) и сохраняет часть сообщений электронной почты в строковую переменную.

Синтаксис

```
variable = EMailRecv("ServerAddr", "ToAddr", "FromAddr",
   "Subject", Message, "Authen", "UserName", "Password",
   Result)
```

EMailSend

Отправляет сообщение электронной почты на один или несколько адресов электронной почты через SMTP-сервер.

Синтаксис

```
variable = EMailSend("ServerAddr", "ToAddr", "FromAddr",
   "Subject", "Message", "Attach", "UserName", "Password",
   Result)
```

EthernetPower

Контролирует состояние электропитания всех устройств локальной сети Ethernet.

Синтаксис

```
EthernetPower(state)
```

FTPClient

Отправляет или извлекает файл через службу FTP (протокол передачи файлов).

Синтаксис

```
Variable = FTPClient("IPAddress", "User", "Password",
   "LocalFileName", "RemoteFileName", PutGetOption)
```

HTTPGET

Отправляет запрос на сервер HTTP (протокол пересылки гипертекста), используя метод Get.

Синтаксис

```
HTTPGET( URI, Response, Header)
```

HTTPOut

Определяет строку кода HTML (язык гипертекстовой разметки), используемого в HTML-файле, создаваемом регистратором данных.

Синтаксис

```
WebPageBegin("WebPageName", WebPageCmd)
   HTTPOut("<p>html string to output " + variable + "
```

```
additional string to output</p>")
   HTTPOut("<p>html string to output " + variable + "
additional string to output</p>")
WebPageEnd
```

HTTPPOST

Отправляет файлы или строки текста в универсальный указатель ресурса в сети Интернет (URL).

Синтаксис

```
HTTPPOST( URI, Contents, Response, Header)
```

HTTPPUT

Отправляет запрос на сервер HTTP сохранить вложенный файл или данные под заданным унифицированным идентификатором ресурса (URI).

Синтаксис

```
HTTPPUT(URI, Contents, Response, Header, NumRecs, FileOption)
```

IPInfo

Возвращает IP-адрес указанного интерфейса регистратора данных в виде строки.

Синтаксис

```
Variable = IPInfo(Interface, Option)
```

IPNetPower

Контролирует состояние электропитания отдельных устройств локальной сети Ethernet.

Синтаксис

```
IPNetPower( IPInterface, State)
```

IPRoute

Настраивает интерфейс (Ethernet или PPP), планируемый к использованию, когда CR1000 отправляет исходящий пакет и оба интерфейса активны.

Синтаксис

```
IPRoute(IPAddr, IPInterface)
```

IPTrace

Записывает отладочные сообщения протокола Интернет (IP) в строковую переменную.

Синтаксис

```
IPTrace(Dest)
```

NetworkTimeProtocol

Выполняет синхронизацию часов регистратора данных с временем сервера Интернет.

Синтаксис

```
variable = NetworkTimeProtocol(NTPServer, NTPOffset,
   NTPMaxMSec)
```

PingIP

Отправляет IP-адрес.

Синтаксис

```
variable = PingIP(IPAddress, Timeout)
```

PPPOpen

Устанавливает соединение по протоколу PPP с сервером.

Синтаксис

```
variable = PPPOpen
```

PPPClose

Закрывает открытое соединение по протоколу PPP с сервером.

Синтаксис

```
variable = PPPClose
```

SNMPVariable

Задает иерархию пользовательской базы управляющей информации для простого протокола управления сетью (SNMP).

Синтаксис

```
SNMPVariable(Name, OID, Type, Access, Valid)
```

TCPClose

Закрывает сокет TCP/IP, который был настроен для коммуникационного взаимодействия.

Синтаксис

```
TCPClose(TCPSocket)
```

TCPOpen

Настраивает сокет TCP/IP для коммуникационного взаимодействия.

Синтаксис

```
TCPOpen(IPAddr, TCPPort, TCPBuffer)
```

UDPDataGram

Отправляет пакеты информации через коммуникационный пользовательский протокол данных (UDP).

Синтаксис

```
UDPDataGram(IPAddr, UDPPort, SendVariable, SendLength,
   RcvVariable, Timeout)
```

UDPOpen

Открывает порт для передачи пакетов UDP.

Синтаксис

```
UDPOpen(IPAddr, UDPPort, UDPBuffsize)
```

WebPageBegin / WebPageEnd

Задает веб-страницу, отображаемую, когда от внешнего источника приходит запрос на определенную страницу HTML.

Синтаксис

```
WebPageBegin("WebPageName", WebPageCmd)
   HTTPOut("<p>html string to output " + variable + "
additional string to output</p>")
   HTTPOut("<p>html string to output " + variable + "
additional string to output</p>")
WebPageEnd
```

XMLParse()

Считывает и ищет XML-файл в регистраторе данных.

Синтаксис

```
XMLParse(XMLContent, XMLValue, AttrName, AttrNameSpace,
   ElemName, ElemNameSpace, MaxDepth, MaxNameSpaces)
```

A.18 Управление модемом

Читать далее. Чтобы разобраться с дистанционной передачей данных, инициируемой регистратором данных, см. *Инициация связи (обратный вызов)* *(стр. 392)*.

DialModem

Отправляет строку набора модема по коммуникационному порту регистратора данных.

Синтаксис

```
DialModem(ComPort, BaudRate, DialString, ResponseString)
```

ModemCallback

Инициирует вызов компьютера через телефонный модем.

Синтаксис

```
ModemCallback(Result, COMPort, BaudRate, Security, DialString,
   ConnectString, Timeout, RetryInterval, AbortExp)
```

ModemHangup / EndModemHangup

Сохраняет код, который будет выполнен при отсоединении COM-порта.

Синтаксис

```
ModemHangup(ComPort)
[instructions to be run upon hang-up]
EndModemHangup
```

A.19 SCADA

Читать далее. См. разделы *Протокол DNP3* *(стр. 408)* и *Протокол Modbus* *(стр. 411)*.

Команды протоколов DNP3 и Modbus выполняются как задачи процесса.

DNP

Настраивает CR1000 как ведомое устройство DNP (удаленная станция или сервер). Третий параметр необязателен.

Синтаксис

```
DNP(ComPort, BaudRate, DisableLinkVerify)
```

DNPUpdate

Определяет, когда ведомое устройство DNP будет обновлять массив элементов DNP. Указывает адрес ведущего устройства DNP для отправки незапрашиваемых ответов.

Синтаксис

```
DNPUpdate(DNPAddr)
```

DNPVariable

Настраивает реализацию DNP в регистраторе данных компании Campbell Scientific, работающего как ведомое устройство DNP.

Синтаксис

```
DNPVariable(Array, Swath, Object, Variation, Class, Flag, Event
   Expression, Number of Events)
```

ModBusMaster

Настраивает регистратор данных как ведущее устройство ModBus для отправки или извлечения данных из ведомого устройства ModBus.

Синтаксис

```
ModBusMaster(ResultCode, ComPort, BaudRate, ModBusAddr,
   Function, Variable, Start, Length, Tries, TimeOut)
```

ModBusSlave

Настраивает CR1000 как ведомое устройство ModBus.

Синтаксис

```
ModBusSlave(ComPort, BaudRate, ModBusAddr, DataVariable,
   BooleanVariable)
```

A.20 Функции калибровки

Calibrate

Запускает калибровку аналоговых измерительных схем.

Синтаксис

```
Calibrate(Dest, Range) (parameters are optional)
```

FieldCal

Настраивает регистратор данных на выполнение калибровки по одной или нескольким переменным в массиве.

Синтаксис

```
FieldCal(Function, MeasureVar, Reps, MultVar, OffsetVar, Mode,
   KnownVar, Index, Avg)
```

FieldCalStrain

Настраивает регистратор данных на выполнение калибровки нуля и шунта для тензометрии.

Синтаксис

```
FieldCalStrain(Function, MeasureVar, Reps, GFAdj, ZeromV/V,
   Mode, KnownRS, Index, Avg, GFRaw, uStrainDest)
```

LoadFieldCal

Загружает значения из файла с расширением .cal в переменные, используемые в CR1000.

Синтаксис

```
LoadFieldCal(CheckSig)
```

NewFieldCal

Запускает сохранение значений калибровки после записи нового файла с расширением .cal.

Синтаксис

```
DataTable(TableName, NewFieldCal, Size)
   SampleFieldCal
EndTable
```

SampleFieldCal

Сохраняет значения в файл с расширением .cal в виде таблицы данных.

Синтаксис

```
DataTable(TableName, NewFieldCal, Size)
   SampleFieldCal
EndTable
```

A.21 Спутниковые системы

Команды для спутниковых систем ARGOS, GOES, OMNISAT и INMARSAT-C. Обратитесь к руководствам по спутниковым передатчикам, которые доступны на сайте *www.campbellsci.com/manuals (http://www.campbellsci.com/manuals).*

A.21.1 Спутниковая система Argos

ArgosData

Указывает данные, которые должны быть переданы на спутник системы ARGOS.

Синтаксис

```
ArgosData(ResultCode, ST20Buffer, DataTable, NumRecords,
   DataFormat)
```

ArgosDataRepeat

Устанавливает частоту повтора для команды **ArgosData()**.

Синтаксис

```
ArgosDataRepeat(ResultCode, RepeatRate, RepeatCount,
   BufferArray)
```

ArgosError

Отправляет команду **Получить и удалить сообщение об ошибке** передатчику спутниковой системы ARGOS.

Синтаксис

```
ArgosError(ResultCode, ErrorCodes)
```

ArgosSetup

Настраивает регистратор данных для передачи данных через спутник системы ARGOS.

Синтаксис

```
ArgosSetup(ResultCode, ST20Buffer, DecimalID, HexadecimalID,
   Frequency)
```

ArgosTransmit

Инициирует однократную передачу на спутник системы ARGOS при выполнении команды.

Синтаксис

```
ArgosTransmit(ResultCode, ST20Buffer)
```

A.21.2 Спутниковая система GOES

GOESData

Отправляет данные в спутниковый передатчик системы GOES компании Campbell Scientific.

Синтаксис

```
GOESData(Dest, Table, TableOption, BufferControl, DataFormat)
```

GOESGPS

Сохраняет данные глобальной системы позиционирования (GPS) со спутника системы GOES в два массива переменных.

Синтаксис

```
GOESGPS(GoesArray1(6), GoesArray2(7))
```

GOESSetup

Программирует передатчик спутниковой системы GOES для связи со спутником.

Синтаксис

```
GOESSetup(ResultCode, PlatformID, MsgWindow, STChannel, STBaud,
   RChannel, RBaud, STInterval, STOffset, RInterval)
```

GOESStatus

Запрашивает состояние и диагностическую информацию от спутникового передатчика системы GOES компании Campbell Scientific.

Синтаксис

```
GOESStatus(Dest, StatusCommand)
```

A.21.3 Спутниковая система OMNISAT

OmniSatData

Отправляет таблицу данных в передатчик спутниковой системы OMNISAT для передачи через спутник GOES или METEOSAT.

Синтаксис

```
OmniSatData(OmniDataResult, TableName, TableOption,
   OmniBufferCtrl, DataFormat)
```

OmniSatRandomSetup

Настраивает передатчик спутниковой системы OMNISAT на передачу данных через спутник GOES или METEOSAT с произвольной скоростью передачи.

Синтаксис

```
OmniSatRandomSetup(ResultCodeR, OmniPlatformID, OmniChannel,
   OmniBaud, RInterval, RCount)
```

OmniSatStatus

Запрашивает у передатчика спутниковой системы OMNISAT информацию о состоянии.

Синтаксис

```
OmniSatStatus(OmniStatusResult)
```

OmniSatSTSetup

Настраивает передатчик спутниковой системы OMNISAT на отправку данных через спутник GOES или METEOSAT с самосинхронизирующейся скоростью передачи.

Синтаксис

```
OmniSatSTSetup(ResultCodeST, ResultCodeTX, OmniPlatformID,
   OmniMsgWindow, OmniChannel, OmniBaud, STInterval, STOffset)
```

A.21.4 Спутниковая система INMARSAT-C

INSATData

Отправляет таблицу данных в передатчик спутниковой системы OMNISAT-I для передачи через спутник INSAT-1.

Синтаксис

```
INSATData(ResultCode, TableName, TX_Window, TX_Channel)
```

INSATSetup

Конфигурирует передатчик спутниковой системы OMNISAT-I для передачи данных через спутник INSAT-1.

Синтаксис

```
INSATSetup(ResultCode, PlatformID, RFPower)
```

INSATStatus

Запрашивает у передатчика информацию о состоянии.

Синтаксис

```
INSATStatus(ResultCode)
```

A.22 Пользовательские функции

Function / Return / Exit Function / EndFunction

Создают пользовательские команды CRBasic.

Синтаксис

```
Function [optional parameters] As [optional data type]
   Return [optional expression]
   ExitFunction [optional]
EndFunction
```

Optional (дополнительные)

Задает список дополнительных параметров, которые могут быть переданы в подпрограмму или функцию.

Синтаксис

```
Function (FunctionName) Param1, Param2, Optional Param3,
   Param4
```

Приложение B. Состояние, настройки и информация о таблицах данных (Status/Settings/DTI)

Связанные темы:
• *Состояние, настройки и информация о таблицах данных (Status/Settings/DTI)* *(стр. 603)*
• *Общее назначение таблицы состояния* *(стр. 604)*
• *Таблица состояния, как средство отладки* *(стр. 485)*22

Таблица **Status (Состояние)**, настройки CR1000 и информационная таблица **DataTableInfo (Информация о таблицах данных)**, вместе называемые **Status/Settings/DTI (состояние/настройки/таблицы)**, содержат регистры, настройки и информацию, относящуюся к установке, программированию и отладке многих расширенных систем CR1000.
Параметров состояния/настроек/таблиц достаточно много. Следует отметить:

- Все параметры состояния/настроек/таблиц, за исключением некоторых, доступны с помощью ключевых слов. Настоящее описание посвящено этим ключевым словам. Ключевые слова и описания приведены в алфавитном порядке в разделе *Описание параметров состояния/настроек/таблиц (в алфавитном порядке)* *(стр. 611)* *настоящего приложения.*
- Поля состояния только считываются (чаще всего). Некоторые из них могут сбрасываться.
- Параметры чтения/записи (чаще всего).
- Информация о таблицах данных только для чтения.
- Каталоги в приложении *Каталоги параметров состояния/настроек/таблиц* *(стр. 604)* содержат список нескольких групп ключевых слов. Каждому ключевому слову, указанному в этих группах, соответствует описание.
- Некоторые параметры состояния/настроек/таблиц имеют несколько имен в зависимости от интерфейса для доступа к ним.
- Ни один интерфейс не имеет доступа ко всем параметрам состояния/настроек/таблиц. Интерфейсы доступа включают:

Таблица 141. Параметры состояния/настроек/таблиц: узлы доступа

Узел доступ	*Расположен в...*
Редактор настроек	*Служебная программа конфигурации устройства, экран Connect (соединение) программного обеспечения LoggerNet, программное обеспечение PakBus Graph. См. Программная поддержка регистратора данных — подробная информация* *(стр. 450)**.*
Состояние	Просмотр информации в виде таблицы данных на *цифровом мониторе* *(стр. 521)**.*
DataTableInfo	Просмотр информации в виде таблицы данных на *цифровом мониторе* *(стр. 521)**.*
Состояние устройства	Пункт меню *ПО поддержки администратора регистратора данных* *(стр. 654)**.*
Изменение настроек	Пункт меню ПО *PakBusGraph.*
Настройки	Пункт меню экранной клавиатуры CR1000KD **Configure, Settings (Настройка, параметры)**
status.keyword/settings.keyword	Синтаксис программы CRBasic

[1]Информация, представленная в поле **Station Status** (Состояние устройства) автоматически не обновляется. Для того, чтобы обновить данные нажмите кнопку **Refresh** (Обновить).

Примечание. При создании таблиц **Status** (Состояние) и **DataTableInfo** (Информация о таблицах данных) используются мощности и пропускная способность процессора. Если мощности CR1000 загружены обработкой, например, при выполнении длинных или сложных операций, постоянный вызов таблицы **Status** (Состояние) может привести к *пропускам сканирования* *(стр. 487)**.*

B.1 Каталоги параметров состояния/настроек/таблиц

Ссылки в следующих таблицах помогут сориентироваться в системе параметров состояния/настроек/таблиц:

Таблица 142. Параметры состояния/настроек/таблиц: каталоги

Часто используемые (стр. 604)	Связь, стандартный программный интерфейс (CPI)	*Данные (стр. 609)*
Список ключевых слов в алфавитном порядке (стр. 605)	Связь, общая информация	*Память (стр. 609)*
Элементы таблицы состояния (стр. 606)	*Связь, протокол PakBus (стр. 608)*	*Прочее (стр. 609)*
Настройки (общие) (стр. 607)	*Связь, протокол TCP/IP I (стр. 608)*	*Устаревшее (стр. 609)*
Настройки (последовательного порта) (стр. 607)	*Связь, протокол TCP/IP II (стр. 608)*	*Операционная система и контроль версий аппаратного обеспечения (стр. 610)*
Настройки (TCP/IP) (стр. 607)	*Связь, протокол TCP/IP III (стр. 608)*	*Показатели электропитания (стр. 610)*
Настройки только в редакторе настроек (стр. 607)	Связь, беспроводной доступ (WiFi)	*Безопасность (стр. 610)*
Таблица с информацией о таблицах данных (стр. 607)	*Программа I на CRBasic (стр. 609)*	*Сигнатуры (стр. 610)*
Автоматическая калибровка (стр. 608)	*Программа II на CRBasic (стр. 609)*	

Таблица 143. Параметры состояния/настроек/таблиц: часто используемые

Действия	***Параметры состояния/настроек/таблиц***	***Таблица, где расположен параметр***
Найти адрес PakBus для CR1000	*PakBusAddress (Адрес PakBus) (стр. 623)*	**Связь, протокол PakBus**
Просмотреть сообщения о компилировании программы на CRBasic, выполняемой в CR1000	*CompileResults (Результаты компилирования) (стр. 614)*	**Программа I на CRBasic**
Ошибки программирования	*ProgErrors (Ошибки программирования) (стр. 626)* *ProgSignature (Сигнатура программы) (стр. 626)* *SkippedScan (Пропущенное сканирование) (стр. 628)* *StartUpCode (Код запуска) (стр. 629)*	**Программа II на CRBasic**
Таблицы данных	*DataFiltDays() (Количество дней заполнения таблицы данных) (стр. 615)* *SkippedRecord() (Пропущенная запись) (стр. 628)*	**Data (Данные)**
Память	*FullMemReset (Полный сброс памяти) (стр. 617)* *MemoryFree (Свободная память) (стр. 621)* *MemorySize (Объем памяти) (стр. 621)*	**Память**
Автоматические сбросы регистратора данных	*WatchdogErrors (Ошибки схемы безопасности) (стр. 632)*	**Прочее**
операционная система (ОС);	*OSDate (Дата выпуска операционной системы) (стр. 622)* *OSSignature (Сигнатура операционной системы) (стр. 622)* *OSVersion (Версия операционной системы) (стр. 622)*	**Операционная система и контроль версий аппаратного обеспечения**
Электропитание	*Battery (Батарея) (стр. 611)* *LithiumBattery (Литиевая батарея) (стр. 619)* *Low12VCount (Количество падений напряжения ниже 12 В) (стр. 620)*	**Показатели электропитания**

Таблица 144. Параметры состояния/настроек/таблиц: список ключевых слов в алфавитном порядке

B

Battery (Батарея) (стр. 611)

Baudrate() (Скорость передачи) (стр. 611)

Beacon() (Маяк)(стр. 612)

BuffDepth (Глубина буфера) (стр. 612)

C

CalDiffOffset) (Относительный сдвиг калибровки) (стр. 612)

CalGain() (Калибровочный коэффициент усиления) (стр. 613)

CalSeOffset() (Несимметричный сдвиг калибровки) (стр. 613)

CardBytesFree (Число свободных байтов на плате) (стр. 613)

CardStatus (Состояние платы) (стр. 613)

CentralRouters() (Центральные маршрутизаторы) (стр. 613)

CommActive() (Связь активна) (стр. 613)

CommConfig() (Конфигурация связи) (стр. 613)

CommsMemAlloc (Распределение ресурсов памяти для связи) (стр. 614)

CommsMemFree(1) (Объем свободной памяти для связи) (стр. 614)

CommsMemFree(2) (Объем свободной памяти для связи) (стр. 614)

CommsMemFree(3) (Объем свободной памяти для связи) (стр. 614)

CompileResults (Результаты компилирования) (стр. 614)

CPUDriveFree (Свободные байты на накопителе ЦПУ) (стр. 615)

CSIO1netEnable (Сеть CSIO1 активна)

CSIO2netEnable (Сеть CSIO2 активна)

D

DataFillDays() (Время заполнения таблицы данных в днях) (стр. 615)

DataRecordSize() (Количество записей в таблице данных) (стр. 615)

DataTableName() (Имя таблицы данных) (стр. 615)

DeleteCardFilesOnMismatch (Удалить файлы платы при несоответствии) (стр. 615)

DNS() (Служба доменных имен) (стр. 615)

E

ErrorCalib (Калибровка с ошибками)(стр. 616)

EthernetEnable (Ethernet активен) (стр. 616)

EthernetPower (питание Ethernet) (стр. 616)

FilesManager (Диспетчер файлов) (стр. 616)

FTPEnabled (FTP активен) (стр. 616)

FTPPassword (Пароль FTP) (стр. 616)

FTPPort (Порт FTP) (стр. 617)

FTPUserName (Имя пользователя FTP) (стр. 617)

FullMemReset (Полный сброс памяти) (стр. 617)

H

HTTP Enabled (HTTP активен)(стр. 617)

HTTPPort (Порт HTTP) (стр. 617)

I

IncludeFile (Включенный файл) (стр. 617)

IPAddressCSIO() (IP-адрес порта CSIO) (стр. 618)

IPAddressEth (IP-адрес для Ethernet) (стр. 618)

IPGateway (IP-шлюз) (стр. 618)

IPGatewayCSIO() (IP-шлюз для порта CSIO) (стр. 618)

IPInfo (Информация об IP-соединении) (стр. 618)

IPMaskCSIO() (IP-маска для порта CSIO) (стр. 618)

IPMaskEth (IP-маска сети Ethernet) (стр. 618)

IPTrace (IP-трассировка) (стр. 618)

IPTraceCode (Код IP-трассировки) (стр. 619)

IPTraceComport (Последовательный порт IP-трассировки) (стр. 619)

IsRouter (Маршрутизатор информационной системы) (стр. 619)

L

LastSlowScan() (Последнее медленное сканирование) (стр. 619)

LastSystemScan (Последнее сканирование системы) (стр. 619)

LithiumBattery (Литиевая батарея) (стр. 619)

Low12VCount (Количество падений напряжения ниже 12 В) (стр. 620)

Low5VCount (Количество падений напряжения ниже 5 В) (стр. 620)

M

MaxBuffDepth (Максимальная глубина буфера) (стр. 620)

MaxPacketSize (Максимальный размер пакета) (стр. 620)

MaxProcTime (Максимальное время обработки) (стр. 620)

MaxSlowProcTime() (Максимальное время медленной обработки) (стр. 620)

MaxSystemProcTime (Максимальное время системной обработки) (стр. 621)

MeasureOps (Коды операций для измерений) (стр. 621)

MeasureTime (Время измерений) (стр. 621)

MemoryFree (Свободная память) (стр. 621)

MemorySize (Объем памяти) (стр. 621)

Messages (Сообщения) (стр. 621)

N

Neighbors() (Соседние узлы) (стр. 622)

O

OSDate (Дата выпуска операционной системы) (стр. 622)

OSSignature (Сигнатура операционной системы) (стр. 622)

OSVersion (Версия операционной системы) (стр. 622)

P

PakBusAddress (Адрес PakBus) (стр. 623)

PakBusEncryptionKey (Ключ шифрования PakBus) (стр. 623)

PakBusPort (Порт PakBus)(стр. 623)

PakBusRoutes (Маршруты PakBus) (стр. 623)

PakBusTCPClients (Клиенты протокола TCP сети PakBus) (стр. 623)

PakBusTCPEnabled (Протокол TCP сети PakBus активен) (стр. 624)

PakBusTCPPassword (Пароль TCP для сети PakBus)(стр. 624)

PanelTemp (Температура панели) (стр. 624)

PingEnabled (Переброска пакетов активна) (стр. 624)

PortConfig() (Конфигурация порта) (стр. 624)

PortStatus() (Состояние порта) (стр. 624)

pppDial (Вызов PPP) (стр. 624)

pppDialResponse (Ответ на вызов PPP) (стр. 625)

pppInterface (Интерфейс PPP) (стр. 625)

pppIPAddr (IP-адрес протокола PPP)(стр. 625)

pppIPMask (IP-маска протокола PPP) (стр. 625)

pppPassword (Пароль PPP) (стр. 625)

pppUsername (Имя пользователя PPP) (стр. 625)

ProcessTime (Время обработки) (стр. 625)

ProgErrors (Ошибки программирования) (стр. 626)

ProgName (Имя программы) (стр. 626)

ProgSignature (Сигнатура программы) (стр. 626)

R

RecNum (стр. 626)

RevBoard (Модифицированная плата) (стр. 626)

RouteFilters (Фильтры маршрута) (стр. 626)

RS232Handshaking (Подтверждение связи RS232) (стр. 626)

RS232Power (Питание RS232) (стр. 627)

RS232Timeout (Тайм-аут RS232) (стр. 627)

RunSignature (Рабочая сигнатура) (стр. 627)

S

SecsPerRecord() (Интервал записи в секундах) (стр. 627)

Security(1) (Безопасность) (стр. 627)

Security(2) (Безопасность) (стр. 627)

Security(3) (Безопасность) (стр. 628)

SerialNumber (Серийный номер) (стр. 628)

ServicesEnabled() (Службы активны) (стр. 628)

SkippedRecord() (Пропущенная запись) (стр. 628)

SkippedScan (Пропущенное сканирование) (стр. 628)

SkippedSlowScan() (Пропущенное медленное сканирование) (стр. 628)

SkippedSystemScan (Пропущенное сканирование системы) (стр. 628)

SlowProcTime() (Время медленной обработки) (стр. 629)

StartTime (Время запуска) (стр. 629)

StartUpCode (Код запуска) (стр. 629)

StationName (Имя устройства) (стр. 629)

SW12Volts() (Переключения клемм 12 В) (стр. 629)

SystemProcTime (Время системной обработки) (стр. 629)

T

TCPClientConnections (Клиентские соединения TCP) (стр. 629)

TCPPort (Порт TCP) (стр. 629)

TelnetEnabled (протокол Telnet активен) (стр. 630)

TimeStamp (Временная метка) (стр. 630)

TLS Certificate (Сертификат протокола TLS) (стр. 630)

TLS Enabled (Протокол TLS активен) (стр. 630)

TLS Private Key (Секретный ключ протокола TLS) (стр. 630)

U

UDPBroadcastFilter (Фильтр для трансляции по протоколу UDP) (стр. 631)

USRDriveFree (Свободные байты на накопителе) (стр. 631)

USRDriveSize (Объем накопителя) (стр. 631)

UTCOffset (Сдвиг мирового координированного времени) (стр. 631)

V

VarOutOfBound (Выход переменной за пределы) (стр. 631)

Verify() (Верифицировать) (стр. 632)

W

WatchdogErrors (Ошибки схемы безопасности) (стр. 632)

Таблица 145. Параметры состояния/настроек/таблиц: элементы таблицы состояния на дисплее с клавиатурой CR1000KD

RecNum (стр. 626)	*VarOutOfBound (Выход переменной за пределы) (стр. 631)*	*PortConfig() (Конфигурация порта) (стр. 624)*
TimeStamp (Временная метка) (стр. 630)	*SkippedScan (Пропущенное сканирование) (стр. 628)*	*SW12Volts() (Переключения клемм 12 В) (стр. 629)*
OSVersion (Версия операционной системы) (стр. 622)	*SkippedSystemScan (Пропущенное сканирование системы) (стр. 628)*	*PakBusRoutes (Маршруты PakBus) (стр. 623)*
OSDate (Дата выпуска операционной системы) (стр. 622)	*SkippedSlowScan() (Пропущенное медленное сканирование) (стр. 628)*	*Messages (Сообщения) (стр. 621)*
OSSignature (Сигнатура операционной системы) (стр. 622)	*ErrorCalib (Калибровка с ошибками) (стр. 616)*	
	MemorySize (Объем памяти) (стр. 621)	
SerialNumber (Серийный номер) (стр. 628)	*MemoryFree (Свободная память) (стр. 621)*	
RevBoard (Модифицированная плата) (стр. 626)		
StationName (Имя устройства) (стр. 629)	*CommsMemFree(1) (Объем свободной памяти для связи) (стр. 614)*	
ProgName (Имя программы) (стр. 626)	*CommsMemFree(2) (Объем свободной памяти для связи) (стр. 614)*	
	CommsMemFree(3) (Объем свободной памяти для связи) (стр. 614)	
StartTime (Время запуска) (стр. 629)	*FullMemReset (Полный сброс памяти) (стр. 617)*	*CalGain() (Калибровочный коэффициент усиления) (стр. 613)*
RunSignature (Рабочая сигнатура) (стр. 627)	*CardStatus (Состояние платы) (стр. 613)*	
ProgSignature (Сигнатура программы) (стр. 626)	*MeasureOps (Коды операций для измерений) (стр. 621)*	*CalSeOffset() (Несимметричный сдвиг калибровки) (стр. 613)*
WatchdogErrors (Ошибки схемы безопасности) (стр. 632)	*MeasureTime (Время измерений) (стр. 621)*	*CalDiffOffset() (Относительный сдвиг калибровки) (стр. 612)*
PanelTemp (Температура панели) (стр. 624)	*ProcessTime (Время обработки) (стр. 625)*	
Battery (Батарея) (стр. 611)	*MaxProcTime (Максимальное время обработки) (стр. 620)*	
LithiumBattery (Литиевая батарея) (стр. 619)	*BuffDepth (Глубина буфера) (стр. 612)*	
	MaxBuffDepth (Максимальная глубина буфера) (стр. 620)	
	LastSystemScan (Последнее сканирование системы) (стр. 619)	
	LastSlowScan() (Последнее медленное сканирование) (стр. 619)	
Low12VCount (Тайм-аут RS232) (стр. 620)	*SystemProcTime (Время системной обработки) (стр. 629)*	
Low5VCount (Тайм-аут RS232) (стр. 620)	*SlowProcTime() (Время медленной обработки) (стр. 629)*	
CompileResults (Результаты компилирования) (стр. 614)	*MaxSystemProcTime (Максимальное время системной обработки) (стр. 621)*	
StartUpCode (Код запуска) (стр. 629)	*MaxSlowProcTime() (Максимальное время медленной обработки) (стр. 620)*	
ProgErrors (Ошибки программирования) (стр. 626)	*PortStatus() (Состояние порта) (стр. 624)*	

Таблица 146. Параметры состояния/настроек/таблиц: настройки (общие) на дисплее с клавиатурой CR1000KD

StationName (Имя устройства) (стр. 629)	*FilesManager (Диспетчер файлов) (стр. 616)*	*RS232Handshaking (Подтверждение связи RS232) (стр. 626)*
Security(1) (Безопасность) (стр. 627)	*RouteFilters (Фильтры маршрута) (стр. 626)*	*RS232Timeout (Тайм-аут RS232) (стр. 627)*
Security(2) (Безопасность) (стр. 627)	*CentralRouters() (Центральные маршрутизаторы) (стр. 613)*	*DeleteCardFilesOnMismatch (Удалить файлы платы при несоответствии) (стр. 615)*
Security(3) (Безопасность) (стр. 628)	*IncludeFile (Включенный файл) (стр. 617)*	
PakBusAddress (Адрес PakBus) (стр. 623)	*UTCOffset (Сдвиг мирового координированного времени) (стр. 631)*	
IsRouter (Маршрутизатор информационной системы) (стр. 619)	*CPUDriveFree (Свободные байты на накопителе ЦПУ) (стр. 615)*	
CommsMemAlloc (Распределение ресурсов памяти для связи) (стр. 614)	*USRDriveFree (Свободные байты на накопителе) (стр. 631)*	
MaxPacketSize (Максимальный размер пакета) (стр. 620)	*CardBytesFree (Число свободных байтов на плате) (стр. 613)*	
PakBusEncryptionKey (Ключ шифрования PakBus) (стр. 623)	*USRDriveSize (Объем накопителя) (стр. 631)*	
PakBusTCPPassword (Пароль TCP для сети PakBus)(стр. 624)	*RS232Power (Тайм-аут RS232) (стр. 627)*	

Таблица 147. Параметры состояния/настроек/таблиц: настройки (последовательного порта) на дисплее с клавиатурой CR1000KD	
Baudrate() (Скорость передачи) (стр. 611)	*Neighbors() (Соседние узлы)* (стр. 622)
Beacon() (Маяк) (стр. 612)	*Verify() (Верифицировать)* (стр. 632)

Таблица 148. Параметры состояния/настроек/таблиц: настройки (TCP/IP) на дисплее с клавиатурой CR1000KD	
IPInfo (Информация об IP-соединении) (стр. 618)	*pppIPAddr (IP-адрес протокола PPP)* (стр. 625)
EthernetEnable (Ethernet активен) (стр. 616)	*pppIPMask (IP-маска протокола PPP)* (стр. 625)
CSIO1netEnable	*pppUsername (Имя пользователя PPP)* (стр. 625)
	pppPassword (Пароль PPP) (стр. 625)
	pppDial (Вызов PPP) (стр. 624)
CSIO2netEnable	*pppDialResponse (Ответ на вызов PPP)* (стр. 625)
	IPTraceComport (Последовательный порт IP-трассировки) (стр. 619)
EthernetPower (Питание Ethernet) (стр. 616)	*IPTraceCode (Код IP-трассировки)* (стр. 619)
IPAddressEth (IP-адрес для Ethernet) (стр. 618)	*DNS() (Служба доменных имен)* (стр. 615)
IPMaskEth (IP-маска сети Ethernet) (стр. 618)	*PakBusTCPClients (Клиенты протокола TCP сети PakBus)* (стр. 623)
IPGateway (IP-шлюз) (стр. 618)	*UDPBroadcastFilter (Фильтр для трансляции по протоколу UDP)* (стр. 631)
	HTTPEnabled (HTTP активен) (стр. 617)
	FTPEnabled (FTP активен) (стр. 616)
	TelnetEnabled (протокол Telnet активен) (стр. 630)
IPAddressCSIO() (IP-адрес порта CSIO) (стр. 618)	*PingEnabled (Переброска пакетов активна)* (стр. 624)
IPMaskCSIO() (IP-маска для порта CSIO) (стр. 618)	*PakBusTCPEnabled (Протокол TCP сети PakBus активен)* (стр. 624)
IPGatewayCSIO() (IP-шлюз для порта CSIO) (стр. 618)	
PakBusPort (Порт PakBus) (стр. 623)	
FTPPort (Порт FTP) (стр. 617)	
HTTPPort (Порт HTTP) (стр. 617)	
FTPUserName (Имя пользователя FTP) (стр. 617)	
FTPPassword (Пароль FTP) (стр. 616)	
pppInterface (Интерфейс PPP) (стр. 625)	

Таблица 149. Параметры состояния/настроек/таблиц: настройки только в редакторе настроек	
TLS Certificate (Сертификат протокола TLS) (стр. 630)	*TLS Private Key (Секретный ключ протокола TLS)* (стр. 630)

Таблица 150. Параметры состояния/настроек/таблиц: ключевые слова таблицы информации о таблицах данных (DTI)		
DataFillDays() (Время заполнения таблицы данных в днях) (стр. 615)	*DataTableName() (Имя таблицы данных)*	*SkippedRecord() (Пропущенная запись)* (стр. 628)
DataRecordSize() (Количество записей в таблице данных) (стр. 615)	*SecsPerRecord() (Интервал записи в секундах)* (стр. 627)	

Таблица 151. Параметры состояния/настроек/таблиц: автоматическая калибровка		
CalDiffOffset) (Относительный сдвиг калибровки) *(стр. 612)*		*SkippedSystemScan (Пропущенное сканирование системы)* *(стр. 628)*
CalGain() (Калибровочный коэффициент усиления) *(стр. 613)*	*ErrorCalib (Калибровка с ошибками)* *(стр. 616)*	*SystemProcTime (Время системной обработки)* *(стр. 629)*
	LastSystemScan (Последнее сканирование системы) *(стр. 619)*	
CalSeOffset() (Несимметричный сдвиг калибровки) *(стр. 613)*	*MaxSystemProcTime (Максимальное время системной обработки)* *(стр. 621)*	

Таблица 152. Параметры состояния/настроек/таблиц: связь, общая информация		
Baudrate() (Скорость передачи) *(стр. 611)*	*CommsMemFree(2) (Объем свободной памяти для связи)* *(стр. 614)*	*RS232Handshaking (Тайм-аут RS232)* *(стр. 626)*
CommsMemAlloc (Распределение ресурсов памяти для связи) *(стр. 614)*	*CommsMemFree(3) (Объем свободной памяти для связи)* *(стр. 614)*	*RS232Power (Тайм-аут RS232)* *(стр. 627)*
		RS232Timeout (Тайм-аут RS232) *(стр. 627)*
CommsMemFree(1) (Объем свободной памяти для связи) *(стр. 614)*		

Таблица 153. Параметры состояния/настроек/таблиц: связь, протокол PakBus		
Beacon() (Маяк) *(стр. 612)*	*PakBusAddress (Адрес PakBus)* *(стр. 623)*	*PakBusTCPEnabled (Протокол TCP сети PakBus активен)* *(стр. 624)*
CentralRouters() (Центральные маршрутизаторы) *(стр. 613)*	*PakBusEncryptionKey (Ключ шифрования PakBus)* *(стр. 623)*	*PakBusTCPPassword (Пароль TCP для сети PakBus)* *(стр. 624)*
IsRouter (Маршрутизатор информационной системы) *(стр. 619)*	*PakBusPort (Порт PakBus)* *(стр. 623)*	*RouteFilters (Фильтры маршрута)* *(стр. 626)*
MaxPacketSize (Максимальный размер пакета) *(стр. 620)*	*PakBusRoutes (Маршруты PakBus)* *(стр. 623)*	*Verify() (Верифицировать)* *(стр. 632)*
Neighbors() (Соседние узлы) *(стр. 622)*	*PakBusTCPClients (Клиенты протокола TCP сети PakBus)* *(стр. 623)*	

Таблица 154. Параметры состояния/настроек/таблиц: связь, протокол TCP_IP I		
CSIO1netEnable	*IPGateway (IP-шлюз)* *(стр. 618)*	
CSIO2netEnable	*IPGatewayCSIO() (IP-шлюз для порта CSIO)* *(стр. 618)*	*IPTrace (IP-трассировка)* *(стр. 618)*
DNS() (Служба доменных имен) *(стр. 615)*		*IPTraceCode (Код IP-трассировки)* *(стр. 619)*
EthernetEnable (Ethernet активен) *(стр. 616)*	*IPInfo (Информация об IP-соединении)* *(стр. 618)*	*IPTraceComport (Последовательный порт IP-трассировки)* *(стр. 619)*
EthernetPower (Питание Ethernet) *(стр. 616)*	*IPMaskCSIO() (IP-маска для порта CSIO)* *(стр. 618)*	*PingEnabled (Переброска пакетов активна)* *(стр. 624)*
IPAddressCSIO() (IP-адрес порта CSIO) *(стр. 618)*	*IPMaskEth (IP-маска сети Ethernet)* *(стр. 618)*	*TelnetEnabled (протокол Telnet активен)* *(стр. 630)*
IPAddressEth (IP-адрес для Ethernet) *(стр. 618)*		

Таблица 155. Параметры состояния/настроек/таблиц: связь, протокол TCP_IP II		
FTPEnabled (FTP активен) *(стр. 616)*		*TLS Private Key (Секретный ключ протокола TLS)* *(стр. 630)*
FTPPassword (Пароль FTP) *(стр. 616)*		
FTPPort (Порт FTP) *(стр. 617)*	*TLS Certificate (Сертификат протокола TLS)* *(стр. 630)*	
FTPUserName (Имя пользователя FTP) *(стр. 617)*		
HTTPEnabled (HTTP активен) *(стр. 617)*		*UDPBroadcastFilter (Фильтр для трансляции по протоколу UDP)* *(стр. 631)*
HTTPPort (Порт HTTP) *(стр. 617)*		

Таблица 156. Параметры состояния/настроек/таблиц: связь, протокол TCP_IP III		
pppDial (Вызов PPP) (стр. 624)	*pppIPAddr (IP-адрес протокола PPP)* (стр. 625)	*pppUsername (Имя пользователя PPP)* (стр. 625)
pppDialResponse (Ответ на вызов PPP) (стр. 625)	*pppIPMask (IP-маска протокола PPP)* (стр. 625)	
pppInterface (Интерфейс PPP) (стр. 625)	*pppPassword (Пароль PPP)* (стр. 625)	

Таблица 157. Параметры состояния/настроек/таблиц: программа I на CRBasic		
BuffDepth (Глубина буфера) (стр. 612)	*MaxBuffDepth (Максимальная глубина буфера)* (стр. 620)	*MeasureTime (Время измерений)* (стр. 621)
CompileResults (Результаты компилирования) (стр. 614)	*MaxProcTime (Максимальное время обработки)* (стр. 620)	*Messages (Сообщения)* (стр. 621)
IncludeFile (Включенный файл) (стр. 617)	*MaxSlowProcTime() (Максимальное время медленной обработки)* (стр. 620)	
LastSlowScan() (Последнее медленное сканирование) (стр. 619)	*MeasureOps (Коды операций для измерений)* (стр. 621)	

Таблица 158. Параметры состояния/настроек/таблиц: программа II на CRBasic		
ProcessTime (Время обработки) (стр. 625)	*SkippedScan (Пропущенное сканирование)* (стр. 628)	*StartTime (Время запуска)* (стр. 629)
ProgErrors (Ошибки программирования) (стр. 626)	*SkippedSlowScan() (Пропущенное медленное сканирование)* (стр. 628)	*StartUpCode (Код запуска)* (стр. 629)
ProgName (Имя программы) (стр. 626)	*SlowProcTime() (Время медленной обработки)* (стр. 629)	*VarOutOfBound (Выход переменной за пределы)* (стр. 631)

Таблица 159. Параметры состояния/настроек/таблиц: Data (Данные)		
DataFillDays() (Время заполнения таблицы данных в днях) (стр. 615)	*DataTableName() (Имя таблицы данных)* (стр. 615)	*SkippedRecord() (Пропущенная запись)* (стр. 628)
DataRecordSize() (Количество записей в таблице данных) (стр. 615)	*SecsPerRecord() (Интервал записи в секундах)* (стр. 627)	

Таблица 160. Параметры состояния/настроек/таблиц: память		
CardBytesFree (Число свободных байтов на плате) (стр. 613)	*FilesManager (Диспетчер файлов)* (стр. 616)	*USRDriveFree (Свободные байты на накопителе)* (стр. 631)
CardStatus (Состояние платы) (стр. 613)	*FullMemReset (Полный сброс памяти)* (стр. 617)	*USRDriveSize (Объем накопителя)* (стр. 631)
CPUDriveFree (Свободные байты на накопителе ЦПУ) (стр. 615)	*MemoryFree (Свободная память)* (стр. 621)	
DeleteCardFilesOnMismatch (Удалить файлы платы при несоответствии) (стр. 615)	*MemorySize (Объем памяти)* (стр. 621)	

Таблица 161. Параметры состояния/настроек/таблиц: прочее		
	PortStatus() (Состояние порта) (стр. 624)	*TimeStamp (Временная метка)* (стр. 630)
	RecNum (стр. 626)	*UTCOffset (Сдвиг мирового координированного времени)* (стр. 631)
PanelTemp (Температура панели) (стр. 624)	*StationName (Имя устройства)* (стр. 629)	*WatchdogErrors (Ошибки схемы безопасности)* (стр. 632)
PortConfig() (Конфигурация порта) (стр. 624)	*SWÎ2Volts() (Переключения клемм 12 В)* (стр. 629)	

Таблица 162. Параметры состояния/настроек/таблиц: устаревшее		
IPTrace (IP-трассировка) (стр. 618)	*TCPClientConnections (Клиентские соединения TCP)* (стр. 629)	*TLS Enabled (Протокол TLS активен)* (стр. 630)
PakBusNodes (Узлы PakBus) (стр. 623)	*TCPPort (Порт TCP)* (стр. 629)	
ServicesEnabled() (Службы активны) (стр. 628)		

Таблица 163. Параметры состояния/настроек/таблиц: операционная система и контроль версий аппаратного обеспечения		
OSDate (Дата выпуска операционной системы) *(стр. 622)*	*OSVersion (Версия операционной системы)* *(стр. 622)*	*SerialNumber (Серийный номер)* *(стр. 628)*
OSSignature (Сигнатура операционной системы) *(стр. 622)*	*RevBoard (Модифицированная плата)* *(стр. 626)*	

Таблица 164. Параметры состояния/настроек/таблиц: показатели электропитания		
Battery (Батарея) *(стр. 611)*		*Low5VCount (Тайм-аут RS232)* *(стр. 620)*
	LithiumBattery (Литиевая батарея) *(стр. 619)*	
	Low12VCount (Количество падений напряжения ниже 12 В) *(стр. 620)*	

Таблица 165. Параметры состояния/настроек/таблиц: безопасность	
PakBusTCPPassword (Пароль TCP для сети PakBus) *(стр. 624)*	*Security(3) (Безопасность)* *(стр. 628)*
Security(1) (Безопасность) *(стр. 627)*	*TLS Certificate (Сертификат протокола TLS)* *(стр. 630)*
Security(2) (Безопасность) *(стр. 627)*	*TLS Private Key (Секретный ключ протокола TLS)* *(стр. 630)*

Таблица 166. Параметры состояния/настроек/таблиц: сигнатуры		
OSSignature (Сигнатура операционной системы) *(стр. 622)*	*ProgSignature (Сигнатура программы)* *(стр. 626)*	*RunSignature (Рабочая сигнатура)* *(стр. 627)*

B.2 Описание параметров состояния/настроек/таблиц (в алфавитном порядке)

Таблица 167. Параметры состояния/настроек/таблиц: B

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
Battery	Состояние устройства: **Напряжение батареи** Клавиатура: **Таблица состояния** ≈ строка 1 CRBasic: **variable = status.keyword** Напряжение батареи, подключенной к клеммам **POWER IN (ВВОД ПИТАНИЯ) 12 В** и **G (ЗЕМЛЯ)**. Измерение выполняется во время автоматической (фоновой) калибровки. Это то же измерение, которое выполняется с помощью команды **Battery() (Батарея)** на CRBasic.	Только чтение IEEE4 вольты	Напряжение основного источника питания	От 9,6 до 16
Baudrate() (Скорость передачи)	Редактор настроек: **Скорость передачи в бодах** Псевдоним: **Скорости передачи данных для коммуникационных портов** Клавиатура: **Настройки (последовательные порты)** ≈ строка 16 CRBasic: **variable = status.keyword; SetSetting(), SerialOut()** Массив целых чисел, задающих скорости передачи данных для последовательных портов **RS-232, CS I/O** и зажима **C**. **Таблица 168. Baudrate() Массив, ключевые слова и настройки по умолчанию** *Номер элемента массива* / *Ключевое слово для порта* / *Скорость передачи в бодах по умолчанию* **(1)** **ComRS232** **-115200** **(2)** **ComME** **-115200** **(3)** **ComSDC7** **115200** **(4)** **ComSDC8** **115200** **(5)** **ComSDC10** **115200** **(6)** **ComSDC11** **115200** **(7)** **Com1** **0** **(8)** **Com2** **0** **(9)** **Com3** **0** **(10)** **Com4** **0** **ComRS232** и **ComME** (CS I/O) поддерживают автоматическое определение скорости передачи. Значение скорости передачи, определяемой автоматически, принимается равным **0** или допустимой записи, перед которой стоит тире (например, **-115200**), при этом в качестве начальной скорости устанавливается введенное значение. Функция автоматического выбора скорости передачи выбирает скорость передачи поступающих данных и настраивает порт на данную скорость. Если скорость передачи данных по любому из портов с клеммами **C** изменяется, программа на CRBasic будет выполнять перекомпиляцию, если изменение соответствует переходу на **0** (отсоединен) или переходу от **0** на определенную скорость передачи (активен). См. таблицу с подробным описанием команд Baudrate() (Скорость передачи), Beacon() (Маяк) и Verify() (Верифицировать).	Чтение/запись массив типа LONG (длинный)	См. таблицу слева	**0** (автоматическое определение скорости передачи или отключено) **1200** **2400** **4800** **9600** **19200** **38400** **57600** **76800** **115200**

Beacon() (Маяк)	Редактор настроек: **Сигнальный интервал** Псевдоним: **Сигнальные интервалы последовательных портов** Клавиатура: **Настройки (последовательных портов)** CRBasic: **variable = settings.keyword;** **SetSettings()** Управляет скоростью, с которой CR1000 транслирует сообщения PakBus по соответствующему порту, для обнаружения новых соседних узлов. Также регулирует интервал проверки, установленный по умолчанию, если значение команды **Verify()** *(стр. 632)* для соответствующего порта равно **0**. **Таблица 169. Массив Beacon() (Маяк), ключевые слова и настройки по умолчанию** ***Номер элемента массива*** / ***Ключевое слово для порта*** **(1) ComRS232** **(2) ComME** **(3) ComSDC7** **(4) ComSDC8** **(5) ComSDC10** **(6) ComSDC11** **(7) Com1** **(8) Com2** **(9) Com3** **(10) Com4** См. таблицу с подробным описанием Baudrate() (Скорость передачи), Beacon() (Маяк) и Verify() (Верифицировать).	Чтение/запись массив типа LONG (длинный) секунды	0	От 0 до 2147483648
BuffDepth (Глубина буфера)	Клавиатура: **Таблица состояния** ≈ строка 42 CRBasic: **variable = status.keyword** Выдает информацию о глубине буфера для обработки в *конвейерном режиме (стр. 152)*, который показывает, насколько задача обработки отстоит от задачи измерения в текущий момент времени.	Только чтение тип LONG (длинный)	0	

Таблица 170. Параметры состояния/настроек/таблиц: C

Ключевое слово	***Псевдоним, доступ, описание***	***Чтение/ запись, тип данных, единицы измерения***	***Значение по умолчанию***	***Нормальный диапазон***
CalDifTOffset() (Относительный сдвиг калибровки)[2]	Клавиатура: **Таблица состояния** ≈ строка 63 CRBasic: **variable = status.keyword** Массив из 18 целых чисел, содержащий информацию об относительных сдвигах для каждого процесса интегрирования или объединения диапазонов. При необходимости обновляется в процессе фоновой калибровки.	Только чтение тип NTLONG (длинный) мВ		Близко к 0

CalGain() (Калибровочный коэффициент усиления)[2]	Клавиатура: **Таблица состояния** ≈ строка 61 CRBasic: **variable = status.keyword** Массив из 18 значений с плавающей точкой, содержащий коэффициенты усиления для каждого процесса интегрирования или объединения диапазонов. Обновляется в процессе фоновой калибровки.	Только чтение IEEE4 мВ		
CalSeOffSet() (Несимметричный сдвиг калибровки)[2]	Клавиатура: **Таблица состояния** ≈ строка 62 CRBasic: **variable = status.keyword** Массив из 18 целых чисел, содержащий информацию о несимметричных сдвигах для каждого процесса интегрирования или объединения диапазонов. При необходимости обновляется в процессе автоматической (фоновой) калибровки.	Только чтение тип LONG (длинный) мВ		Близко к 0
CardBytesFree	Клавиатура: **Таблица состояния** ≈ строка 18 CRBasic: **variable = settings.keyword; SetSettings()** Число свободных байтов на съемной плате памяти.	Только чтение тип LONG (длинный)	**-1** (без платы)	**>0** **-1** (без платы)
CardStatus	Состояние устройства: **Состояние платы** Клавиатура: **Таблица состояния** ≈ строка 37 CRBasic: **variable = status.keyword** Содержит строку с самой последней информацией о состоянии съемной платы памяти. Сообщения являются самоопределяющимися, например, **Card OK (с платой все в порядке), No Card Present (плата отсутствует), Card Not Being Used (плата не используется)**	Только чтение тип STRING (строка)		
CentralRouters() (Центральные маршрутизаторы)	Редактор настроек: **CentralRouters() (Центральные маршрутизаторы)** Клавиатура: **Настройки (общие)** ≈ строка 13 CRBasic: variable = **settings.keyword; SetSettings()** Массив из восьми адресов PakBus для маршрутизаторов, которые могут работать как центральные маршрутизаторы. Посредством задания непустого списка для этой настройки CR1000 конфигурируется как ответвленный маршрутизатор, это означает, что теперь не потребуется отслеживать соседние узлы, за исключением узлов, расположенных на его ответвлении. Сконфигурированная таким образом, система CR1000 игнорирует любые списки соседних узлов, полученные с адресов из настроек центральных маршрутизаторов, и пересылает полученные сообщения к ближайшему маршрутизатору, заданному по умолчанию, если в ее таблице маршрутизации отсутствует адрес назначения для этих сообщений. Каждый элемент должен быть отформатирован с использованием запятой, разделяющей отдельные значения.	Чтение/запись тип LONG (длинный)	0	
CommActive() (Связь активна)[3]	CRBasic: **variable = status.keyword** «Скрытый» массив указывает, активна ли связь в текущий момент времени по соответствующим портам. Порядок элементов массива следующий: ComRS232, ComME, ComSDC7, ComSDC8, ComSDC10, ComSDC11, Com1, Com2, Com3, Com4.	Только чтение тип BOOLEAN (булев)	**Ложь** (за исключением активных последовательных портов)	**Истина** или **ложь**
CommConfig() (Конфигурация связи)	CRBasic: **variable = status.keyword** «Скрытый» массив указывает на конфигурацию последовательных портов. При использовании **SerialOpen()** значения указывают на параметры форматирования для этой команды. Коммуникационное взаимодействие по протоколу PakBus может происходить одновременно по тому же порту, если порт был предварительно открыт (в случае CP UARTS) для PakBus или если порт всегда открыт (**CS I/O** и **RS-232**) для PakBus, код равен **4**. Порядок элементов массива следующий: RS-232, CS I/O ME, C0M310, CS I/O SDC7, CS I/O SDC8, CS I/O SDC10, CS I/O SDC11, Com1, Com2, Com3, Com4	Чтение/запись тип LONG (длинный)	**RS-232** всегда аппаратно активен **RS-232** через SDC8 = **4** (активен) **COM1, COM2, COM3** или **COM4** = **0** (отключен)	**0** = программа отключена **4** = программа активна

CommsMemAlloc (Распределение ресурсов памяти для связи)	Редактор настроек: **Распределение ресурсов для связи** Дополнительные названия: **узлы PakBus, распределение узлов PakBus, номер узла сети PakBus** Клавиатура: **Настройки (общие)** ≈ строка 7 CRBasic: **variable = settings.keyword; SetSettings()** Задает объем памяти, который система CR1000 выделяет для поддержания маршрутной информации PakBus. Приблизительно предоставляет максимальное количество узлов PakBus, которые может отслеживать система CR1000 в своих таблицах маршрутизации (см. раздел *CommsMemFree(2) (Объем свободной памяти для связи)* *стр. 492*).	Чтение/запись тип LONG (длинный)	При запуске без связи по протоколу TCP/IP: **1530** Указывает на IP-пакеты: • 15 большая очередь • 30 маленькая очередь • очередь на прием пуста	
CommsMemFree(1) (Объем свободной памяти для связи)	Клавиатура: **Таблица состояния** ≈ строка 33 CRBasic: **variable = status.keyword** Массив указывает количество буферов, используемых при коммуникационном взаимодействии, за исключением случая использования внешнего дисплея с клавиатурой. По два разряда на каждую категорию размера буфера. Самый младший разряд указывает количество самых малых буферов. Самый старший разряд указывает количество самых больших буферов. Когда протокол защиты транспортного уровня (TLS) не активен, используются четыре категории: «очень малый», «малый», «средний» и «большой». Когда протокол TLS активен, используется дополнительная пятая категория — «огромный», и под каждую категорию выделяется большее количество буферов. См. раздел *CommsMemFree(l) (Объем свободной памяти для связи)* *(стр. 491)*.	Только чтение тип LONG (длинный)	?	**Протокол защиты транспортного уровня (TLS) не активен:** очень малый — 05 малый — 15 средний — 25 большой — 15 огромный — 0 **Протокол TLS не активен:** очень малый — 160 малый — 99 средний — 99 большой — 30 огромный — 02
CommsMemFree(2) (Объем свободной памяти для связи)	Клавиатура: **Таблица состояния** ≈ строка 34 CRBasic: **variable = status.keyword** Массив указывает количество буферов, оставшихся для маршрутизации PakBus и списков соседних узлов. Каждый маршрут или соседний узел требуют один буфер. См. раздел *CommsMemFree(2) (Объем свободной памяти для связи)* *(стр. 492)*.	Только чтение тип LONG (длинный)	?	?
CommsMemFree(3) (Объем свободной памяти для связи)	Клавиатура: **Таблица состояния** ≈ строка 35 CRBasic: **variable = status.keyword** Массив указывает три двузначных поля справа (самый младший разряд) налево (самый старший разряд): сколько доступно «малых» IP-пакетов, «больших» IP-пакетов и полученные IP-пакеты в очереди приема, которые еще не обработаны. См. раздел *CommsMemFree(3) (Объем свободной памяти для связи)* *(стр. 493)*.	Только чтение тип LONG (длинный)		При запуске без связи по протоколу TCP/IP: 1530 — 30 малых, 15 больших IP-пакетов доступно без очереди на прием.
CompileResults	Состояние **устройства: результаты для последней скомпилированной программы** Клавиатура: **Таблица состояния** ≈ строка 23 CRBasic: **variable = status.keyword** Содержит сообщения об ошибках, возникших при компилировании или во время исполнения.	Только чтение тип STRING (строка)		0

CPUDriveFree (Свободные байты на накопителе ЦПУ)	Клавиатура: **Настройки (общие)** ≈ строка 16 CRBasic: **variable = settings.keyword; SetSettings()** Байты, оставшиеся на центральном процессоре: на накопителе. Этот накопитель находится во флеш-памяти с последовательным интерфейсом и всегда присутствует. Программы на CRBasic обычно хранятся здесь.	Только чтение целое число		

[2] Порядок и описания элементов массива автоматической калибровки:

(1) интегрирование 250 мс для диапазона 5000 мВ	(7) интегрирование 60 Гц для диапазона 5000 мВ	(13) интегрирование 50 Гц для диапазона 5000 мВ
(2) интегрирование 250 мс для диапазона 2500 мВ	(8) интегрирование 60 Гц для диапазона 2500 мВ	(14) интегрирование 50 Гц для диапазона 2500 мВ
(3) интегрирование 250 мс для диапазона 250 мВ	(9) интегрирование 60 Гц для диапазона 250 мВ	(15) интегрирование 50 Гц для диапазона 250 мВ
(4) интегрирование 250 мс для диапазона 25 мВ	(10) интегрирование 60 Гц для диапазона 25 мВ	(16) интегрирование 50 Гц для диапазона 25 мВ
(5) интегрирование 250 мс для диапазона 7,5 мВ	(11) интегрирование 60 Гц для диапазона 7,5 мВ	(17) интегрирование 50 Гц для диапазона 7,5 мВ
(6) интегрирование 250 мс для диапазона 2,5 мВ	(12) интегрирование 60 Гц для диапазона 2,5 мВ	(18) интегрирование 50 Гц для диапазона 2,5 мВ

[3] В общем случае команда **CommActive (Связь активна)** принимает значение **ИСТИНА** при получении поступающих символов вне зависимости от протокола. Команда принимает значение **ЛОЖЬ** после окончания тайм-аута длительностью 40 секунд, во время которого не обрабатываются поступающие символы, или когда используется протокол PakBus, и протокол последовательной передачи пакетов данных по COM-порту устанавливает отключение от сети. Поэтому следует отметить, что для протоколов отличных от PakBus (ModBus, DNP3, общие протоколы), которые обслуживаются командой **SerialIO()**, команда **CommActive (Связь активна)** будет иметь значение **ИСТИНА** до тех пор, пока принимаются символы с частотой большей, чем каждые 40 секунд. Кроме того, протокол двухточечного соединения (PPP) будет активировать COM-порт с тайм-аутом длительностью 31 мин. При закрытии протокола PPP будет выполняться отмена тайм-аута, а команда **CommActive (Связь активна)** принимать значение **ЛОЖЬ**. Кроме того, если процесс вызова продолжается, команда **CommActive (Связь активна)** принимает значение **ИСТИНА**. Другим событием, которое может привести к активации команды **ComME**, является команда системы GOES. В заключение, название команды **CommActive (Связь активна)** может ввести в заблуждение. Например, если отсутствуют поступающие символы для активации тайм-аута длительностью 40 секунд, во время которого команда **CommActive (Связь активна)** принимает значение **ИСТИНА**, а имеются только выходные данные, тогда команда **CommActive (Связь активна)** не примет значения ИСТИНА. Для протоколов отличных от PakBus активное значение **ИСТИНА** задерживается на 40 секунд после обработки последних поступивших символов. Для протокола PPP COM-порт всегда имеет значение **ИСТИНА**, пока открыт протокол PPP.

Таблица 171. Параметры состояния/настроек/таблиц: D

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
DataFillDays() (Время заполнения таблицы данных в днях)	Клавиатура: **DataTableInfo** ≈ строка 5 CRBasic: **variable = status.keyword** Содержит информацию о времени, которое требуется для заполнения таблицы данных. Каждая таблица имеет свой элемент в двумерном массиве. Первое измерение относится к внутриплатной памяти. Второе измерение относится к карте памяти стандарта CF (Compact Flash).	Только чтение массив типа LONG (длинный) дни		
DataRecordSize() (Количество записей в таблице данных)	Клавиатура: **DataTableInfo** : ≈3 CRBasic: **variable = status.keyword** Предоставляет информацию о количестве записей в таблице данных.	Только чтение массив типа LONG (длинный)		
DataTableName()	Клавиатура: **DataTableInfo** ≈ строка 1 CRBasic: **variable = status.keyword** Предоставляет информацию о наименованиях таблиц данных. Элементы массива расположены в порядке объявления таблиц данных в программе на CRBasic.	Только чтение массив типа STRING (строка)		
DeleteCardFilesOnMismatch (Удалить файлы платы при несоответствии)	Клавиатура: **Настройки (общие)** ≈ строка 23 Редактор настроек: **Удалить файлы данных команды CardOut в случае несоответствия таблиц данных команды CardOut** CRBasic: **variable = settings.keyword; SetSettings()**	Чтение/запись тип BOOLEAN (булев)	**Ложь**	**Истина** или **ложь**
DNS() (Служба доменных имен)	Имя в редакторе настроек: **Серверы доменных имен** Клавиатура: **Настройки (TCP/IP)** ≈ строка 32 CRBasic: **variable = settings.keyword; SetSettings()**	Чтение/запись тип STRING (строка)	0.0.0.0	0-255,0-255,0-255,0-255

	Задает адреса для двух серверов доменных имен, которые могут быть использованы системой CR1000 для отнесения доменных имен к IP-адресам. Следует отметить, что если протокол динамического конфигурирования узла (DHCP) используется для установления информации, касающейся взаимодействия по IP-протоколу, адреса, полученные через протокол DHCP, добавляются в этот список.			

Таблица 172. Параметры состояния/настроек/таблиц: E

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
ErrorCalib (Калибровка с ошибками)	Клавиатура: **Таблица состояния** ≈ строка 30 CRBasic: **variable = status.keyword** Количество измеренных ошибочных значений для калибровки. Ошибочные значения отбрасываются. Автоматическая калибровка выполняется при скрытом сканировании в режиме медленной последовательности. См. раздел Автоматическая калибровка CR1000 — общие сведения *(стр. 92)*.	Только чтение тип LONG (длинный) единицы счета	0	0
EthernetEnable (Ethernet активен)	Клавиатура: **Настройки (TCP/IP)** ≈ строка 2 Редактор настроек: **Интерфейс сети Ethernet активен** CRBasic: **variable = settings.keyword; SetSettings()**	Чтение/запись тип BOOLEAN (булев)	**Истина**	**Истина** или **ложь**
EthernetPower	Клавиатура: **Настройки (TCP/IP)** ≈ строка 6 Редактор настроек: **Питание сети Ethernet** CRBasic: **variable = settings.keyword; SetSettings()**	Чтение/запись тип UINT2 минуты		От 0 до 4, 4 = всегда отключено

Таблица 173. Параметры состояния/настроек/таблиц: F

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
FilesManager (Диспетчер файлов)	Редактор настроек: **Диспетчер файлов** Клавиатура: **Настройки (общие)** ≈ строка 11 CRBasic: **variable = settings.keyword; SetSettings()** Задает количество файлов назначенного типа, которые сохраняются при получении из определенного узла. См. раздел *Диспетчер файлов (стр. 384)*.	Чтение/запись тип STRING (строка)	**(0, , 0)**	От 0 до 351 символов
FTPEnabled (FTP активен)	Клавиатура: **Настройки (TCP/IP)** ≈ строка 36 Редактор настроек: **Протокол передачи файлов (FTP) активирован** CRBasic: **variable = settings.keyword; SetSettings()** Устанавливает **1**, если должна быть активирована служба FTP. Эта служба заблокирована по умолчанию. Привязана к команде **ServicesEnabled(), которая вышла в настоящее время из употребления.**	Чтение/запись тип BOOLEAN (булев)	**Ложь**	**Истина** или **ложь**
FTPPassword (Пароль FTP)	Редактор настроек: **Пароль FTP** Клавиатура: **Настройки (TCP/IP)** ≈ строка 22 CRBasic: **variable = settings.keyword; SetSettings()** Задает пароль, используемый для входа на FTP-сервер.	Чтение/запись тип STRING (строка)		От 0 до 63 символов

FTPPort (Порт FTP)	Редактор настроек: **Служебный порт FTP** Клавиатура: **Настройки (TCP/IP)** ≈ строка 17 CRBasic: **variable = settings.keyword; SetSettings()** Конфигурирует порт TCP, через который работает служба FTP. В общем случае значение, заданное по умолчанию, является достаточным, пока не возникнет необходимость задать другое значение, чтобы настроить правила преобразования портов в брандмауэре с функцией трансляции сетевых адресов.	Чтение/запись тип UINT2	**21**	**От 0** до **65535**
FTPUserName (Имя пользователя FTP)	Клавиатура: **Настройки (TCP/IP)** ≈ строка 21 Редактор настроек: **Имя пользователя FTP** CRBasic: **variable = settings.keyword; SetSettings()** Задает имя пользователя, используемое для входа на FTP-сервер. Пустая строка или анонимный вход (по умолчанию) прекращают работу FTP-сервера.	Чтение/запись тип STRING (строка)	анонимный	От 0 до 63 символов
FullMemReset (Полный сброс памяти)	Клавиатура: **Таблица состояния** ≈ строка 36 CRBasic: **variable = settings.keyword; SetSettings()** Ввести **98765** для запуска полного сброса памяти. См. раздел *Полный сброс памяти* *(стр. 381)*.	Чтение/запись тип LONG (длинный)	**0**	Для сброса ввести **98765**

Таблица 174. Параметры состояния/настроек/таблиц: H

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
HTTPEnabled (HTTP активен)	Редактор настроек: **Протокол передачи гипертекста (HTTP) активирован** Образована от: **ServicesEnabled()** Клавиатура: **Настройки (TCP/IP)** ≈ строка 35 CRBasic: **variable = settings.keyword; SetSettings()** Заменяет старую команду **ServicesEnabled()** **(Службы активны).** Активирует (**истина**) или отключает (**ложь**) службу HTTP.	Чтение/запись тип BOOLEAN (булев)	**Истина**	**Истина** или **ложь**
HTTPPort (Порт HTTP)	Редактор настроек: **Служебный порт HTTP** Клавиатура: **Настройки (TCP/IP)** ≈ строка 18 CRBasic: **variable = settings.keyword; SetSettings()** Конфигурирует порт TCP, через который работает служба HTTP (интернет-сервер). В общем случае значение, заданное по умолчанию, является достаточным, пока не возникнет необходимость задать другое значение, чтобы настроить правила преобразования портов в брандмауэре с функцией трансляции сетевых адресов.	Чтение/запись тип LONG (длинный)	**80**	**От 0** до **65535**

Таблица 175. Параметры состояния/настроек/таблиц: I

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
IncludeFile (Включенный файл)	Клавиатура: **Настройки (общие)** ≈ строка 14 Дополнительные названия: **Имя включаемого файла** Редактор настроек: **Имя включаемого файла** CRBasic: **variable = settings.keyword; SetSettings()**	Чтение/запись тип STRING (строка)	Не вводится	От 0 до 63 символов

	Имя файла, который должен включаться в конце текущей программы на CRBasic или может выполняться в качестве программы по умолчанию. См. раздел *Включаемый файл (стр. 147)*. Задать {накопитель}:{имя файла}, где накопитель: = CPU:, USR:, USB: или *CRD: (стр. 653)*. Расширения файлов программы должны быть допустимы для программы на CRBasic (.dld, crl).			
IPAddressCSIO() (IP-адрес порта CSIO)	Редактор настроек: **IP-адрес порта CS I/O** Клавиатура: **Настройки (TCP/IP)** ≈ строка 13 CRBasic: **variable = settings.keyword; SetSettings()**	Чтение/запись массив типа STRING (строка)	0.0.0.0	0-255,0–255,0–255,0–255
IPAddressEth (IP-адрес для Ethernet)	Редактор настроек: **IP-адрес для Ethernet** Клавиатура: **Настройки (TCP/IP)** ≈ строка 7 CRBasic: **variable = settings.keyword; SetSettings()** Задает IP-адрес для интерфейса сети Ethernet. Если значение равно нулю, адрес, маска сети и шлюз конфигурируются автоматически, используя протокол DHCP. Доступна только при подключенном соединении Ethernet. Изменение приведет к перекомпилированию программы на CRBasic.	Чтение/запись тип 4B STRING (строка)	0.0.0.0	0–255,0–255.0–255,0–255
IPGateway (IP-шлюз)	Редактор настроек: **Шлюз сети Ethernet, заданный по умолчанию** Дополнительные названия: **Шлюз, заданный по умолчанию** Клавиатура: **Настройки (TCP/IP)** ≈ строка 9 CRBasic: **variable = settings.keyword; SetSettings()** Задает адрес IP-маршрутизатора, к которому система CR1000 будет пересылать все нелокальные IP-пакеты, для которых в системе нет маршрута. Изменение приведет к перекомпилированию программы на CRBasic.	Чтение/запись тип 4B STRING (строка)	0.0.0.0	0–255,0–255.0–255,0–255
IPGatewayCSIO() (IP-шлюз для порта CSIO)	Редактор настроек: **Шлюз порта CS I/O, заданный по умолчанию** Клавиатура: **Настройки (TCP/IP)** ≈ строка 15 CRBasic: **variable = settings.keyword; SetSettings()**	Чтение/запись тип 4B STRING (строка)	0.0.0.0	0–255,0–255.0–255,0–255
IPInfo	Редактор настроек: **Информация по протоколу TCP/IP** Клавиатура: **Настройки (TCP/IP)** ≈ строка 1 CRBasic: **variable = settings.keyword; SetSettings(); IPInfo()** Указывает текущие параметры для IP-соединения (интернет-соединения).	Только чтение тип STRING (строка)		Отсутствует
IPMaskCSIO() (IP-маска для порта CSIO)	Редактор настроек: **Маска подсети для порта CS I/O** Клавиатура: **Настройки (TCP/IP)** ≈ строка 14 CRBasic: **variable = settings.keyword; SetSettings()**	Чтение/запись массив типа STRING (строка)	255.255.255.0	0–255,0–255.0–255,0–255
IPMaskEth (IP-маска сети Ethernet)	Редактор настроек: **Маска подсети для Ethernet** Клавиатура: **Настройки (TCP/IP)** ≈ строка 8 CRBasic: **variable = settings.keyword; SetSettings()** Задает маску подсети для интерфейса Ethernet. Эта настройка доступна при подключенном соединении Ethernet. Изменение приведет к перекомпилированию программы на CRBasic.	Чтение/запись тип STRING (строка)	255.255.255.0	0–255,0–255.0–255,0–255
IPTrace	Устаревшая. Привязана к команде **IPTraceComport (Последовательный порт IP-трассировки).**			

IPTraceCode (Код IP-трассировки)	Редактор настроек: **Код IP-трассировки** Клавиатура: **Настройки (TCP/IP)** ≈ строка 31 CRBasic: **variable = settings.keyword; SetSettings()** Эта настройка контролирует, какой тип информации отправлен через порт, указанный с помощью команды **IPTracePort (Порт IP-трассировки)** и протокола эмуляции терминала (Telnet). Полезные значения: 0 Трассировка не активна 1 Только запуск и схема безопасности 2 Подробный протокол PPP 4 Печать общих информационных сообщений 16 Отображение сообщений об ошибках сетевого интерфейса 256 Трассировка транспортного протокола (UDP/TCP/RVD) 8192 Трассировка протокола FTP 65535 Общая трассировка	Чтение/запись тип UINT2	0	От 0 до 65535, см. описание слева.
IPTraceComport (Последовательный порт IP-трассировки)	Редактор настроек: **COM-порт для IP-трассировки** Образована от: **IPTrace (IP-трассировка)** Клавиатура: **Настройки (TCP/IP)** ≈ строка 30 CRBasic: **variable = settings.keyword; SetSettings()** Указывает порт (если таковой имеется), по которому отправляется информация о трассировке по протоколу TCP/IP. Тип информации контролируется командой **IPTraceCode (Код IP-трассировки)**.	Чтение/запись тип LONG (длинный)	0 (не активен)	От 0 до 65535
IsRouter	Редактор настроек: **Маршрутизатор информационной системы** Клавиатура: **Настройки (общие)** ≈ строка 6 CRBasic: **variable = settings.keyword; SetSettings()** Контролирует конфигурацию системы CR1000 как маршрутизатора или оконечного узла.	Чтение/запись тип BOOLEAN (булев)	**Ложь**	**Истина** = маршрутизатор **Ложь**= оконечный узел

Таблица 176. Параметры состояния/настроек/таблиц: L

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
LastSlowScan() (Последнее медленное сканирование)	Клавиатура: **Таблица состояния** ≈ строка 45 CRBasic: **variable = status.keyword** Предоставляет информацию о том, когда выполнялось последнее сканирование в режиме **SlowSequence (медленная последовательность)** в программе на CRBasic . См. описание команд *MaxSlowProcTime (Максимальное время медленной обработки)* *(стр. 620)*, *SkippedSlowScan (Пропущенное медленное сканирование)* *(стр. 628)*, *SlowProcTime (Время медленной обработки)* *(стр. 629)*.	Только чтение массив типа NSEC (наносекунды) дата/время	Отсутствует	Недавнее прошлое
LastSystemScan (Последнее сканирование системы)	Клавиатура: **Таблица состояния** ≈ строка 44 CRBasic: **variable = status.keyword** Предоставляет информацию о времени последней автоматической (фоновой) калибровки, которая выполняется при скрытом сканировании в режиме медленной последовательности. См. описание команд MaxSystemProcTime (Максимальное время системной обработки) *(стр. 621)*, SkippedSystemScan (Пропущенное сканирование системы) *(стр. 628)*, SystemProcTime (Время системной обработки) *(стр. 629)* и раздел Автоматическая калибровка CR1000 — общие сведения *(стр. 92)*.	Только чтение тип NSEC (наносекунды) дата/время	Отсутствует	В пределах последних нескольких минут
LithiumBattery (Литиевая батарея)	Состояние устройства: **Литиевая батарея** Клавиатура: **Таблица состояния** ≈ строка 17 CRBasic: **variable = status.keyword** Напряжение внутренней литиевой батареи. Обновляется в процессе автоматической (фоновой) калибровки. Заменить литиевую батарею, если < 2,7 В постоянного тока. См. Замена внутренней батареи *(стр. 473)*.	Только чтение тип FLOAT (с плавающей точкой) вольты	Отсутствует	От 2,7 до 3:6

Low12VCount (Количество падений напряжения ниже 12 В)	Состояние устройства: **Количество падений напряжения ниже 12 В.** Клавиатура: **Таблица состояния** ≈ строка 21 CRBasic: **variable = status.keyword** Возрастает на 1 каждый раз, когда напряжение первичного источника питания CR1000 падает ниже -9,0. Обновляется при каждом обновлении таблицы **Status (Состояние).** Приведенные ниже сведения применимы для регистраторов данных CR800 и CR1000. Изменение может быть сделано для C6 и CR3000. Триггеры схемы сравнения низкого напряжения 12 В постоянного тока при 9,0 В постоянного тока. Заданная минимальная величина входного напряжения 9,6 В постоянного тока не приведет к режиму пониженного напряжения для 12 В постоянного тока, но режим пониженного напряжения для 12 В постоянного тока остановит выполнение программы до того, как будут потеряны измерения.	Только сброс тип LONG (длинный) единицы счета	0	От 0 до 99 0 = сброс
Low5VCount (Количество падений напряжения ниже 5 В)	Состояние устройства: **Количество падений напряжения ниже 5 В.** Клавиатура: **Таблица состояния** ≈ строка 22 CRBasic: **variable = status.keyword** Количество случаев падения напряжения источника питания 5 В постоянного тока ниже функционального порогового значения.	Только сброс тип LONG (длинный) единицы счета	0	От 0 до 99 0 = сброс

Таблица 177. Параметры состояния/настроек/таблиц: M

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
MaxBuffDepth (Максимальная глубина буфера)	Клавиатура: **Таблица состояния** ≈ строка 43 CRBasic: **variable = status.keyword** Максимальное количество буферов, которые система CR1000 будет использовать для обработки измерений, выполненных с задержкой по времени.	Чтение/запись тип LONG (длинный) нет единиц измерения	0	?
MaxPacketSize (Максимальный размер пакета)	Редактор настроек: **Максимальный размер пакета** Клавиатура: **Настройки (общие)** ≈ строка 8 CRBasic: **variable = settings.keyword; SetSettings()** Максимальное количество байтов на пакет совокупности данных.	Чтение/запись тип LONG (длинный) байты	1000	?
MaxProcTime (Максимальное время обработки)	Клавиатура: **Таблица состояния** ≈ строка 41 CRBasic: **variable = status.keyword** Максимальное время, требуемое для выполнения обработки при текущем сканировании. Значение сбрасывается по завершении сканирования. Вычисляется оперативно. См. описание *MeasureTime (Время измерений) (стр. 621)*, *ProcessTime (Время обработки) (стр. 625)*, *SkippedScan (Пропущенное сканирование) (стр. 628)*, *MaxProcTime (Максимальное время обработки) (стр. 620)*.	чтение/запись тип LONG (длинный) мкс	отсутствует	**0** = сброс
MaxSlowProcTime() (Максимальное время медленной обработки)	Клавиатура: **Таблица состояния** ≈ строка 49 CRBasic: **variable = status.keyword** Максимальное время, требуемое для выполнения процесса сканирования в режиме **SlowSequence (медленная последовательность)** в программе на CRBasic. **0** пока выполняется сканирование. См. описание *LastSlowScan (Последнее медленное сканирование) (стр. 619)*, *SkippedSlowScan (Пропущенное медленное сканирование) (стр. 628)*, *SlowProcTime (Время медленной обработки) (стр. 629)*.	Чтение/запись массив типа LONG (длинный) мкс	**0** до сканирования	0 = сброс

MaxSystemProcTime (Максимальное время системной обработки)	Клавиатура: **Таблица состояния** ≈ строка 48 CRBasic: **variable = status.keyword** Максимальное время, требуемое для процесса калибровки, которая выполняется при скрытом сканировании в режиме медленной последовательности. Отображает **0**, пока выполняется автоматическая калибровка. См. описание LastSystemScan (Последнее сканирование системы) *(стр. 619)*, SkippedSystemScan (Пропущенное сканирование системы) *(стр. 628)*, SystemProcTime (Время системной обработки) *(стр. 629)* и раздел Автоматическая калибровка CR1000 — общие сведения *(стр. 92)*.	только чтение тип LONG (длинный) мкс	**0** до сканирования	0 = сброс
MeasureOps (Коды операций для измерений)	Клавиатура: **Таблица состояния** ≈ строка 38 CRBasic: **variable = status.keyword** Предоставляет информацию о количестве кодов операций для специализированного указателя последовательности для выполнения всех измерений. Вычисляется во время компиляции. Содержит коды операций для калибровки (время компиляции), автоматической (фоновой) калибровки (системы) и медленных последовательностей. Предполагается что все команды измерений выполняются при каждом сканировании.	только чтение тип LONG (длинный) без единиц измерения	отсутствует	отсутствует
MeasureTime (Время измерений)	Клавиатура: **Таблица состояния** ≈ строка 39 CRBasic: **variable = status.keyword** Предоставляет информацию о времени необходимом для выполнения всех измерений при текущем сканировании. Вычисляется во время компиляции. Включает время интегрирования и установления сигнала. В конвейерном режиме обработка выполняется параллельно с этим временем, поэтому сумма **MeasureTime (Времени измерений)** и **ProcessTime (Времени обработки)** не равна требуемому времени сканирования. Предполагается, что все команды измерений будут выполняться при каждом сканировании. См. описание **ProcessTime (Время обработки)** и **MaxProcTime (Максимальное время обработки)**.	только чтение тип LONG (длинный) мкс	отсутствует	отсутствует
MemoryFree (Свободная память)	Состояние устройства: **Свободная память** Клавиатура: **Таблица состояния** ≈ строка 32 CRBasic: **variable = status.keyword** Свободная память статического запоминающего устройства с произвольной выборкой (SRAM) на ЦП. Вся свободная память может быть недоступна для таблиц данных. Поскольку память распределена и свободна, могут быть созданы «дырки» свободной памяти, которые не используются для памяти окончательных данных.	только чтение тип LONG (длинный) байты	**3794224**	≥ **4096**
MemorySize (Объем памяти)	Состояние устройства: **Память** Клавиатура: **Таблица состояния** ≈ строка 31 CRBasic: **variable = status.keyword** Суммарная память SRAM в системе CR1000. См. таблицу *Выделение памяти в CR1000 (стр. 371)*.	только чтение тип LONG (длинный) байты	**4194304**	**4194304**
Messages (Сообщения)	Клавиатура: **Таблица состояния** ≈ строка 54 CRBasic: **variable = status.keyword** Содержит строку для сообщений, вводимых вручную.	чтение/запись тип STRING (строка)	отсутствует	отсутствует

Таблица 178. Параметры состояния/настроек/таблиц: N

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
Neighbors() (Соседние узлы)	Редактор настроек: **Разрешенные соседние узлы** Клавиатура: **Настройки (последовательных портов)** CRBasic: **variable = settings.keyword; SetSettings()** Массив целых чисел, указывающий соседние узлы PakBus для коммуникационных портов. Номер элемента массива — Ключевое слово для порта **(1) ComRS232** **(2) ComME** **(3) ComSDC7** **(4) ComSDC8** **(5) ComSDC10** **(6) ComSDC11** **(7) Com1** **(8) Com2** **(9) Com3** **(10) Com4** См. раздел *Neighbors (Соседние узлы)* *(стр. 406)*.	чтение/запись тип STRING (строка)	0, 0	отсутствует

Таблица 179. Параметры состояния/настроек/таблиц: O

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
OSDate (Дата выпуска операционной системы)	Состояние устройства: **Дата выпуска операционной системы** Клавиатура: **Таблица состояния** ≈ строка 4 CRBasic: **variable = status.keyword** Дата выпуска операционной системы в формате ггммчч.	только чтение тип STRING (строка)	отсутствует	отсутствует
OSSignature (Сигнатура операционной системы)	Состояние устройства: **Сигнатура операционной системы** Клавиатура: **Таблица состояния** ≈ строка 5 CRBasic: **variable = status.keyword** Сигнатура операционной системы.	только чтение тип LONG (длинный)	отсутствует	отсутствует
OSVersion (Версия операционной системы)	Состояние устройства: **Версия операционной системы** Редактор настроек: **Версия операционной системы** Клавиатура: **Таблица состояния** ≈ строка 3 CRBasic: **variable = status.keyword** Версия операционной системы в CR1000.	только чтение тип STRING (строка)	отсутствует	отсутствует

Таблица 180. Параметры состояния/настроек/таблиц: P				
Ключевое слово	***Псевдоним, доступ, описание***	***Чтение/ запись, тип данных, единицы измерения***	***Значение по умолчанию***	***Нормальный диапазон***
PakBusAddress	Редактор настроек: **Адрес PakBus** Клавиатура: **Настройки (общие)** ≈ строка 5 CRBasic: **variable = settings.keyword; SetSettings()** Адрес PakBus для данной системы CR1000. Присваивает уникальный адрес, если данная система CR1000 должны быть размещена в сети PakBus. Допустимы адреса с 1 до 4094, однако адреса ≥ 4000 обычно резервируются для программной поддержки регистратора данных *(стр. 95)*. Для многих устройств компании Campbell Scientific, включая регистраторы данных, по умолчанию задан адрес 1.	чтение/запись тип LONG (длинный)	1	от 1 до 3999
PakBusEncryptionKey (Ключ шифрования PakBus)	Редактор настроек: **Ключ шифрования PakBus** Клавиатура: **Настройки (общие)** ≈ строка 9 CRBasic: **variable = settings.keyword; SetSettings()**	чтение/запись тип STRING (строка)	нет	от 0 до 63 символов
PakBusNodes	Устаревшая. Заменено/привязано к **CommsMemAlloc (Распределение ресурсов памяти для связи)**			
PakBusPort (Порт PakBus)	Редактор настроек: **Служебный порт PakBus/TCP** Клавиатура: **Настройки (TCP/IP)** ≈ строка 16 CRBasic: **variable = settings.keyword; SetSettings()** Заменяет старую настройку **TCPPort (Порт TCP)**. Действует, только если служба PPP активируется с использованием совместимого с протоколом PPP *сетевого соединения (стр. 652)*.Задает служебный порт TCP для коммуникационного взаимодействия по сети PakBus с CR1000. Пока существуют задачи для брандмауэра, вероятно, нет необходимости изменять значение, заданное по умолчанию для данной настройки.	только чтение тип LONG (длинный)	6785	от 0 до 65535
PakBusRoutes (Маршруты PakBus)	Редактор настроек: **Маршруты** Клавиатура: **Таблица состояния** ≈ строка 53 CRBasic: **variable = status.keyword** Перечисляет маршруты или соседние узлы для маршрутизатора, полученные CR1000 при прочтении настроек. Каждый маршрут представлен четырьмя элементами, разделенными запятыми и заключенными в круглые скобки: **(порт, адрес соседнего узла, адрес сети pakbus, время отклика)** См. раздел *PakBusRoutes (Маршруты PakBus) (стр. 405)*.	только чтение тип STRING (строка)	(1, 4089, 4089, 1000)	?
PakBusTCPClients (Клиенты протокола TCP сети PakBus)	Редактор настроек: **Клиенты PakBus/TCP** Псевдоним: **Клиентские соединения PakBus/TCP** Клавиатура: **Настройки (TCP/IP)** ≈ строка 33 CRBasic: **variable = settings.keyword; SetSettings()** До четырех адресов, задающих отходящие соединения PakBus/TCP, которые поддерживает регистратор данных. Формальный синтаксис настойки: TCP Connections := 4{address_pair}. address_pair := «(«address»,«tcpport»)». address := domain-name | ip-address. Пример для двух соединений: (192.168.4.203, 6785) (JOHN_DOE.server.com, 6785)	чтение/запись тип STRING (строка)	**(, 0)**	отсутствует

PakBusTCPEnabled (Протокол TCP сети PakBus активирован)	Редактор настроек: ??? Клавиатура: **Настройки (TCP/IP)** ≈ строка 39 Образована от: **ServicesEnabled()** CRBasic: **variable = settings.keyword; SetSettings()** Активирует (**Истина**) или отключает (**Ложь**) службу PakBus TCP.	чтение/запись тип BOOLEAN (булев)	Истина	**Истина** или **ложь**
PakBusTCPPassword (Пароль TCP для сети PakBus)	Редактор настроек: **Пароль PakBus/TCP** Клавиатура: **Настройки (общие)** ≈ строка 10 CRBasic: **variable = settings.keyword; SetSettings()** Когда пароль активен (имеет непустое значение), вход выполняется с использованием дайджеста сообщения (MD5) со случайными числами, и этот пароль будет успешно использоваться, пока коммуникационное взаимодействие по сети PakBus продолжается по IP-сокету.	чтение/запись тип STRING (строка)	нет (не активен)	от 0 до 63 символов
PanelTemp	Состояние устройства: **Температура панели** Клавиатура: **Таблица состояния** ≈ строка 15 CRBasic: **variable = status.keyword** Текущая температура монтажной панели Измерение выполняется во время фоновой калибровки.	только чтение тип FLOAT (с плавающей точкой) °C	отсутствует	от -40 до 85
PingEnabled (Переброска пакетов активна)	Редактор настроек: **Переброска пакетов активна** Клавиатура: **Настройки (TCP/IP)** ≈ строка 38 CRBasic: **variable = settings.keyword; SetSettings()** Активирует (**Истина**) или отключает (**Ложь**) службу переброски пакетов по межсетевому протоколу контрольных сообщений (ICMP). Заменяет старую команду **ServicesEnabled() (Службы активны).**	чтение/запись тип BOOLEAN (булев)	**Истина**	**Истина** или **ложь**
PortConfig() (Конфигурация порта)	Клавиатура: **Таблица состояния** ≈ строка 51 CRBasic: **variable = status.keyword** Конфигурация клемм **C**. Элементы массива в числовом порядке клемм **C**.	только чтение массив типа STRING (строка)	**Входной параметр**	**Input, Output, SDM, SDI-12, Tx, Rx**
PortStatus() (Состояние порта)	Клавиатура: **Таблица состояния** ≈ строка 50 CRBasic: **variable = status.keyword** Устанавливает клеммы **C**, настроенные на управление. Включено/высоко (**Истина**) или отключено/низко (**Ложь**). Элементы массива в числовом порядке клемм **C**. Обновляется каждые 500 мс.	чтение/запись массив типа BOOLEAN (булев)	**Ложь**	**Истина** или **ложь**
pppDial (Вызов PPP)	Редактор настроек: **Вызов PPP** Клавиатура: **Настройки (TCP/IP)** ≈ строка 28 CRBasic: **variable = settings.keyword; SetSettings()** Задает строку вызова, которая следует за абстрактным типом данных (ATD) (например, #777 для модема Redwing CDMA) или списком команд управления модемом, разделенных ';' (например, ATV1; AT+CGATT=0;ATD* 99***1#), используемыми для инициализации и вызова по модему перед попыткой соединения по протоколу PPP. Пустая строка означает, что вызов не требуется перед установлением соединения по протоколу PPP. Программа на CRBasic будет перекомпилирована, если значение будет изменено с NULL (НУЛЬ) на иное значение или наоборот: со значения, отличного от NULL (НУЛЬ), на NULL (НУЛЬ).	чтение/запись тип STRING (строка)	нет	отсутствует

pppDialResponse (Ответ на вызов PPP)	Редактор настроек: **Ответ на вызов PPP** Клавиатура: **Настройки (TCP/IP)** ≈ строка 29 CRBasic: **variable = settings.keyword; SetSettings()** Задает ответ, ожидаемый после вызова модема, перед тем, как будет установлено соединение по протоколу PPP. Программа на CRBasic будет перекомпилирована, если значение будет изменено с NULL (НУЛЬ) на иное значение или наоборот: со значения, отличного от NULL (НУЛЬ), на NULL (НУЛЬ).	чтение/запись тип STRING (строка)	CONNECT (СОЕДИНЕ-НИЕ)	отсутствует
pppInterface (Интерфейс PPP)	Редактор настроек: **Интерфейс PPP** Клавиатура: **Настройки (TCP/IP)** ≈ строка 23 CRBasic: **variable = settings.keyword; SetSettings()** Контролирует, для какого коммуникационного порта CR1000 настроена служба PPP. Предупреждение: если задано значение **CS I/O ME**, не присоединяйте другие устройства к порту **CS I/O**. Изменение приведет к перекомпилированию программы на CRBasic.	чтение/запись тип LONG (длинный)	0 (неактивен)	???
pppIPAddr (IP-адрес протокола PPP)	Клавиатура: **Настройки (TCP/IP)** ≈ строка 24 Редактор настроек: **IP-адрес протокола PPP** CRBasic: **variable = settings.keyword; SetSettings()** Задает IP-адрес, который используется для интерфейса PPP, если этот интерфейс активен (настройки **PPP Port / PPP Interface (Порт PPP / Интерфейс PPP)**, должны иметь значения, отличные от **Inactive (Неактивен)**). Синтаксис: **nnn.nnn.nnn.nnn**. Значение **0.0.0.0** или пустая строка указывают на то, что для принятия данного адреса, а также маски подсети должен использоваться протокол динамического конфигурирования узла (DHCP).	чтение/запись тип STRING (строка)	0.0.0.0	0–255,0–255,0–255,0–255
pppIPMask (IP-маска протокола PPP)	Редактор настроек: ??? Клавиатура: **Таблица состояния** ≈ строка 25 CRBasic: **variable = settings.keyword; SetSettings()**	чтение/запись тип STRING (строка)	нет	0–255,0–255,0–255,0–255
pppPassword (Пароль PPP)	Редактор настроек: **PPP табуляция: Пароль** Клавиатура: **Таблица состояния** ≈ строка 27 CRBasic: **variable = settings.keyword; SetSettings()** Задает пароль, который используется для входа на PPP-сервер, когда настройка интерфейса PPP установлена на один из клиентских вызовов. Также указывает пароль, который должен быть введен клиентом протокола PPP, когда настройка интерфейса PPP установлена на один из клиентских вызовов.	чтение/запись тип STRING (строка)	нет	от 0 до 63 символов
pppUsername (Имя пользователя PPP)	Редактор настроек: **Имя пользователя PPP** Клавиатура: **Настройки (TCP/IP)** ≈ строка 26 CRBasic: **variable = settings.keyword; SetSettings()** Задает имя пользователя, используемое для входа на PPP-сервер.	чтение/запись тип STRING (строка)	нет	от 0 до 63 символов
ProcessTime (Время обработки)	Клавиатура: **Таблица состояния** ≈ строка 40 CRBasic: **variable = status.keyword** Время обработки для последнего сканирования. Время измеряется от момента окончания команды **EndScan** (после фиксации события, связанного с измерениями) до момента начала команды **EndScan** (перед ожиданием начала события, связанного с измерениями) для следующего сканирования. Вычисляется оперативно. См. описание *MeasureTime (Время измерений)* *(стр. 621)*, *MaxProcTime (Максимальное время обработки)* *(стр. 620)*, *SkippedScan (Пропущенное сканирование)* *(стр. 628)*, *MaxProcTime (Максимальное время обработки)* *(стр. 620)*.	только чтение тип LONG (длинный) мкс	0	отсутствует

ProgErrors (Ошибки программирования)	Клавиатура: **Таблица состояния** ≈ строка 25 CRBasic: **variable = status.keyword** Количество ошибок при компиляции и во время выполнения для исполняемой программы.	только чтение тип LONG (длинный) единицы счета	0	≥ 0
ProgName	Состояние устройства: **Текущая программа** Клавиатура: **Таблица состояния** ≈ строка 10 CRBasic: variable = status.keyword Наименование текущей (выполняемой) программы	только чтение тип STRING (строка)	отсутствует	отсутствует
ProgSignature (Сигнатура программы)	Состояние устройства: **Сигнатура программы** Клавиатура: **Таблица состояния** ≈ строка 13 CRBasic: **variable = status.keyword** Сигнатура для текста файла выполняемой программы (включая комментарии). Не изменяется при изменениях в операционной системе. См. описание *RunSignature (Рабочая сигнатура)* *(стр. 627)*.Компилятор предварительного прохода CRBasic выдает эту сигнатуру с параметром **(или параметрами)** командной строки. В результате будут добавлена строка следующего вида: “Progname.CReX - Compiled in PipelineMode ProgSignature = XXXX.”	только чтение тип LONG (длинный)	отсутствует	отсутствует

Таблица 181. Параметры состояния/настроек/таблиц: R

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
RecNum (Номер записи)	Клавиатура: **Таблица состояния, таблица информации о таблицах данных (DTI)** ≈ заголовок слева CRBasic: **variable = status.keyword** Номер записи увеличивается по мере появления очередной записи данных в таблице состояния.	только чтение тип LONG (длинный)	отсутствует	от 0 до 2^{32}
RevBoard (Модификация платы)	Клавиатура: **Таблица состояния** ≈ строка 8 CRBasic: **variable = status.keyword** Модификация платы в виде **xxx.yyy**, где **xxx** = номер модификации аппаратного обеспечения; **yyy** = модификация программного обеспечения чипа синхронизации. Сохраняется во флеш-памяти.	только чтение тип STRING (строка)	отсутствует	отсутствует
RouteFilters (Фильтры маршрута)	Редактор настроек: **Фильтры маршрута** Клавиатура: **Настройки (общие)** ≈ строка 12 CRBasic: **variable = settings.keyword; SetSettings()** Ограничивает маршрутизацию или обработку некоторых типов сообщений сети PakBus . См. раздел *Фильтры маршрута* *(стр. 405)*	чтение/запись тип STRING (строка)	(0, 0, 0, 0)	от 0 до 351 символов
RS232Handshaking (Подтверждение связи RS232)	Редактор настроек: **Размер буфера аппаратного квитирования RS232** Клавиатура: **Настройки (общие)** ≈ строка 21 CRBasic: **variable = settings.keyword; SetSettings()** Если не равно 0, аппаратное квитирование для порта **RS-232** активно. Эта настройка задает максимальный размер пакета, отправляемого между проверками готовности к передаче (CTS).	чтение/запись тип LONG (длинный)	**0**	**от 0** до **65535**

RS232Power (Питание RS232)	Редактор настроек: **RS232 всегда включен** Клавиатура: **Настройки (общие)** ≈ строка 20 CRBasic: **variable = settings.keyword; SetSettings()** Контролирует, остается ли активным порт RS-232 даже в отсутствии связи. Если квитирование RS-232 активно (значение **RS232Handshaking** не равно 0), эта настройка должна иметь значение **Истина.**	чтение/запись тип BOOLEAN (булев)	**Ложь**	**Истина** или Ложь
RS232Timeout (Тайм-аут RS232)	Редактор настроек: **Тайм-аут аппаратного квитирования RS232** Клавиатура: **Настройки (общие)** ≈ строка 22 CRBasic: **variable = settings.keyword; SetSettings()** Тайм-аут аппаратного квитирования RS232. Задает время, в течение которого регистратор данных будет находится в режиме ожидания пакетов, если готовность к передаче не подтверждена.	чтение/запись тип IEEE4 десятки мс	**0**	???
RunSignature (Рабочая сигнатура)	Состояние устройства: **Рабочая сигнатура** Клавиатура: **Таблица состояния** ≈ строка 12 CRBasic: **variable = settings.keyword; SetSettings()** Сигнатура двоичной (скомпилированной) структуры выполняемой программы. Значение не зависит от комментариев или изменений, не касающихся функциональности. Зачастую изменяется при изменениях в операционной системе. См. описание *ProgSignature (Сигнатура программы)* *(стр. 626)*.Компилятор предварительного прохода может выдать текст программы, но формирование двоичной сигнатуры невозможно, поскольку порядок байтов, размер данных и структурный формат компилятора отличаются для персонального компьютера и CR1000.	только чтение тип LONG (длинный)	отсутствует	отсутствует

Таблица 182. Параметры состояния/настроек/таблиц: S

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
SecsPerRecord() (Интервал записи в секундах)	Дополнительные названия: **Интервал для таблиц данных** Клавиатура: **DataTableInfo** ≈ строка 4 CRBasic: **variable = status.keyword** Предоставляет интервал вывода данных для таблицы данных.	только чтение тип LONG (длинный) секунды	отсутствует	отсутствует
Security(1)	Редактор настроек: **Уровень безопасности 1** Дополнительные названия: **Код безопасности 1** Клавиатура: **Настройки (общие)** ≈ строка 4 CRBasic: **variable = settings.keyword; SetSettings()** Первый уровень в массиве из трех кодов безопасности. Не показывается, если безопасность не активна. **0** полностью отключает безопасность. См. *Безопасность — общие сведения* *(стр. 92)*.	чтение/запись тип LONG (длинный)	**0**	**от 0** до **65535** (**0** = отключено)
Security(2) (Безопасность)	Редактор настроек: **Уровень безопасности 2** Дополнительные названия: **Код безопасности 2** Клавиатура: **Настройки (общие)** ≈ строка 4 CRBasic: **variable = settings.keyword; SetSettings()** Второй уровень в массиве из трех кодов безопасности. Не показывается, если безопасность не активна. **0** отключает уровни 2 и 3. См. *Security(1)* *(стр. 627)* и раздел *Безопасность — общие сведения* *(стр. 92)*.	чтение/запись тип LONG (длинный)	**0**	**от 0** до **65535** (**0** = отключено)

Security(3) (Безопасность)	Редактор настроек: **Уровень безопасности 3** Дополнительные названия: **Код безопасности 3** Клавиатура: **Настройки (общие)** ≈ строка 4 CRBasic: **variable = settings.keyword; SetSettings()** Третий уровень в массиве из трех кодов безопасности. Не показывается, если безопасность не активна. **0** отключает уровень 3. См. *Security(1) (стр. 627)* и раздел *Безопасность — общие сведения (стр. 92)*.	чтение/запись тип LONG (длинный)	**0**	**от 0** до **65535** (**0** = отключено)
SerialNumber (Серийный номер)	Редактор настроек: **Серийный номер** Клавиатура: **Таблица состояния** ≈ строка 7 CRBasic: **variable = status.keyword** Серийный номер CR1000 присвоен на заводе-изготовителе. Сохраняется во флеш-памяти.	только чтение тип LONG (длинный)	отсутствует	отсутствует
ServicesEnabled() (Службы активны)	Устаревшая. Заменено и привязано к **HTTPEnabled (HTTP активирован), PakBusTCPEnabled (Протокол TCP сети Pakbus активирован), PingEnabled (Переброска пакетов активирована), TelnetEnabled (Telnet активирован), TLSEnabled (Протокол TLS активирован).**			
SkippedRecord() (Пропущенная запись)	Состояние устройства: **Пропущенные записи в XXXX** Клавиатура/дисплей: **DataTableInfo** ≈ строка 2 CRBasic: **variable = status.keyword** Предоставляет информацию о том, сколько записей пропущено в таблице данных. Элементы массива расположены в порядке объявления таблиц данных в программе на CRBasic.	только чтение массив типа LONG (длинный) единицы счета	**0**	**≥ 0** **0** = сброс
SkippedScan (Пропущенное сканирование)	Состояние устройства: **Пропущенное сканирование** Клавиатура: **Таблица состояния** ≈ строка 27 CRBasic: **variable = status.keyword** Количество *пропусков сканирования программы (стр. 487)* , которые возникают во время исполнения текущей программы. Не включает преднамеренный пропуск сканирования, который может возникнуть при использовании команд **ExitScan** и **Do / Loop**. Содержит количество ошибок кадра стандартного программного интерфейса (CPI). См. описание *MeasureTime (Время измерений) (стр. 621)*, *MaxProcTime (Максимальное время обработки) (стр. 620)*, *ProcessTime (Время обработки) (стр. 625)*, *MaxProcTime (Максимальное время обработки) (стр. 620)*.	чтение/запись тип LONG (длинный) единицы счета	**0**	**≥ 0** **0** = сброс
SkippedSlowScan() (Пропущенное медленное сканирование)	Состояние устройства: **Пропущенное медленное сканирование** Клавиатура: **Таблица состояния** ≈ строка 29 CRBasic: **variable = status.keyword** Целое число для каждого сканирования в режиме **SlowSequence (Медленная последовательность)** в программе на CRBasic. Количество пропусков сканирования. См. описание *LastSlowScan (Последнее медленное сканирование) (стр. 619)*, *MaxSlowProcTime (Максимальное время медленной обработки) (стр. 620)*, *SlowProcTime (Время медленной обработки) (стр. 629)*.	чтение/запись тип LONG (длинный) единицы счета	0	≥ 0 0 = сброс
SkippedSystemScan (Пропущенное сканирование системы)	Состояние устройства: **Пропущенное сканирование системы** Клавиатура: **Таблица состояния** ≈ строка 28 CRBasic: **variable = status.keyword** Количество пропусков сканирования при автоматической (фоновой) калибровке. Автоматическая калибровка выполняется при скрытом сканировании в режиме медленной последовательности. Ввести **0** для сброса. См. описание LastSystemScan (Последнее сканирование	чтение/запись тип LONG (длинный) единицы счета	**0**	**≥ 0** **0** = сброс

	системы) *(стр. 619)*, MaxSystemProcTime (Максимальное время системной обработки) *(стр. 621)*, SystemProcTime (Время системной обработки) *(стр. 629)* и раздел Автоматическая калибровка CR1000 — общие сведения *(стр. 92)*.			
SlowProcTime() (Время медленной обработки)	Клавиатура: **Таблица состояния** ≈ строка 47 CRBasic: **variable = status.keyword** Целое число для каждого сканирования в режиме **SlowSequence (Медленная последовательность)** в программе на CRBasic. Указывает время, которое требуется для выполнения сканирования. См. описание *LastSlowScan (Последнее медленное сканирование) (стр. 619)*, *MaxSlowProcTime (Максимальное время медленной обработки) (стр. 620)*, *SkippedSlowScan (Пропущенное медленное сканирование) (стр. 628)*.	тип LONG (длинный) мкс	большое число, пока выполняется **SlowSequence (Медленная последовательность)**.	
StartTime (Время запуска)	Состояние устройства: **Время запуска** Клавиатура: **Таблица состояния** ≈ строка 11 CRBasic: **variable = status.keyword** Время начала исполнения программы.	только чтение тип NSEC (наносекунды) дата и время		
StartUpCode (Код запуска)	Клавиатура: **Таблица состояния** ≈ строка 24 CRBasic: **variable = status.keyword** Указывает, зачем выполняемая программа была скомпилирована. **Истина** означает, что программа была скомпилирована в результате запуска регистратора из-за условий пониженного потребления энергии. **Ложь** означает, что компиляция была обусловлена командой **Program Send (Отправка программы)**, транзакцией **File Control (Управление файлом)** или сбросом схемы безопасности.	только чтение тип BOOLEAN (булев)	**Ложь**	**Истина** или **ложь**
StationName	Состояние устройства: **Имя устройство, о котором предоставляется уведомление** Редактор настроек: **Имя устройства** Клавиатура: **Таблица состояния** ≈ строка 9 Клавиатура: **Настройки (общие)** ≈ строка 1 CRBasic: **variable = settings.keyword; SetSettings(); StationName()** Сохраняет имя устройства во флеш-памяти. Имя устройства автоматически не соответствует тому, которое введено в ПО поддержки регистратора данных. См. описание имен устройств в руководствах к ПО поддержки регистратора данных. Имя устройства в ПО поддержки регистратора данных – это имя, появляющееся в заголовке файлов данных, извлекаемых с персонального компьютера. Данное имя устройства может быть выбрано в таблице данных, используя синтаксис доступа к таблицам данных *(стр. 167)*.	чтение/запись тип STRING (строка)		
SW12Volts() (Переключения клемм 12 В)	Клавиатура: **Таблица состояния** ≈ строка 52 CRBasic: **variable = status.keyword** Состояние переключения для клеммы 12 В постоянного тока.	чтение/запись тип BOOLEAN (булев)	**Ложь**	**Истина** или **ложь**
SystemProcTime (Время системной обработки)	Дополнительные названия: **Время выполнения фоновой калибровки** Клавиатура: **Таблица состояния** ≈ строка 46 CRBasic: **variable = status.keyword** Время, требуемое для выполнения автоматической (фоновой) калибровки. См. описание LastSystemScan (Последнее сканирование системы) *(стр. 619)*, MaxSystemProcTime (Максимальное время системной обработки) *(стр. 621)*, SystemProcTime (Время системной обработки) *(стр. 629)* и раздел Автоматическая калибровка CR1000 — общие сведения *(стр. 92)*.	только чтение тип FLOAT (с плавающей точкой) мкс	большое число, пока выполняется автоматическая калибровка	

Таблица 183. Параметры состояния/настроек/таблиц: T

Ключевое слово	*Псевдоним, доступ, описание*	*Чтение/ запись, тип данных, единицы измерения*	*Значение по умолчанию*	*Нормальный диапазон*
TCPClientConnections (Клиентские соединения TCP)	Устаревшая. Привязано к (или заменено на) *PakBusTCPClients (Клиенты протокола TCP сети PakBus) (стр. 623)*.			
TCPPort (Порт TCP)	Устаревшая. Привязано к (или заменено на) **PakBusPort (Порт PakBus)** *(стр. 623)*.			

TelnetEnabled (Протокол Telnet активен)	Редактор настроек: **Протокол Telnet активен** Образована от: **ServicesEnabled()** Клавиатура: **Настройки (TCP/IP)** ≈ строка 37 CRBasic: **variable = settings.keyword; SetSettings()** Активирует (**Истина**) или отключает (**Ложь**) службу Telnet.	чтение/запись тип BOOLEAN (булев)	**Истина**	**Истина** или **ложь**
TimeStamp	Клавиатура: Заголовок справа **Таблицы состояния** Клавиатура: Заголовок справа **Таблицы DTI** CRBasic: **variable = status.keyword** Время сканирования, в течение которого была сформирована запись. См. описание *Time Stamps (Метки времени)* *(стр. 303)*.	только чтение тип NSEC (наносекунды) время	отсутствует	отсутствует
TLS Certificate (Сертификат протокола TLS)	Редактор настроек: **TLS| Set Certificate (Задать сертификат)** Дополнительные названия: **Имя файла сертификата для протокола TLS** Задает имя файла для сертификата x509 в формате электронной почты с повышенной защитой (PEM).	STRING	отсутствует	отсутствует
TLS Enabled (Протокол TLS активен)	Дополнительные названия: **Безопасность на транспортном уровне (TLS) активна**, устаревшее. Заменено/привязано к **TLSPassword (Пароль протокола TLS)** *(стр. 630)*.			
TLS Private Key (Секретный ключ протокола TLS)	Редактор настроек: **TLS | Set Private Key (Установить секретный ключ)** Псевдоним: **TLS Private Key File Name (Имя файла с секретным ключом протокола TLS)** CRBasic: **variable = settings.keyword; SetSettings()** Задает имя файла для секретного ключа в формате RSA (шифрование по схеме открытого ключа).	STRING	отсутствует	отсутствует
TLSConnections (Соединения TLS)	Редактор настроек: **Максимальное количество соединений TLS-сервера** Клавиатура: **Настройки (TCP/IP)** ≈ строка 41 CRBasic: **variable = settings.keyword; SetSettings()** Используется, когда CR1000 является сервером, и задает максимальное количество параллельных соединений TLS-клиентов, которые могут быть подключены.	чтение/запись **тип LONG (длинный)**	0	0 to 255
TLSPassword (Пароль TLS)	Редактор настроек: **Пароль для секретного ключа протокола TLS** Клавиатура: **Настройки (TCP/IP)** ≈ строка 42 CRBasic: **variable = settings.keyword; SetSettings()** Задает пароль, который используется для дешифрования файла секретного ключа.	чтение/запись тип STRING (строка)	нет	отсутствует
TLSstatus (Состояние TLS)	Редактор настроек: **Состояние TLS** Дополнительные названия: **Состояние безопасности на транспортном уровне (TLS)** Клавиатура: **Настройки (TCP/IP)** ≈ строка 20 CRBasic: **variable = settings.keyword; SetSettings()**	только чтение тип STRING (строка)	нет	?

Таблица 184 Параметры состояния/настроек/таблиц: U				
Ключевое слово	***Псевдоним, доступ, описание***	***Чтение/ запись, тип данных, единицы измерения***	***Значение по умолчанию***	***Нормальный диапазон***
UDPBroadcastFilter (Фильтр для трансляции по протоколу UDP)	Редактор настроек: **IP-трансляция с фильтром** Клавиатура: **Настройки (TCP/IP)** ≈ строка 34	чтение/запись тип UINT2	**0**	**от 0** до **65535**
USRDriveFree (Свободные байты на накопителе)	Клавиатура: **Настройки (общие)** ≈ строка 17 CRBasic: **variable = settings.keyword; SetSettings()** Байты, оставшиеся на накопителе, считываемом USR. Накопитель, считываемый USR, создан пользователем и обычно используется для хранения файлов с расширением .jpg и др.	только чтение тип LONG (длинный) байты	**0**	?
USRDriveSize (Объем накопителя)	Редактор настроек: **USR: Размер накопителя** Клавиатура: **Таблица состояния** ≈ строка 19 CRBasic: **variable = settings.keyword; SetSettings()** Конфигурирует накопитель, считываемый USR. Если 0, накопитель удаляется. Если не 0, накопитель создается. Изменение приведет к перекомпилированию программы на CRBasic.	чтение/запись тип LONG (длинный) байты	**0**	**8192** минимально
UTCOffset (Сдвиг мирового координированного времени)	Редактор настроек: **Сдвиг мирового координированного времени** Клавиатура/дисплей: **Настройки (общие)** ≈ строка 15 CRBasic: **variable = settings.keyword; SetSettings()** Разность между местным временем (часы CR1000) и мировым координированным временем. Используется в электронной почте и заголовках HTML (эти протоколы требуют, чтобы метки времени были проставлены в соответствии с мировым координированным временем), а также командами **GPS(), NetworkTimeProtocol()** и **DaylightSavingTime()**.	чтение/запись тип LONG (длинный) секунды	-1 (отключено)	от -43200 до 43200 (-1=отключено)

Таблица 185. Параметры состояния/настроек/таблиц: V				
Ключевое слово	***Псевдоним, доступ, описание***	***Чтение/запись, тип данных, единицы измерения***	***Значение по умолчанию***	***Нормальный диапазон***
VarOutOfBound (Выход переменной за пределы)	Состояние устройства: **Выход переменной за пределы** Клавиатура/дисплей: **Таблица состояния** ≈ строка 26 CRBasic: **variable = status.keyword** Количество попыток записи в массив с выходом за пределы объявленного размера. Запись не производится. Указывает на ошибку в программе CRBasic. Если массив используется в цикле или в выражении, прекомпилятор или компилятор не выполняют проверки наличия обращений к недоступной части массива (т. е. обращение к массиву с индексом переменной, например arr(index) = arr(index-l), где индекс является переменной).	только чтение тип LONG (длинный) единицы счета	**0**	**≥ 0** **0** = сброс

Verify() **(Верифицировать)**	Редактор настроек: **Интервал верификации** Дополнительные названия: **Интервалы между проверками последовательных портов** Клавиатура: **Настройки (последовательных портов)** CRBasic: **variable = settings.keyword; SetSettings()** Массив из целых чисел, задающий интервал, который описывается как интервал верификации связи в приветственных сообщениях передачи данных по сети PakBus. Косвенно управляет частотой, с которой CR1000 делает попытку начать приветственную передачу с соседним узлом, если на данном интервале не происходит другого коммуникационного взаимодействия. Номер элемента массива / Ключевое слово для порта: **(1) ComRS232** **(2) ComME** **(3) ComSDC7** **(4) ComSDC8** **(5) ComSDC10** **(6) ComSDC11** **(7) Com1** **(8) Com2** **(9) Com3** **(10) Com4** См. таблицу с подробным описанием команд Baudrate() (Скорость передачи), Beacon() (Маяк) и Verify() (Верифицировать).	чтение/запись тип LONG (длинный) секунды	**0**	**от 0** до **2147483648**

Таблица 186. Параметры состояния/настроек/таблиц: W

Ключевое слово	***Псевдоним, доступ, описание***	***Чтение/ запись, тип данных, единицы измерения***	***Значение по умолчанию***	***Нормальный диапазон***
WatchdogErrors **(Ошибки схемы безопасности)**	Клавиатура: **Таблица состояния** ≈ строка 14 Состояние устройства: **Ошибки схемы безопасности** CRBasic: **variable = status.keyword** Количество ошибок схемы безопасности, возникающих во время выполнения данной программы. Сбрасывается автоматически при компилировании новой программы.	только чтение тип LONG (длинный) единицы счета	**0**	≥ **0** **0** = сброс

Приложение C. Схема расположения последовательных портов

C.1 Последовательный порт CS I/O

Конфигурация выводов для порта CS I/O системы CR1000 приведена в таблице *Название выводов CS I/O* *(стр. 633)*.

Таблица 187. Название выводов CS I/O

АББР: аббревиатура названия функции.

КОНТАКТ: Номер контакта.

ВЫХОД: Выход сигнала CR1000 к внешним устройствам.

ВХОД: Вход сигнала в CR1000 от внешних устройств.

КОНТАКТ	АББР	ВХОД/ ВЫХОД	***Описание***
1	5 Vdc	O	5 В: Источники 5 В постоянного тока, используемые для питания внешних устройств.
2	SG		Заземление сигнала Обеспечивает замыкание цепи питания для контакта 1 (5 В) и используется как опорный сигнал для различных уровней напряжения.
3	RING	I	Вызов: Вызывается внешним устройством для перевода CR1000 в телекоммуникационный режим.
4	RXD	I	Прием данных: Последовательные данные, передаваемые внешним устройством, принимаются через контакт 4.
5	ME	O	Модем активен: Активируется, когда CR1000 устанавливает, что модем вызывается по линии.
6	SDE	O	Синхронное устройство активно: Используется для адресации к синхронным устройствам (SD) и может использоваться как линия передачи сигнала для принтеров.
7	CLK/HS	ВХОД/ ВЫХОД	Часы/квитирование: Используется с линиями SDE и TXD для адресации и передачи данных к устройствам SD. Когда не используется как часы, контакт 7 может использоваться как линия квитирования (во время вывода на принтер, высокий уровень сигнала активирует, низкий уровень сигнала отключает).
8	+12 В пост. тока		
9	TXD	O	Передача данных: Последовательные данные передаются от CR1000 к внешним устройствам через контакт 9; логическая разметка для низкого уровня сигнала (0 В), логический интервал для высокого уровня сигнала (5 В), стандартная асинхронная работа с кодировкой ASCII, 8-битный, отсутствие контроля по четности, 1 стартовый бит, скорость передачи данных в бодах: 300, 1200, 2400, 4800, 9600, 19200, 38400, 115200 (на выбор пользователя).

C.2 Последовательный порт RS-232

C.2.1 Схема расположения выводов

Конфигурация выводов для 9-контактного порта **RS-232** системы CR1000 приведена в таблице *Расположение выводов RS-232 CR1000* *(стр. 634)*. Информация для использования нуль-модема **RS-232** представлена в таблице *Стандартный нульмодемный кабель или подключение контактов через адаптер* *(стр. 635)*.

Порт **RS-232** системы CR1000 функционирует как устройство телекоммуникационного оборудования (DCE) или как устройство оконечного оборудования обработки данных (DTE). Если **RS-232** функционирует как устройство телекоммуникационного оборудования, необходим нульмодемный кабель. Чаще всего **RS-232** используется как средство подключения к устройству телекоммуникационного оборудования компьютера. Стандартный

кабель с разъемами типа DB9-DB9 может использоваться для соединения устройства телекоммуникационного оборудования компьютера с устройством оконечного оборудования CR1000. В следующей таблице описаны функции контактов **RS-232** со стандартной системой обозначений наименований для телекоммуникационного оборудования.

Примечание. Контакты 1, 4. 6 и 9 функционируют иначе, чем у стандартного устройства телекоммуникационного оборудования. Это сделано для обеспечения подключения к модему или к другому устройству телекоммуникационного оборудования через нуль-модем.

Таблица 188. Схема расположения выводов RS-232 системы CR1000

PIN: номер контакта
O: выход сигнала CR1000 к устройству RS-232
I: вход сигнала в CR1000 от устройства RS-232
X: сигнал не имеет подключения (плавающий)

КОНТАКТ	*Функция телекоммуникационного оборудования*	*Функция регистратора*	*ВХОД / ВЫХОД*	*Описание*
1	DCD	DTR (соединен с контактом 6)	O[1]	Готовность оконечного оборудования
2	TXD	TXD	O	Асинхронная передача данных
3	RXD	RXD	I	Асинхронный прием данных
4	DTR	N/A	X[1]	Не подключен
5	GND	GND	GND	Земля
6	DSR	DTR	O[1]	Готовность оконечного оборудования
7	CTS	CTS	I	Готовность к передаче
8	RTS	RTS	O	Запрос на передачу
9	RI	RI	I[1]	Вызов

[1] Функция контакта отличается от функции контакта стандартного устройства телекоммуникационного оборудования. Эти контакты будут приспособлены для подключения к модему или к другим устройствам телекоммуникационного оборудования через нульмодемный кабель.

C.2.2 Режимы питания

Порт **RS-232** остается включенным при следующих условиях: 1) когда установлена настройка **RS232Power (Питание RS232)** или 2) когда в программе используется команда **SerialOpen()** для ***COMRS232***. В этих условиях **RS-232** работает без тайм-аута. Если команда **SerialClose()** применяется после **SerialOpen(),** порт отключается и остается в режиме ожидания до поступления символов.

В условиях нормальной эксплуатации порт отключается и ожидает входных данных. После получения входных данных начинается отсчет времени программного тайм-аута длительностью 40 секунд, по истечении которого произойдет отключение. Тайм-аут длительностью 40 секунд обычно имеет место при коммуникационном взаимодействии с *ПО поддержки регистратора данных (стр. 95)*, поскольку программа передает информацию как часть протокола, сообщающего CR1000, что можно отключать порт.

Аппаратное обеспечение сконфигурировано таким образом, чтобы в режиме ожидания обнаружить активность и инициировать включение. Режим ожидания имеет проблему, связанную с потерей первого символа входящего потока данных. PakBus учитывает это и использует «кольцевые пакеты», в начале которых располагаются дополнительные байты синхронизации. Команда **SerialOpen()** оставляет интерфейс включенным, так что входящие байты не теряются.

Если в регистраторе есть данные для отправки через порт **RS-232**, не являющиеся ответом на полученный пакет (например, отправка маркера),

устройство включит интерфейс, отправит данные и вернется в режим ожидания без тайм-аута длительностью 40 секунд.

Таблица 189. Стандартный нульмодемный кабель или подключение контактов через адаптер

Сокет DB9 #		***Сокет DB9 #***
		4
		3
		2
		1 & 6
		5
		8
		7
9	большинство нуль-модемов не имеют подключения[1]	9

[1] Если нуль-модем не подключен от одного 9-контактного разъема к другому 9-контактному разъему, модем необходимо сконфигурировать для вывода сигнала RING (ВЫЗОВ) (или ввести другие символы перед сигналом DTR) по передающей линии модема, чтобы запустить регистратор данных и активировать линию передачи сигнала DTR или модем.

Приложение D. Таблица кодов ASCII / ANSI

Подробнее:
• *Термин: ASCII/ANSI (стр. 507)*
• *Таблица ASCII/ANSI (стр. 637)*

Американский стандартный код для обмена информацией (ASCII) / Американский национальный институт стандартов (ANSI)

Таблица 190. Десятичные и шестнадцатеричные коды и символы, используемые приборами CR1000

Десятичный	Шестнадцатеричный	Дисплей с клавиатурой	LoggerNet	Гипертерминал	Десятичный	Шестнадцатеричный	Дисплей с клавиатурой	LoggerNet	Гипертерминал
0	0		НУЛЬ	НУЛЬ	128	80		€	Ç
1	1		□	☺	129	81		□	ü
2	2		□	☻	130	82		‚	é
3	3		□	♥	131	83		*f*	â
4	4		□	♦	132	84		„	ä
5	5		□	♣	133	85		…	à
6	6		□	♠	134	86		†	å
7	7		□	•	135	87		‡	ç
8	8		□	◘	136	88		^	ê
9	9			ht	137	89		‰	ë
10	a		lf	lf	138	8a		Š	è
11	b		□	vt	139	8b		‹	ï
12	c		□	ff	140	8c		Œ	î
13	d		cr	cr	141	8d		□	ì
14	e		□	♫	142	8e		Ž	Ä
15	f		□	☼	143	8f		□	Å
16	10		□	►	144	90		□	É
17	11		□	◄	145	91		'	æ
18	12		□	↕	146	92		'	Æ
19	13		□	!!	147	93		"	ô
20	14		□	¶	148	94		"	ö
21	15		□	§	149	95		•	ò
22	16		□	–	150	96		-	û
23	17		□	↨	151	97		-	ù
24	18		□	↑	152	98		~	ÿ
25	19		□	↓	153	99		™	Ö
26	1a		□	→	154	9a		š	Ü
27	1b		□		155	9b		›	¢

Таблица 190. Десятичные и шестнадцатеричные коды и символы, используемые приборами CR1000

Десятичный	Шестнадцатеричный	Дисплей с клавиатурой	LoggerNet	Гипертерминал	Десятичный	Шестнадцатеричный	Дисплей с клавиатурой	LoggerNet	Гипертерминал
28	1c		□	∟	156	9c		œ	£
29	1d		□	↔	157	9d		□	¥
30	1e		□	▲	158	9e		ž	Pt
31	1f		□	▼	159	9f		Ÿ	*f*
32	20	SP	SP	SP	160	a0			á
33	21	!	!	!	161	a1		¡	í
34	22	"	"	"	162	a2		¢	ó
35	23	#	#	#	163	a3		£	ú
36	24	$	$	$	164	a4		¤	ñ
37	25	%	%	%	165	a5		¥	Ñ
38	26	&	&	&	166	a6		¦	ª
39	27	'	'	'	167	a7		§	º
40	28	(	(	(	168	a8		¨	¿
41	29	)	)	)	169	a9		©	⌐
42	2a	*	*	*	170	aa		ª	¬
43	2b	+	+	+	171	ab		«	½
44	2c	,	,	,	172	ac		¬	¼
45	2d	-	-	-	173	ad			¡
46	2e	.	.	.	174	ae		®	«
47	2f	/	/	/	175	af		–	»
48	30	0	0	0	176	b0		°	░
49	31	1	1	1	177	b1		±	▒
50	32	2	2	2	178	b2		²	▓
51	33	3	3	3	179	b3		³	│
52	34	4	4	4	180	b4		´	┤
53	35	5	5	5	181	b5		µ	╡
54	36	6	6	6	182	b6		¶	╢
55	37	7	7	7	183	b7		·	╖
56	38	8	8	8	184	b8		¸	╕
57	39	9	9	9	185	b9		¹	╣
58	3a	:	:	:	186	ba		º	║
59	3b	;	;	;	187	bb		»	╗
60	3c	<	<	<	188	bc		¼	╝
61	3d	=	=	=	189	bd		½	╜
62	3e	>	>	>	190	be		¾	╛

Таблица 190. Десятичные и шестнадцатеричные коды и символы, используемые приборами CR1000

Десятичный	Шестнадцатеричный	Дисплей с клавиатурой	LoggerNet	Гипертерминал	Десятичный	Шестнадцатеричный	Дисплей с клавиатурой	LoggerNet	Гипертерминал
63	3f	?	?	?	191	bf		¿	┐
64	40	@	@	@	192	c0		À	└
65	41	A	A	A	193	c1		Â	┴
66	42	B	B	B	194	c2		Â	┬
67	43	C	C	C	195	c3		Ã	├
68	44	D	D	D	196	c4		Ä	—
69	45	E	E	E	197	c5		Å	┼
70	46	F	F	F	198	c6		Æ	╞
71	47	ЗЕМЛЯ	ЗЕМЛЯ	ЗЕМЛЯ	199	c7		Ç	╟
72	48	H	H	H	200	c8		È	╚
73	49	I	I	I	201	c9		É	╔
74	4a	J	J	J	202	ca		Ê	╩
75	4b	K	K	K	203	cb		Ë	╦
76	4c	L	L	L	204	cc		Ì	╠
77	4d	M	M	M	205	cd		Í	=
78	4e	N	N	N	206	ce		Î	╬
79	4f	O	O	O	207	cf		Ï	╧
80	50	P	P	P	208	d0		Đ	╨
81	51	Q	Q	Q	209	d1		Ñ	╤
82	52	R	R	R	210	d2		Ò	╥
83	53	S	S	S	211	d3		Ó	╙
84	54	T	T	T	212	d4		Ô	╘
85	55	U	U	U	213	d5		Õ	╒
86	56	V	V	V	214	d6		Ö	╓
87	57	W	W	W	215	d7		×	╫
88	58	X	X	X	216	d8		Ø	╪
89	59	Y	Y	Y	217	d9		Ù	┘
90	5a	Z	Z	Z	218	da		Ú	┌
91	5b	[	[	[	219	db		Û	▉
92	5c	\	\	\	220	dc		Ü	▄
93	5d	]	]	]	221	dd		Ý	▌
94	5e	^	^	^	222	de		Þ	▐
95	5f	_	_	_	223	df		ß	▀
96	60	`	`	`	224	e0		à	α
97	61	a	a	a	225	e1		á	ß
98	62	b	b	b	226	e2		â	Γ

Таблица 190. Десятичные и шестнадцатеричные коды и символы, используемые приборами CR1000

Десятичный	*Шестнадцатеричный*	*Дисплей с клавиатурой*	*LoggerNet*	*Гипертерминал*	*Десятичный*	*Шестнадцатеричный*	*Дисплей с клавиатурой*	*LoggerNet*	*Гипертерминал*
99	63	c	c	c	227	e3		ã	π
100	64	d	d	d	228	e4		ä	Σ
101	65	e	e	e	229	e5		å	σ
102	66	f	f	f	230	e6		æ	µ
103	67	g	g	g	231	e7		ç	τ
104	68	h	h	h	232	e8		è	Φ
105	69	i	i	i	233	e9		é	Θ
106	6a	j	j	j	234	ea		ê	Ω
107	6b	k	k	k	235	eb		ë	δ
108	6c	l	l	l	236	ec		ì	∞
109	6d	m	m	m	237	ed		í	φ
110	6e	n	n	n	238	ee		î	ε
111	6f	o	o	o	239	ef		ï	∩
112	70	P	P	P	240	f0		ð	≡
113	71	q	q	q	241	f1		ñ	±
114	72	r	r	r	242	f2		ò	≥
115	73	s	s	s	243	f3		ó	≤
116	74	t	t	t	244	f4		ô	⌠
117	75	u	u	u	245	f5		õ	⌡
118	76	v	v	v	246	f6		ö	÷
119	77	w	w	w	247	f7		÷	≈
120	78	x	x	x	248	f8		ø	°
121	79	y	y	y	249	f9		ù	.
122	7a	z	z	z	250	fa		ú	.
123	7b	{	{	{	251	fb		û	√
124	7c	|	|	|	252	fc		ü	^{n}
125	7d	}	}	}	253	fd		ý	^{2}
126	7e	~	~	~	254	fe		þ	■
127	7f		□	⌂	255	ff		ÿ	

Приложение E. Формат данных FP2

Данные в формате FP2 представляют собой двоичные значения, передаваемые в обратном порядке байтов. См. приложение *Порядок следования байтов (стр. 643)*. Путем представления битов в каждой паре байтов как ABCDEFGHIJKLMNOP биты описаны в таблице *Описание битов в формате данных FP2 (стр. 641)*.

Таблица 191. Описание битов в формате данных FP2

Бит	*Описание*
A	Полярность. 0 = +, 1 = -
B, C	Указатели разрядов десятичного числа, описанные в таблице *Биты указателей разрядов десятичного числа в формате FP2*.
D - P	13-битное двоичное значение. D является *старшим битом (стр. 249)*. Максимальное 13-битное значение составляет 8191, но компания Campbell Scientific задает в качестве максимально допустимого значения 7999.

Указатели разрядов десятичного числа можно рассматривать как степень с отрицательным основанием –10 с указателями, представленными в таблице *Биты указателей разрядов десятичного числа в формате FP2 (стр. 641)*.

Таблица 192. Биты указателей разрядов десятичного числа в формате FP2

B	C	Десятичный разряд
0	0	XXXX,
0	1	XXX,X
1	0	XX,XX
1	1	X,XXX

Приложение F. Порядок следования байтов

Синонимы:

- «Порядок байтов» и «порядок следования байтов»
- «Прямой порядок байтов» и «первый младший байт»
- «Обратный порядок байтов» и «первый старший байт»

Порядок следования байтов лежит в основе работы процессора прибора. Порядок определяется производителем процессора. Доступное описание порядка следования байтов можно найти на сайте Wikipedia.com. Результаты использования порядка следования байтов для такого прибора, как регистратор данных CR1000, обычно скрыты в операционной системе. Однако следующие функции CR1000 раскрывают порядок следования байтов и могут потребовать программирования для настройки несоответствий:

- Программирование последовательного ввода/вывода (*Последовательный ввод/вывод: считывание последовательных данных* *(стр. 245)*)
- Программирование Modbus (*Modbus* *(стр. 411)*)
- Команда **MoveBytes()** (см. *Справку к редактору CRBasic*)
- Команда **SDMGeneric()** (см. *Справку к редактору CRBasic*)
- Некоторые команды PakBus, такие как GetDataRecord (см. *Справку к редактору CRBasic*)

Например, когда регистратор данных CR1000 принимает данные от регистратора данных CR9000, порядок байтов для четырех байтов IEEE4 или целочисленного значения данных должен быть изменен на обратный, перед тем как значение будет правильно представлено в CR1000.

Таблица 193. Порядок следования байтов в приборах компании Campbell Scientific

Приборы с прямым порядком байтов	*Приборы с обратным порядком байтов*
Регистратор данных CR6	Регистраторы данных серии CR200(X)
Регистратор данных CR9000X	Регистраторы данных серии CR800
Регистраторы данных серии CRVW	Регистратор данных CR1000
Регистрирующий датчик CRS451	Регистратор данных CR3000
	Регистратор данных CR5000

Использование порядка следования байтов рассмотрено в следующих разделах:

- Раздел *Считывание регистров Modbus в обратном формате* *(стр. 415)*
- Приложение *Формат данных FP2* *(стр. 641)*

Приложение G. Перечни продуктов для поддержки

Вспомогательные продукты обеспечивают и расширяют возможности CR1000 по измерению и управлению. Перечисленные продукты производятся компанией группы Campbell Scientific, если не указано иное. Ознакомьтесь с литературой по продуктам на сайте *www.campbellsci.com* или обратитесь к специалисту по внедрению компании Campbell Scientific, чтобы определить, какие продукты лучше всего подходят для конкретных приложений. Приведенные ниже списки не являются исчерпывающими, но отражают состояние на дату публикации руководства.

G.1 Регистраторы данных — список

Связанные темы:
• *Регистратор данных — быстрый запуск (стр. 43)*
• *Регистратор данных — общие сведения (стр. 75)*
• *Регистратор данных — список (стр. 645)*

Другие устройства регистрации данных Campbell Scientific можно использовать в сетях с CR1000. Сигналы данных и управляющие сигналы могут проходить от устройства к устройству, когда CR1000 играет роль ведущего, однорангового или ведомого устройства. Регистраторы данных взаимодействуют в сети через PakBus®. Modbus, DNP3, RS-232, SDI-12 или CANbus с помощью модуля SDM-CAN.

Таблица 194. Регистраторы данных

Модель	*Описание*
Регистраторы данных серии CR200X	Ограниченный вход, без возможности расширения. Подходят для сети станций с небольшим числом конкретных входов. Некоторые модели оснащены встроенными радио приемопередатчиками для широкополосной связи в различных диапазонах частот.
Регистраторы данных серии CR800	Ограниченный вход, но с возможностью расширения. Подходят для сети станций с небольшим числом конкретных входов. Устройство CR850 оснащено встроенной клавиатурой и дисплеем.
Регистратор данных для измерения и контроля CR6	12 универсальных клемм входов поддерживают аналоговые или импульсные входы. 4 клемм входа/выхода можно настроить для управления или для нескольких протоколов связи. Этот инструмент очень гибок, допускает расширение и может работать в сети.
Система измерения и контроля CR1000	16 клемм аналогового входа, две клеммы импульсного входа, восемь клемм управления / ввода/вывода. С возможностью расширения.
Микрорегистратор CR3000	28 клемм аналогового входа, четыре клеммы импульсного входа, восемь клемм управления / ввода/вывода. Быстрее, чем CR1000. С возможностью расширения.
Модули измерения, контроля и ввода/вывода серии CR9000X	Высокая скорость, возможность настройки, модульность, возможность расширения

G.2 Внешнее оборудование для измерения и контроля — списки

Связанные темы:
• *Внешнее оборудование для измерения и контроля — общие сведения (стр. 85)*

• *Внешнее оборудование для измерения и контроля — подробная информация (стр. 366)*
• *Внешнее оборудование для измерения и контроля — списки (стр. 645)*

G.3 Списки входных модулей датчиков

Внешнее входное оборудование расширяет входной номинал датчика CR1000, нормирует сигналы датчиков или распределяет нагрузку по измерениям.

G.3.1 Список модулей аналогового ввода

Модули аналогового ввода увеличивают возможности CR1000. Некоторые мультиплексоры допускают мультиплексирование клемм возбуждения (аналоговый выход).

Таблица 195. Модули аналогового ввода

Модель	Описание
AM16/32B	64 канала — возможность настройки для многих типов датчиков. Мультиплексирование аналоговых входов и возбуждения.
AM25T	25 каналов — мультиплексирование аналоговых входов. Предназначено для термопар и дифференциальных входов

G.3.2 Список модулей импульсного входа

Связанные темы:
Модули ввода по низкоуровневому переменному току — общие сведения (стр. 367)
Низкоуровневые измерения с переменным током — подробная информация (стр. 352)
• *Импульсные модули ввода — списки (стр. 646)*

Эти модули расширяют и увеличивают возможности импульсного и частотного входа.

Таблица 196. Модули импульсного входа

Модель	Описание
SDM-INT8	Восьмиканальный таймер интервалов
SDM-SW8A	Восьмиканальный модуль переключения-замыкания.
LLAC4	Четырехканальный модуль переменного тока низкого уровня

G.3.3 Список модулей последовательного ввода/вывода

Внешнее оборудование ввода/вывода расширяют и дополняют возможности ввода и нормируют последовательные сигналы.

Таблица 197. Список модулей последовательного ввода/вывода

Модель	Описание
SDM-SIO1	Одноканальный модуль расширения ввода/вывода
SDM-SIO4	Четырехканальный модуль расширения ввода/вывода
SDM-IO16	16-канальный модуль расширения ввода/вывода

G.3.4 Список модулей вибрационных струнных входов

Модули вибрационных струнных входов улучшают измерения вибрационных струнных датчиков.

Таблица 198. Модули вибрационных струнных входов

Модель	*Описание*
CDM-VW300	Двухканальное динамическое вибрационное струнное измерительное устройство VSPECT
CDM-VW305	Восьмиканальное динамическое вибрационное струнное измерительное устройство VSPECT
Серия AVW200	Двухканальное статическое вибрационное струнное измерительное устройство VSPECT

G.3.5 Список пассивных преобразователей сигнала

Преобразователи сигнала изменяют выход датчика так, чтобы он был совместим с CR1000.

G.3.5.1 Список модулей резистивного моста TIM

Таблица 199. Модули резистивного моста TIM[1]

Модель	*Описание*
4WFBS120	120 Ом, модуль TIM в четырехжильном подключении с полным мостом
4WFBS350	350 Ом, модуль TIM в четырехжильном подключении с полным мостом
4WFBS1K	1 кОм, модуль TIM в четырехжильном подключении с полным мостом
3WHB10K	10 кОм, модуль TIM в трехжильном полумостовом подключении
4WHB10K	10 кОм, модуль TIM в четырехжильном полумостовом подключении
4WPB100	100 Ом, модуль TIM в четырехжильном подключении с мостом PRT
4WPB1K	1 кОм, модуль TIM в четырехжильном подключении с мостом PRT

[1] **TIM —** модуль терминального ввода

G.3.5.2 Список модулей делителей напряжения

Таблица 200. Модули делителей напряжения

Модель	*Описание*
VDIV10:1	Делитель напряжения 10:1
VDIV2:1	Делитель напряжения 2:1
CVD20	Шестиканальный делитель напряжения 20:1

G.3.5.3 Список модулей токовых шунтов

Таблица 201. Модули токовых шунтов

Модель	*Описание*
CURS100	Модуль токового шунта на 100 Ом

G.3.5.4 Список ограничителей переходного напряжения

Таблица 202. Ограничители переходного напряжения	
Модель	***Описание***
16980	Устройство защиты от скачков напряжения для радио УВЧ/ОВЧ
14462	Устройство защиты от скачков напряжения для радио RF401 и регистратора данных CR206
16982	Устройство защиты от скачков напряжения для радио RF416 и регистратора данных CR216
16981	Устройство защиты от скачков напряжения для передатчиков GOES
6536	4-проводной стабилизатор напряжения для SRM-5A
4330	2-проводной стабилизатор напряжения для модемов стационарных телефонов
SVP48	Многоканальный стабилизатор напряжения общего назначения

G.3.6 Список крышек клеммных колодок

Клеммные колодки закрывают и изолируют входные клеммы для повышения качества измерений, осуществляемых термопарами.

Таблица 203. Крышки клеммных колодок	
Регистратор данных	***Номер детали крышки клеммной колодки***
CR6	Крышка отсутствует
CR800	Крышка отсутствует
CR1000	17324
CR3000	18359

G.4 Модули управления ПЛК — списки

Связанные темы:
- *Управление ПЛК — общие сведения* *(стр. 74)*
- *Управление ПЛК — подробная информация* *(стр. 244)*
- *Модули управления ПЛК — общие сведения* *(стр. 368)*
- *Модули управления ПЛК — списки* *(стр. 648)*
- *Контроль ПЛК — команды* *(стр. 562)*
- Переключаемое выходное напряжение — технические характеристики
- Переключаемое выходное напряжение — общие сведения
- *Переключаемое выходное напряжение — подробная информация* *(стр. 103)*

G.4.1 Список модулей цифрового ввода/вывода

Модули расширения ввода/вывода увеличивают число каналов для чтения или вывода логических сигналов 5 В пост. тока.

Таблица 204. Модули цифрового ввода/вывода	
Модель	***Описание***
SDM-IO16	16-канальный модуль расширения ввода/вывода

G.4.2 Список модулей непрерывного аналогового вывода (Continuous Analog Output, CAO)

Модули CAO позволяют CR1000 выводить непрерывное настраиваемое напряжение, которое может потребоваться для ленточных диаграмм и приложений, в которых требуется плавное регулирование.

Таблица 205. Модули непрерывного аналогового вывода (CAO)

Модель	*Описание*
SDM-AO4A	Четырехканальный, выход непрерывного аналогового напряжения
SDM-CVO4	Четырехканальный, аналоговый выход непрерывного напряжения и тока

G.4.3 Управляющие цепи реле — список

Управляющие цепи реле позволяют CR1000 управлять большими напряжениями.

Таблица 206. Управляющие цепи реле — продукты

Модель	*Описание*
A21REL-12	Четыре реле, управляемые четырьмя портами управления
A6REL-12	Шесть реле, управляемых шестью портами управления / с переводом на ручное управление
LR4	Четырехканальное реле с блокировкой
SDM-CD8S	Восьмиканальный релейный контроллер пост. тока
SDM-CD16AC	16-канальный релейный контроллер перем. тока
SDM-CD16S	16-канальный релейный контроллер пост. тока
SDM-CD16D	16-канальный модуль вывода 0 или 5 В пост. тока
SW12V	Одноканальная цепь управления 12 В пост. тока

G.4.4 Список модулей возбуждения током

Модули возбуждения током, как правило, используются с блоками матричного потенциала почвы 229-L.

Таблица 207. Модули возбуждения током

Модель	*Описание*
CE4	Четырехканальный модуль возбуждения током
CE8	Восьмиканальный модуль возбуждения током

G.5 Датчики — списки

Связанные темы:
- *Датчики — краткое руководство* *(стр. 42)*
- *Измерения — общие сведения* *(стр. 62)*
- *Измерения — подробная информация* *(стр. 303)*
- *Датчики — списки* *(стр. 649)*

Большинство электронных датчиков, независимо от производителя, совместимы с CR1000. Для некоторых датчиков требуется внешнее нормирование сигнала. Производительность некоторых датчиков повышается с помощью модулей входа.

G.5.1 Список типов проводных датчиков

Campbell Scientific предлагает следующие типы проводных датчиков для встраивания в системы CR1000. Обратитесь к специалисту Campbell Scientific, чтобы узнать конкретные номера модели и руководство по интеграции.

Таблица 208. Типы проводных датчиков

Температура воздуха	Давление
	Содержание влаги в дорожном полотне
Относительная влажность	Глубина снега
Барометрическое давление	Эквивалентный слой воды
Проводимость	Тепловой поток в почве
Цифровая камера	Температура почвы
Растворенный кислород	Объемная влажность почвы
Расстояние	Профиль объемной влажности почвы
	Потенциал почвенной влаги
Электрический ток	Солнечная радиация
Электрическое поле (молния)	Деформация
Испарение	Температура поверхности
Переохлажденный дождь и гололед	Помутнение
Влажность и температура топлива	Видимость
Географическое положение (GPS)	Уровень воды
Поток тепла, пара и CO_2	Расход воды
Увлажнение листа Остаточная радиация	Качество воды
ОВП / pH	Пробоотборник для воды
Осадки	Температура воды
Фактическая погода	Скорость ветра / направление ветра

G.5.2 Список беспроводных сетевых датчиков

Беспроводные датчики используют технологию беспроводных датчиков Campbell (Campbell wireless sensor, CWS) на основе радиосвязи с передачей сигнала в широком спектре. Доступны следующие беспроводные датчики.

Таблица 209. Модули беспроводных датчиков

Модель	***Описание***
Серия CWB100	Радиомодуль для регистратора данных.
Серия CWS220	Инфракрасный радиометр
Серия CWS655	Приповерхностный датчик объемной влажности почвы
Серия CWS900	Настраиваемый дистанционный модуль входа датчика

Таблица 210. Типы датчиков, доступные для подключения к CWS900	
Температура воздуха	Относительная влажность
Растворенный кислород	Тепловой поток в почве
Инфракрасная температура поверхности	Температура почвы
Влажность листа	Солнечная радиация
Давление	Температура поверхности
Квантовый датчик	Скорость ветра / направление ветра
Дождь	

G.6 Внешнее оборудование ввода данных и телекоммуникационное внешнее оборудование — списки

Связанные темы:
- *Сбор и обмен данными — краткое руководство (стр. 45)*
- *Сбор и обмен данными — общие сведения (стр. 88)*
- *Сбор и обмен данными — подробная информация (стр. 391)*
- *Периферийное оборудование для сбора и обмена данными — списки (стр. 651)*

Многие телекоммуникационные устройства могут использоваться с регистратором данных CR1000.

G.6.1 Дисплей с клавиатурой — список

Связанные темы:
- *Экранная клавиатура — общие сведения (стр. 83)*
- *Экранная клавиатура — подробная информация (стр. 451)*
- *Экранная клавиатура — список (стр. 651)*
- *Пользовательские меню — общие сведения (стр. 84, стр. 581)*

Таблица 211. Регистратор данных / доступность и совместимость дисплея с клавиатурой[1]	
Модель регистратора данных	***Совместимые дисплеи с клавиатурой***
CR6	*CR1000KD[2] (стр. 511)*, *CD100 (стр. 509)*, CD295
CR800	CR1000KD[2], CD100, CD295
CR850	Встроенный дисплей с клавиатурой, CR1000KD[2], CD100, CD295
CR1000	CR1000KD[2], CD100, CD295
CR3000	Встроенный дисплей с клавиатурой, CR1000KD[2] (требуется специальная ОС), CD100 (требуется специальная ОС),CD295

[1] Дисплеи с клавиатурой либо встраиваются в регистратор данных, либо подключаются через порт **CS I/O**.

[2] CR1000KD можно смонтировать на поверхность с помощью двух отверстий под винты № 4–40 x 0,187 на задней стороне.

G.6.2 Список кабельных коммуникационных устройств с одиночным подключением

Таблица 212. Кабельные коммуникационные устройства с одиночным подключением

Модель	*Описание*
SC32B	Оптически изолированный **CS I/O** к интерфейсу RS-232 на ПК (требуется кабель RS-232)
SC929	**CS I/O** — кабель интерфейса RS-232
SC-USB	Оптически изолированный **RS-232** — кабель USB
17394	**RS-232** — кабель USB (не оптически изолированный)
10873	**RS-232** — кабель RS-232; гнездо с девятью контактами - штекер с девятью контактами
SRM-5A с SC932A	**CS I/O** — RS-232, телефонные модемы ближней связи
SDM-CAN	Регистратор данных — интерфейс шины CAN
FC100	Волоконнооптический модем. В большинстве установок требуется два.

G.6.3 Список кабельных сетевых устройств

Таблица 213. Кабельные сетевые устройства

Модель	*Описание*
MD485	Многоточечный интерфейс RS-485

G.6.4 Каналы TCP/IP — список

Таблица 214. Каналы TCP/IP

Модель	*Описание*
Серия RavenX	Беспроводной, сотовый, с подключением к порту **RS-232**. Ключ PPP/IP должен быть активирован, чтобы можно было использовать IP-стек CR1000.
NL240	Беспроводной сетевой канальный интерфейс с подключением к порту **CS I/O**.
NL201	Сетевой канальный интерфейс с подключением к порту **CS I/O**.
NL115	С подключением к порту **внешнего оборудования**. Использует IP-стек CR1000. Содержит гнездо для платы CF.
NL120	С подключением к порту **внешнего оборудования**. Использует IP-стек CR1000. Без гнезда для платы CF.

G.6.5 Список телефонных модемов

Таблица 215. Телефонные модемы

Модель	*Описание*
COM220	9600 бод
COM320	9600 бод, синтезированная речь
Серия RAVENX	Сотовый сетевой канал

G.6.6 Список радио для частных сетей

Таблица 216. Радио для частных сетей	
Модель	***Описание***
Серия RF401	Широкий спектр. 100 мВт. Подключение CS I/O к удаленному регистратору данных CR1000. Совместимо с RF430.
Серия RF430	Широкий спектр. 100 мВт. USB-подключение к базовому ПК. Совместимо с RF400.
RF450	Широкий спектр. 1 Вт
Серия RF300	ОВЧ / УВЧ. 5 Вт, лицензированное, одночастотное

G.6.7 Список спутниковых приемопередатчиков

Таблица 217. Спутниковые приемопередатчики	
Модель	***Описание***
ST-21	Передатчик Argos
TX320	Передатчик HDR GOES
DCP200	Платформа сбора данных GOES

G.7 Запоминающие устройства — список

Связанные темы:
- *Память — общие сведения* *(стр. 87)*
- *Память — подробная информация* *(стр. 370)*
- *Устройства хранения данных — список* *(стр. 653)*

Запоминающие устройства позволяют собирать данные на месте с помощью небольшого устройства и переносить их на ПК (перенос данных вручную).

Устройства массовой памяти Campbell Scientific подключаются к порту CS I/O CR1000.

Таблица 218. Устройства массовой памяти	
Модель	***Описание***
SC115	Карта флеш-памяти на 2 Гб (флеш накопитель)

Модули памяти на базе платы CF подключаются к порту внешнего оборудования CR1000. Используйте только платы CF промышленного уровня на 16 Гб или меньше.

Таблица 219. Модуль памяти на базе платы CF	
Модель	***Описание***
CFM100	Только гнездо для платы CF.
NL115	Сетевой канал с гнездом для платы CF

G.8 Программная поддержка регистратора данных — списки

Подробнее:
• *ПО поддержки регистратора данных — краткое руководство (стр. 46)*
• *ПО поддержки регистратора данных — общие сведения (стр. 95)*
• *ПО поддержки регистратора данных — подробная информация (стр. 450)*
• *ПО поддержки регистратора данных — списки (стр. 654)*

Campbell Scientific предлагает программные продукты для упрощения программирования и обслуживания CR1000, сбора данных и представления данных. ПО начального уровня (таблица *ПО начального уровня (стр. 654)*) — это продукты, предназначенные для начинающих интеграторов. Программные продукты для поддержки регистраторов данных (таблица *Программная поддержка регистраторов данных (стр. 654)*) объединяют программирование CR1000, телекоммуникации и сбор данных в одном пакете. Клиенты LoggerNet (таблица *Клиенты LoggerNet (стр. 655)*) доступны для расширенных приложений *LoggerNet*. Комплекты для разработки ПО (таблица *Комплекты для разработки ПО (стр. 656)*) предназначены для вариантов применения, для которых стандартные программные продукты не подходят. Доступна также ограниченная программная поддержка для приложений iOS, Android и Linux.

Примечание. Более подробную информацию о ПО, предлагаемом Campbell Scientific, можно найти на сайте *www.campbellsci.com* http://www.campbellsci.com. Для получения рекомендации по выбору ПО, подходящего для вашей сферы деятельности, обратитесь к техническому специалисту Campbell Scientific.

G.8.1 Список ПО начального уровня

Short Cut, PC200W и *VisualWeather* предназначены для начинающих интеграторов, но содержат и функции, полезные для продвинутых приложений.

Таблица 220. ПО начального уровня

Модель	*Описание*
Short Cut	Простой в использовании мастер программирования на CRBasic, графический интерфейс пользователя; ПК, совместим с Windows®.
PC200W Starter Software	Простое в использовании базовое ПО для поддержки *регистратора данных (стр. 512)* для прямых телекоммуникационных соединений, ПК, совместима с Windows®.
VisualWeather	Простое в использовании ПО для поддержки регистратора данных, специализированное для вариантов применения, связанных с погодой и сельским хозяйством, ПК, совместимо с Windows®.

G.8.2 Программная поддержка регистратора данных — список

PC200W, PC400, RTDAQ и *LoggerNet* предоставляют все более широкие возможности для интеграции, программирования, сбора данных и телекоммуникации. Доступна также ограниченная *программная поддержка (стр. 95)* для приложений iOS, Android и Linux.

Таблица 221. Программная поддержка регистратора данных		
Программное обеспечение	***Совместимость***	***Описание***
PC200W Starter Software	ПК, Windows	Базовое ПО поддержки регистратора данных для прямого подключения.
PC400	ПК, Windows	ПО поддержки регистратора данных среднего уровня. Поддерживает отдельные регистраторы данных с большинством телекоммуникационных опций.
LoggerNet	ПК, Windows	ПО поддержки регистратора данных высшего уровня. Поддерживает сети регистраторов данных.
LoggerNet Admin	ПК, Windows	Продвинутый *LoggerNet* для больших сетей регистраторов данных.
LoggerNet Linux	Linux	Включает *LoggerNet Server* для использования в средах Linux и *LoggerNet Remote* для управления сервером из сред Windows.
RTDAQ	ПК, Windows	ПО поддержки регистратора данных для промышленного применения и работы в реальном времени.
VisualWeather	ПК, Windows	ПО поддержки регистратора данных, предназначенное для вариантов применения, связанных с погодой и сельским хозяйством.
LoggerLink	iOS и Android	ПО поддержки регистратора данных для устройств iOS и Android. Только IP-подключения к регистратору данных.

G.8.2.1 Комплекс LoggerNet — список

Комплекс *LoggerNet* использует архитектуру клиент-сервер, которая упрощает реализацию широкого спектра вариантов применения и позволяет приобретать индивидуальное ПО для специфических нужд.

Таблица 222. Комплекс LoggerNet[1,2]	
Программное обеспечение	***Описание***
LoggerNetAdmin	ПО поддержки администратора регистратора данных
LNLinux	Сервер *LoggerNet* на базе Linux
LoggerNetRem	Позволяет осуществлять администрирование *LoggerNetAdmin* через TCP/IP с удаленного ПК.
LNDB	*ПО базы данных LoggerNet*
LoggerNetData	Обеспечивает вывод данных в реальном времени или исторических данных, осуществляет пост-обработку файлов данных и создает отчеты. Включает *Split, RTMC, View Pro* и *Data Filer*.
PC-OPC	Сервер Campbell Scientific OPC. Передает данные регистратора данных в сторонние системы. Графические пакеты, совместимые с OPC.
PakBus Graph	Входит в пакет с *LoggerNet*. Отображает и предоставляет доступ к настройкам сети PakBus.

Таблица 222. Комплекс LoggerNet[1,2]	
Программное обеспечение	***Описание***
RTMCPro	Расширенная версия *RTMC*. *RTMC Pro* предоставляет дополнительные возможности и большую гибкость, включая сигналы тревоги с несколькими состояниями, условия рассылки электронных писем в случае тревоги, гиперссылки и передачу файлов по FTP.
RTMCRT	Позволяет просматривать и печатать визуальные представления данных реального времени с несколькими вкладками. Визуальные представления создаются в *RTMC* или *RTMC Pro*.
RTMC Web Server	Преобразует визуальные представления данных реального времени в файлы HTML, что позволяет совместно использовать их с помощью интернет-браузера.
CSIWEBS	Веб-сервер. Преобразует визуальные представления RTMC и RTMC Pro в HTML.
CSIWEBSL	Веб-сервер для Linux. Преобразует визуальные представления RTMC и RTMC Pro в HTML
[1] Клиенты требуют, чтобы на ПК работал *LoggerNet* (приобретается отдельно).	
[2] Клиенты на базе *RTMC* требуют, чтобы на ПК работал *LoggerNet* или *RTDAQ* (приобретаются отдельно).	

G.8.3 Список программных средств

Таблица 223. Программные средства		
Программное обеспечение	***Совместимость***	***Описание***
Network Planner	ПК, Windows	Доступен как часть комплекса *LoggerNet*. Помогает в разработке сетей и настройке сетевых элементов.
Служебная программа конфигурации устройства (DevConfig)	ПК, Windows	Объединен с PC400, LoggerNet и *RTDAQ*. Кроме того, доступен бесплатно на сайте *www.campbellsci.com*. Используется для задания настроек и обновления операционных систем для устройств Campbell Scientific.

G.8.4 Список комплектов для разработки ПО

Таблица 224. Комплекты для разработки ПО		
Программное обеспечение	***Совместимость***	***Описание***
LoggerNet-SDK	ПК, Windows	Позволяет разработчикам ПО создавать пользовательские клиентские приложения, которые через сервер *LoggerNet* подключаются к любому регистратору данных, поддерживаемому *LoggerNet*. Требуется *LoggerNet*.

Таблица 224. Комплекты для разработки ПО		
Программное обеспечение	***Совместимость***	***Описание***
LoggerNetS-SDK	ПК, Windows	Сервер SDK для LoggerNet. Позволяет разработчикам ПО создавать пользовательские клиентские приложения, которые через сервер *LoggerNet* подключаются к любому регистратору данных, поддерживаемому *LoggerNet*. Содержит полную библиотеку DLL сервера *LoggerNet*, которую можно распространять вместе с пользовательскими клиентскими приложениями.
JAVA-SDK	ПК, Windows	Позволяет разработчикам ПО создавать приложения Java для связи с регистраторами данных.
TDRSDK	ПК, Windows	Комплект разработчика ПО для ПК и Windows для связи с временным рефлектометром TDR100.

G.9 Источники питания — продукты

Связанные темы:
- Источники питания — технические характеристики
- *Источники питания — краткое руководство* *(стр. 44)*
- *Источники питания — общие сведения* *(стр. 85)*
- *Источники питания — общие сведения* *(стр. 100)*
- *Источники питания — продукты* *(стр. 657)*
- *Источники питания* *(стр. 101)*
- *Диагностика — источники питания* *(стр. 494)*

Campbell Scientific предлагает несколько источников питания для CR1000.

G.9.1 Список комбинаций батарея / регулятор

Читайте далее. Информация о том, какие источники питания подходят для тех или иных вариантов применения, можно найти в выпущенном Campbell Scientific примечании по применению, представленном на сайте *www.campbellsci.com*.

Таблица 225. Комбинации батарея / регулятор	
Модель	***Описание***
PS100	12 Ач, перезаряжаемая батарея и регулятор (необходим первичный источник питания).
PS200	12 Ач, интеллектуальные функции, перезаряжаемая батарея и регулятор (необходим первичный источник питания).
PS24	24 Ач, перезаряжаемая батарея и регулятор (необходим первичный источник питания).
PS84	84 Ач, перезаряжаемая батарея, регулятор Sunsaver и корпус (необходим первичный источник питания).

G.9.2 Список батарей

Таблица 226. Батареи	
Модель	***Описание***
BPALK	D-cell. Набор алкалиновых батарей 12 В пост. тока
BP7	7 Ач, герметизированная перезаряжаемая батарея (необходим регулятор и первичный источник питания). Содержит монтажный кронштейн для корпусов Campbell Scientific.
BP12	12 Ач, герметизированная перезаряжаемая батарея (необходим регулятор и первичный источник питания). Содержит монтажный кронштейн для корпусов Campbell Scientific.
BP24	24 Ач, герметизированная перезаряжаемая батарея (необходим регулятор и первичный источник питания). Содержит монтажный кронштейн для корпусов Campbell Scientific.
BP84	84 Ач, герметизированная перезаряжаемая батарея (необходим регулятор и первичный источник питания). Содержит монтажный кронштейн для корпусов Campbell Scientific.

G.9.3 Список регуляторов

Таблица 227. Регуляторы	
Модель	***Описание***
CH100	Регулятор зарядки на 12 В пост. тока (необходим первичный источник энергии)
CH200	Регулятор зарядки на 12 В пост. тока (необходим первичный источник энергии)

G.9.4 Список первичных источников питания

Таблица 228. Первичные источники питания	
Модель	***Описание***
29796	24 В пост. тока, выход 1,67 А, от 100 до 240 В перем. тока, вход 1 А, кабель 5 футов
SP5-L	Солнечная панель на 5 Вт (необходим регулятор)
SP10	Солнечная панель на 10 Вт (необходим регулятор)
SP10R	Солнечная панель на 10 Вт (содержит регулятор)
SP20	Солнечная панель на 20 Вт (необходим регулятор)
SP20R	Солнечная панель на 20 Вт (содержит регулятор)
SP50-L	Солнечная панель на 50 Вт (необходим регулятор)

SP90-L	Солнечная панель на 90 Вт (необходим регулятор)
DCDC18R	Импульсный повышающий регулятор для 12–18 В пост. тока (позволяет использовать напряжение бортовой сети автомобиля для перезарядки герметизированных перезаряжаемых батарей)

G.9.5 Список комплектов источников питания 24 В пост. тока

Таблица 229. Комплекты источников питания 24 В пост. тока

Модель	*Описание*
28370	24 В пост. тока, 3.8 А NEC класс 2 (батарея не включена)
28371	24 В пост. тока, 10 А (батарея не включена)
28372	24 В пост. тока, 20 А (батарея не включена)

G.10 Корпуса — продукты

Таблица 230. Корпуса — продукты

Модель	*Описание*
ENC10/12	Защищенный от атмосферных воздействий корпус 10 x 12 дюймов (не подходит для CR3000)
ENC12/14	Защищенный от атмосферных воздействий корпус 12 x 14 дюймов. Доступна версия с предварительной прокладкой кабелей.
ENC14/16	Защищенный от атмосферных воздействий корпус 14 x 16 дюймов. Доступна версия с предварительной прокладкой кабелей.
ENC16/18	Защищенный от атмосферных воздействий корпус 16 x 18 дюймов. Доступна версия с предварительной прокладкой кабелей.
ENC24/30	Защищенный от атмосферных воздействий корпус 24 x 30 дюймов.
ENC24/30S	Защищенный от атмосферных воздействий корпус из нержавеющей стали 24 x 30 дюймов.

Таблица 231. Корпуса с предварительной прокладкой кабелей

Модель	*Описание*
PWENC12/14	Защищенный от атмосферных воздействий корпус 12 x 14 дюймов с предварительной прокладкой кабелей.
PWENC14/16	Защищенный от атмосферных воздействий корпус 14 x 16 дюймов с предварительной прокладкой кабелей.
PWENC16/18	Защищенный от атмосферных воздействий корпус 16 x 18 дюймов с предварительной прокладкой кабелей.

G.11 Списки штативов, башен и креплений

Таблица 232. Штативы, башни и крепления

Модель	*Описание*
CM106B	Башня со штативом, 3 м (10 футов), гальванизированная сталь
CM110	Башня со штативом, 3 м (10 футов), нержавеющая сталь
CM115	Башня со штативом, 4,5 м (15 футов), нержавеющая сталь
CM120	Башня со штативом, 6 м (20 футов), нержавеющая сталь

UT10	Отдельно стоящая башня, 3 м (10 футов), алюминий
UT20	Отдельно стоящая башня, 6 м (20 футов), алюминий, возможно крепление растяжками
UT30	Отдельно стоящая башня, 10 м (30 футов), алюминий, возможно крепление растяжками
CM375	Мачта 10 м (30 футов), гальванизированная и нержавеющая сталь, необходимо крепление растяжками.
CM300	Мачта 0,58 м (23 дюйма), нержавеющая сталь, отдельно стоящая, штатив, возможно крепление растяжками
CM305	Мачта 1,2 м (47 дюймов), нержавеющая сталь, отдельно стоящая, штатив, возможно крепление растяжками
CM310	Мачта 1.42 м (56 дюйма), нержавеющая сталь, отдельно стоящая, штатив, возможно крепление растяжками

G.12 Список корпусов

Таблица 233. Защита от влаги — продукты

Модель	*Описание*
6714	Влагопоглотитель, пакет на 4 единицы (кол-во: 20). Обычно используется в корпусах ENC для защиты CR1000.
A150-L	Корпус с клеммами для отдельного датчика, вентилируемый, с влагопоглотителем.
4091	Влагопоглотитель, пакет 0,75 г. Как правило, используется с датчиками влажности Sentek.
25366	Туба CS450, CS451, CS455 и CS456 с влагопоглотителем на замену. Как правило, используется с датчиками CS4xx.
10525	Влагопоглотитель и держатель для документов, устанавливается пользователем. Как правило, используется с корпусами ENC.
3885	Влагопоглотитель, пакет на 1/2 единицы (кол-во: 50).
CS210	Датчик влажности в корпусе, кабель 11 дюймов.

Алфавитный указатель

5

5VoltLow ... 603

7

7999 ... 131

A

ANSI ... 507, 637
API ... 92, 423
Argos ... 599
AutoRange ... 308, 309

B

BOOL8 ... 130, 199, 200, 508
Boolean ... 130, 162, 483, 508

C

CAO ... 367
CompactFlash ... 205, 386, 461
cr ... 248
CR1000KD ... 75, 84, 182, 451, 511, 651
CR10X ... 145, 259, 584
CR200 ... 584
CR23X ... 259, 584
CR510 ... 259, 584
CRD привод ... 205, 376, 511
CVI ... 511
CWB100 ... 558

D

DCE ... 81, 513, 514, 521
Default.CR1 ... 116
DevConfig ... 111, 112, 513
DHCP ... 295, 513
DisableVar ... 195, 482
DNP3 ... 91, 408, 597
DNS ... 296, 513
DTE ... 81, 513, 514, 521

E

ESD ... 80, 514, 533
ESS ... 514

F

False (ложь) ... 165
FAT ... 373
FTP ... 516
FTP-клиент ... 294
FTP-сервер ... 294
GOES ... 600

H

Hello-запрос ... 396
Hello-сообщение ... 396
HTML ... 293, 517
HTTP ... 291, 517

I

IEEE4 ... 130, 517
INF ... 482, 517
INMARSAT-C ... 601
IP - Modbus ... 416
IP ... 289, 295, 518, 603
IP-Address PPP ... 518, 603
IP-адрес ... 518, 603
IP-шлюз ... 603

L

lf ... 248
LoggerNet ... 655
LSB ... 248, 249, 643

M

MD5-дайджест ... 520
MeasOff ... 323
Modbus ... 91, 295, 412, 414, 520, 597
MSB ... 248, 249, 643

N

NAN ... 130, 195, 309, 310, 482, 521
NIST ... 521

O

OID ... 309
OMNISAT ... 601
OutputOpt ... 296

P

PakBus 88, 395, 398, 522, 584
PC200W 48, 654
PC400 654
PCM 309
PDM 79, 368
Ping 295, 398, 523, 603
PPP — имя пользователя 603
PPP — настройки 603
PPP — ответ на вызов 603
PPP 289, 593
PTemp 328
PWM (ШИМ) 79, 368

R

RevDiff 323
RevEx 323
RS-232 ..... 48, 62, 73, 97, 249, 490, 526, 603
RTDAQ 654
RTU 413
RX 249

S

SCADA 91, 408, 597
SDI-12 ..... 97, 267, 270, 527, 555
SDM 62, 78, 527
Short Cut 50, 654
SMTP 296, 528
SNMP 295
SNP 528
SP 249
SRAM 370, 374
SubScan 156

T

TCP 289, 295, 593
TCP/IP 290, 530
Telnet 295, 530
TIM 367
TrigVar 195, 196
True (истина) 165
TTL 531
TX 249

U

UDP 593
UINT2 ..... 130, 531
USB: накопитель 116, 375, 386, 520, 653
USR 375

V

Visual Weather 654

W

Web API 92, 423

X

XML 533
XOR (исключающее ИЛИ) 565

А

Автобод 584
Автоматическая калибровка 323
Агрегация выражений 128
Адрес — Modbus 414
Адрес — PakBus 603
Адрес — SDI-12 269
Адрес PakBus 393, 394, 603
Адрес 393, 394, 603
Адресат данных 541
Амперная нагрузка 103
Амперы (А) 507
Аналоговое управление 367
Аналогово-цифровое преобразование 305, 326, 337, 507
Аналогово-цифровое 305, 326, 337, 507
Аналоговые измерения 482
Аналоговый вход 65, 97
Аналоговый выход 79, 97, 367, 552
Аналоговый датчик 364
Аналоговый 62, 507
Анемометр 70
Арифметика 161
Арифметические функции 568
Асинхронная связь 78, 247
Атрибуты файла 383
Атрибуты 383

Б

Баланс мощности 101, 281, 282
Безопасность 92, 603
Бесконечный 482
Битовый сдвиг 565
Биты данных 248
Блокировка 92
Бод 48, 111, 490, 584, 597

Большая программа 391

В

В перем. тока..... 532
В пост. тока..... 532
Введение 33
Веб-сервер 291
Веб-страница 593
Вектор ветра 296, 298, 299
Вектор 298, 299
Версия OS 603
Включаемый файл..... 147, 603
Включение..... 386
Влага..... 93, 99
Влагопоглотитель 93, 99, 513
Влажность 93, 99
Внешнее оборудование для печати..... 524
Внешнее оборудование управления 366
Внешнее оборудование 366, 523
Внешний источник питания..... 80
Внутренняя батарея 45, 94, 474
Внутренняя калибровка..... 344
Возбуждение 104, 514, 552
Вольтметр 532
Вольты 532
Восстановление данных 504
Вперед 33
Времена выполнения команд..... 550
Время выполнения 515
Время запуска..... 603
Время крайних значений..... 62, 78
Время обработки 603
Время сканирования 154, 527
Время успокоения сигнала 317, 319
Время успокоения..... 316, 317, 318, 319, 320, 321, 364
Время..... 202, 463
Встроенное ПО 86
Входной регистр 413
Входные пределы..... 97, 305, 310
Входы/выходы управления..... 97, 510
Выполнение при компиляции 544
Выполнение..... 151
Выражение — логическое..... 164
Выражение — строка 166
Выражение 160, 162, 163, 165, 515
Выход за пределы диапазона 239, 482, 521
Выход..... 522
Выходной контакт RS-232..... 633
Выходной контакт 633
Выходной массив..... 522

Г

Газоразрядные трубки..... 105
Генератор 50, 125
Герц..... 517
Гипсовый блок..... 341
Глобальная переменная 517
Глоссарий 507
Глубина буфера..... 603
Градиент 329, 330
Графики 455

Д

Дата ОС 603
Дата..... 463
Датчик — аналоговый..... 305
Датчик — мост..... 337
Датчик — напряжение 305
Датчик — термопара 327
Датчик RS-232..... 245, 364
Датчик SDI-12..... 364
Датчик давления 321
Датчик токовой петли 66, 80, 337
Датчик 42, 85, 649
Двоичное управление..... 368
Двоичный формат..... 139
Двоичный..... 508
Деактивировать переменную..... 145, 146, 195,482
Деформация..... 342
Диагностика — электропитание 495
Диагностика 550
Диапазон аналоговых входов 97, 308
Диапазон входов..... 97, 308
Диапазон рабочих температур 99
Диапазон температур 99
Диодная цепь (схема ИЛИ) 102
Диспетчер файлов..... 603
Дисплей — пользовательский 455
Дисплей с клавиатурой 83, 84, 182, 451, 581
Дисплей 83, 451
Дифференциальный..... 64, 65, 513
Длина кабеля 317, 364
Длина провода 364
Длинный провод..... 317
Дни записи данных 603
Дождемер 364
Документация..... 126
Долговременная настройка 115
Доступ к таблице данных..... 592
Драйвер реле 105, 369
Дуплекс 248

З

Заголовок таблицы данных..... 165
Задание времени и даты..... 463

Задание временных характеристик 307
Задача..... 152, 530
Заземление 80, 105, 514
Заземление 80, 94, 105, 107, 310, 517
Закон Ома..... 522
Замкнутый интервал..... 145
Запись RS-232..... 362
Запись SDI-12..... 363
Запись TTL 362
Заполнить память и остановиться 370, 515
Запоминающее устройство..... 116, 375, 386, 520, 653
Защита 93
Защита ESD 105, 107
Защита от молнии 107
Защита от перенапряжения..... 101, 105, 107

И

ИБП 44, 85, 100, 531
Изменение полярности 325
Измерение — внешнее оборудование . 366
Измерение — коды операций..... 603
Измерение — команда 158, 550
Измерение — ошибка..... 319
Измерение — последовательность 307
Измерение — продолжительность..... 603
Измерение — расчет времени 307
Измерение — синхронизация..... 365
Измерение SDI-12..... 482
Измерение импульса..... 553
Измерение милливольт..... 305
Измерение напряжения 305, 551
Измерение с помощью термопары 109, 327, 551
Измерение состояния 78
Измерение..... 303
Измерительный преобразователь 42, 321
Измерительный прибор..... 42, 649
Измерительный усилитель 305
Имена файлов..... 389
Импульс 62, 525
Импульсно-временная модуляция..... 79, 368
Импульсный вход..... 69, 70
Импульсный датчик 364
Имя таблицы данных..... 129, 603
Имя..... 159, 603
Инициализация переменной..... 137
Инициализировать..... 129
Инициировать связь 290, 392, 518, 597
Инициируемый вручную 520
Интеграция 315, 316
Интегрированная обработка..... 570
Интервал выполнения 154
Интервал сканирования 97, 154
Интервал..... 512
Интерфейс PPP 603
Информационные услуги 289, 593
Информация PakBus 603
Информация PPP 603
Информация TCP 603
Информация по IP 603
Информация по TCP/IP 603
Ионный датчик 109
Искровой разрядник 105

К

Калибровка — в полевых условиях 210
Калибровка — в полевых условиях: двухточечная..... 213
Калибровка — в полевых условиях: коэффициент наклона / смещение..... 218
Калибровка — в полевых условиях: нуль 214
Калибровка — в полевых условиях: пример 213
Калибровка — в полевых условиях: смещение..... 216
Калибровка — калибровка в полевых условиях только коэффициент наклона 220
Калибровка — одноточечная калибровка в полевых условиях 212
Калибровка — ошибка..... 603
Калибровка — ручная калибровка в полевых условиях..... 211
Калибровка — фоновая 603
Калибровка — функции..... 598
Калибровка в полевых условиях — множитель 219
Калибровка в полевых условиях — нуль 215
Калибровка в полевых условиях — смещение..... 217, 219
Калибровка в полевых условиях — только множитель 221
Калибровка в полевых условиях 73, 210
Калибровка шунта..... 226, 227
Калибровка 73, 94, 153, 211, 323,

3
44
Канал ввода 65
Карта устройств 398
Катушка 412
Квитирование связи 517
Клемма 12 В 80, 105
Клемма 5 В 80
Клеммная коробка 337
Клиенты 655
Код запуска 603
Код платы 603
Код 509
Коды операторов 603
Коды функций — Modbus 415
Колебания 526
Количество записей 147
Кольцевая линия (контакт 3) 525
Кольцевая память 370, 526
Команда SDI-12 269
Команда 158
Команды — ABS 568
Команды — AcceptDataRecords 584
Команды — ACOS 566
Команды — ACPower 556
Команды — AddPrecise 572
Команды — Alias 123, 129, 138, 159, 538
Команды — AM25T 559
Команды — AND 565
Команды — AngleDegrees 537
Команды — ArgosData 599
Команды — ArgosDataRepeat 599
Команды — ArgosError 599
Команды — ArgosSetup 599
Команды — ArgosTransmit 599
Команды — ArrayIndex 589
Команды — ArrayLength 589
Команды — As 538
Команды — ASCII 507, 575, 637
Команды — ASIN 566
Команды — ATN 566
Команды — ATN2 566
Команды — Average 542
Команды — AvgRun 572
Команды — AvgSpa 571
Команды — AVW200 559
Команды — Battery 44, 85, 100, 101, 281, 474, 495, 550, 603
Команды — BeginProg / EndProg 545
Команды — BrFull 551
Команды — BrFull6W 551
Команды — BrHalf 551
Команды — BrHalf3W 551
Команды — BrHalf4W 551
Команды — Broadcast 396, 585
Команды — CalFile 589
Команды — Calibrate 598, 602
Команды — Call 545
Команды — CallTable 545
Команды — Case 545
Команды — CDM_VW300Config 559
Команды — CDM_VW300Dynamic 559
Команды — CDM_VW300Rainflow 559
Команды — CDM_VW300Static 559
Команды — Ceiling 568
Команды — CheckPort 554
Команды — CheckSum 575
Команды — CHR 575
Команды — ClockChange 578
Команды — ClockReport 578, 585
Команды — ClockSet 578
Команды — ComPortIsActive 550
Команды — Const 138, 539
Команды — ConstTable / EndConstTable 539
Команды — COS 566
Команды — COSH 566
Команды — Covariance 542
Команды — CovSpa 571
Команды — CPISpeed 559
Команды — CS110 556
Команды — CS110Shutter 556
Команды — CS616 556
Команды — CS7500 556
Команды — CSAT3 556
Команды — CWB100 558
Команды — CWB100Diagnostics 558
Команды — CWB100Routes 558
Команды — CWB100RSSI 558
Команды — Data / Read / Restore 548
Команды — DataEvent 540
Команды — DataGram 585
Команды — DataInterval 144, 540
Команды — DataLong / Read / Restore 548
Команды — DataTable / EndTable 143, 540
Команды — DataTime 540
Команды — DaylightSaving 578
Команды — DaylightSavingUS 578
Команды — Delay 545
Команды — DewPoint 570
Команды — DHCPRenew 593
Команды — DialModem 597
Команды — DialSequence / EndDialSequence 585
Команды — DialVoice 580

Команды — Dim 513, 538
Команды — DisplayLine 581
Команды — DisplayMenu / EndMenu 581
Команды — DisplayValue 581
Команды — DNP 409, 597
Команды — DNPUpdate 409, 597
Команды — DNPVariable 597
Команды — Do / While / Until / Exit Do / Loop. 545
Команды — EC100 556
Команды — EC100Configure 556
Команды — EMailRecv 593
Команды — EMailSend 593
Команды — EndSequence 545
Команды — EQV 565
Команды — Erase 589
Команды — ESSInitialize 544
Команды — ESSVariables 538
Команды — EthernetPower 593
Команды — ETsz 544
Команды — ExciteV 552
Команды — Exit 545
Команды — EXP 568
Команды — FFT 542
Команды — FFTSpa 571
Команды — FieldCal 213, 598
Команды — FieldCalStrain 223, 225, 598
Команды — FieldNames 542
Команды — FileClose 589
Команды — FileCopy 589
Команды — FileEncrypt 589
Команды — FileList 589
Команды — FileManage 589
Команды — FileMark 592
Команды — FileOpen 589
Команды — FileRead 589
Команды — FileReadLine 589
Команды — FileRename 589
Команды — FileSize 589
Команды — FileTime 589
Команды — FileWrite 589
Команды — FillStop 540
Команды — FindSpa 589
Команды — FIX 568
Команды — FLOAT 130, 162, 163, 483, 516
Команды — Floor 568
Команды — For / To / Step / ExitFor / Next 545
Команды — FormatFloat 575
Команды — FormatLong 575
Команды — FormatLongLong 575
Команды — FP2 ..... 130, 516, 641
Команды — FRAC 568
Команды — FTPClient 593
Команды — Function / Return / Exit Function / EndFunction 602
Команды — GetDataRecord 585
Команды — GetFile 585
Команды — GetRecord 592
Команды — GetVariables 585
Команды — GOESData 600
Команды — GOESGPS 600
Команды — GOESSetup 600
Команды — GOESStatus 600
Команды — GPS 556
Команды — HEX 575
Команды — HexToDec 575
Команды — Histogram 573
Команды — Histogram4D 573
Команды — HTTPGet 593
Команды — HTTPOut 593
Команды — HTTPPost 593
Команды — HTTPPut 593
Команды — HydraProbe 556
Команды — If / Then / Else / ElseIf / EndIf 545
Команды — IfTime 578
Команды — IIF 565
Команды — IMP 565
Команды — Include 589
Команды — INSATData 601
Команды — INSATSetup 601
Команды — INSATStatus 601
Команды — InStr 575
Команды — InstructionTimes 550
Команды — INT 568
Команды — INTDV 568
Команды — IPInfo 593
Команды — IPNetPower 593
Команды — IPRoute 593
Команды — IPTrace 593
Команды — Is 593
Команды — Left 575
Команды — Len 575
Команды — LevelCrossing 573
Команды — LI7200 556
Команды — LI7700 556
Команды — LN or LOG 568
Команды — LoadFieldCal 598
Команды — LOG10 568
Команды — LONG 130, 162, 163, 483, 519
Команды — LowerCase 575
Команды — LTrim 575
Команды — Maximum 542
Команды — MaxSpa 571

Команды — Median 542
Команды — MenuItem 581
Команды — MenuPick 581
Команды — MenuRecompile 581
Команды — Mid 575
Команды — Minimum 542
Команды — MinSpa 571
Команды — MOD 568
Команды — ModBusMaster 414, 597
Команды — ModBusSlave 414, 597
Команды — ModemCallback 597
Команды — ModemHangup / EndModemHangup 597
Команды — Moment 542
Команды — Move 589
Команды — MoveBytes 414, 583
Команды — MovePrecise 544
Команды — MuxSelect 559
Команды — Network 585
Команды — NetworkTimeProtocol 593
Команды — NewFieldCal 598
Команды — NewFieldNames 538
Команды — NewFile 589
Команды — NOT 565
Команды — OmniSatData 601
Команды — OmniSatRandomSetup ... 601
Команды — OmniSatStatus 601
Команды — OmniSatSTSetup 601
Команды — OpenInterval 145, 540
Команды — Optional 602
Команды — OR 565
Команды — PakBusClock 578, 585
Команды — PanelTemp 550
Команды — PeakValley 542
Команды — PeriodAvg 554
Команды — PingIP 593
Команды — PipelineMode 537
Команды — PortGet 554
Команды — PortPairConfig 554
Команды — PortsConfig 554
Команды — PortSet 554
Команды — PPPClose 593
Команды — PPPOpen 593
Команды — PreserveVariables 538
Команды — PRT 234, 235, 570
Команды — PRTCalc 570
Команды — Public 524, 538
Команды — PulseCount 553
Команды — PulseCountReset 544
Команды — PulsePort 554
Команды — PWR 568
Команды — RainFlow 573
Команды — RainFlowSample 544, 573
Команды — Randomize 572
Команды — Read 548
Команды — ReadIO 554
Команды — ReadOnly 129, 538
Команды — RealTime 550, 578
Команды — RectPolar 568
Команды — Replace 575
Команды — ResetTable 592
Команды — Resistance 337
Команды — Restore 548
Команды — Right 575
Команды — RMSSpa 571
Команды — RND 572
Команды — Round 568
Команды — Route 585
Команды — RoutersNeighbors 585
Команды — Routes 585, 603
Команды — RTrim 575
Команды — RunProgram 589
Команды — Sample 542
Команды — SampleFieldCal 542, 598
Команды — SampleMaxMin 542
Команды — SatVP 570
Команды — Scan / ExitScan / ContinueScan / NextScan ... 545
Команды - SDI-12 268
Команды — SDI12Recorder 277, 555
Команды — SDI12SensorResponse 276, 555
Команды — SDI12SensorSetup 276, 555
Команды — SDMAO4 559
Команды — SDMAO4A 559
Команды — SDMCAN 559
Команды — SDMCD16AC 559
Команды — SDMCD16Mask 559
Команды — SDMCVO4 559
Команды — SDMGeneric 559
Команды — SDMINT8 559
Команды — SDMIO16 559
Команды — SDMSIO4 559
Команды — SDMSpeed 559
Команды — SDMSW8A 559
Команды — SDMTrigger 559
Команды — SDMX50 559
Команды — SecsSince1990 578
Команды — Select Case / Case / Case Is / Case Else / EndSelect 545
Команды — SemaphoreGet 153, 548
Команды — SemaphoreRelease 548
Команды — SendData 585
Команды — SendFile 585
Команды — SendGetVariables 585
Команды — SendTableDef 585
Команды — SendVariables 585
Команды — SequentialMode 537
Команды — SerialBrk 583
Команды — SerialClose 250, 583
Команды — SerialFlush 250, 583

Команды — SerialIn 250, 583
Команды — SerialInBlock..... 250, 583
Команды — SerialInChk 583
Команды — SerialInRecord..... 250, 583
Команды — SerialOpen..... 250, 583
Команды — SerialOut..... 250, 583
Команды — SerialOutBlock..... 250, 583
Команды — SetSecurity 537
Команды — SetSetting 592
Команды — SetStatus 592
Команды — SGN 568
Команды — ShutDownBegin..... 548
Команды — ShutDownEnd..... 548
Команды — Signatures 92, 528, 550
Команды — SIN..... 566
Команды — SINH 566
Команды — SlowSequence..... 155, 528, 545, 603
Команды — SNMPVariable 593
Команды — SolarPosition 570
Команды — SortSpa..... 571
Команды — SplitStr 575
Команды — SPrintF..... 575
Команды — Sqr 568
Команды — StaticRoute 585
Команды — StationName..... 129, 537, 603
Команды — StdDev..... 542
Команды — StdDevSpa 571
Команды — StrainCalc 570
Команды — StrComp 575
Команды — STRING 130, 483, 529
Команды — Sub / Exit Sub / End Sub..... 537
Команды — SubMenu / EndSubMenu 581
Команды — SubScan / NextSubScan 156, 545
Команды — SW12..... 552
Команды — TableHide 145, 540
Команды — TableName.EventCount 592
Команды — TableName.FieldName 592
Команды — TableName.Output 592
Команды — TableName.Record 592
Команды — TableName.TableFull 592
Команды — TableName.TableSize 592
Команды — TableName.TimeStamp 592
Команды — TAN..... 566
Команды — TANH..... 566
Команды — TCDiff 551
Команды — TCPClose 593
Команды — TCPOpen..... 593
Команды — TCSe 551
Команды — TDR100 559
Команды — TGA 556
Команды — Therm107 556
Команды — Therm108 556
Команды — Therm109 556
Команды — Thermistor..... 551
Команды — TimedControl 559
Команды — TimeIntoInterval..... 578
Команды — TimeIsBetween..... 545, 578
Команды — Timer..... 578
Команды — TimerIO..... 554
Команды — TimeUntilTransmit 585
Команды — Totalize 542
Команды — TotalRun 563
Команды — TriggerSequence 545
Команды — Trim..... 575
Команды — UDPDataGram..... 593
Команды — UDPOpen 593
Команды — Units..... 129, 138, 538
Команды — UpperCase..... 575
Команды — VaporPressure..... 570
Команды — VibratingWire 554
Команды — VoiceBeg / EndVoice..... 580
Команды — VoiceHangup 580
Команды — VoiceKey..... 580
Команды — VoiceNumber..... 580
Команды — VoicePhrases..... 580
Команды — VoiceSetup 580
Команды — VoiceSpeak..... 580
Команды — VoltDiff 551
Команды — VoltSE..... 551
Команды — WaitDigTrig 545
Команды — WaitTriggerSequence..... 545
Команды — WebPageBegin / WebPageEnd 593
Команды — WetDryBulb..... 570
Команды — While / Wend 545
Команды — WindVector 544
Команды — WorstCase..... 592
Команды — WriteIO..... 554
Команды — XMLParse..... 593
Команды — XOR 565
Команды ввода/вывода..... 518
Команды обработки — выход..... 512
Команды обработки..... 524
Команды работы со строками 575
Команды управления..... 524
Комментарий..... 126
Коммуникационные порты 603
Коммутационная панель 43, 44, 47, 76, 328
Компилировать программу 180
Конкатенация 283
Константа — предварительно определенная..... 138
Константа..... 129, 137, 138, 510
Контакт CLK/HS..... 633
Контакт ME 633
Контакт RING..... 633
Контакт RX..... 633
Контакт SDE 633
Контакт TX 633

Контур заземления 109
Конфигурация дисплея 463
Конфигурация устройства..... 111, 112
Конфигурация 111
Корпуса с защитой от окружающей среды 99
Корпуса 92, 99
Коэффициент Пуассона..... 523
Краткое руководство 41

Л

ЛВС — PakBus 400
ЛВС PakBus..... 400
Лимит диапазона 130
Лимит подачи тока..... 105, 368
Линейный датчик 73
Литиевая батарея..... 45, 474, 603
Логика TTL..... 531
Логика 165
Логический оператор..... 565
Логическое выражение..... 164, 165
Логометрический 341
Локальная переменная 136, 519

М

Маркер 396, 508, 603
Маршрутизатор..... 394, 395, 603
Массив переменных 136
Массив 129, 135, 162,522
Математика 161, 482, 563
Математическая операция 161
Математический оператор..... 563
мВ..... 521
Международная система единиц 528
Меню — пользовательское..... 182, 581
Меню эмулятора терминала..... 502
Меры предосторожности..... 93, 473
Метка времени..... 144, 202, 603
Милли 520
Модем с поддержкой голосовых сообщений 580
Модем/клемма 520
Модификатор переменной 538
Модификатор таблицы данных 540
Модуль расширения входа 85
Модуль терминального ввода 367
Молниеотвод..... 107
Молния..... 43, 94, 105, 514
Мониторинг данных 48, 55
Мониторинг данных 48, 55
Монтаж..... 43, 99
Мост — четвертьмостовый шунт..... 226
Мост сопротивления..... 337
Мост 67, 337, 339
Мостовое измерение 104, 341, 551
Мультиметр 520
Мультиплексоры 366
Мусор 517

Н

Надежное питание..... 100
Накопитель USR 603
Накопитель USR 603
Накопитель..... 370
Написание программы 125
Настройка — PakBus..... 463
Настройка — через CRBasic..... 115
Настройка 462
Настройки — резидентные файлы..... 115
Настройки Ethernet 603
Настройки FTP 603
Настройки HTTP..... 603
Настройки TCP..... 603
Настройки Telnet 603
Научное представление..... 139
Не является числом 482
Непрерывный аналоговый выход..... 367
Несимметричное измерение..... 64, 65, 108, 109, 528
Несколько операторов..... 128
Несколько строк 128
Низкоуровневый переменный ток 352, 367
Номер записи 603
Номинальная мощность..... 85
Нулевой базис..... 210
Нулевой шунт 227
Нуль 227
Нуль-модем 513, 514, 521

О

Обмен Hello-сообщениями..... 517
Обнаружен разомкнутый вход..... 309, 321
Обработка — вектор ветра 296
Обработка — встроенная..... 570
Обработка — выход 145, 542
Обработка — пространственная 571
Обработка вектора ветра..... 297
Обработка..... 563
Обратная полярность..... 47, 103
Обратный вызов..... 290, 392, 509, 518, 597

Обратный порядок байтов 248, 249, 643
Обращение возбуждения 325
Обращение входа 325
Общие сведения — электропитание 494
Общие сведения PakBus 393
Общие сведения 75
Общий режим 305, 308, 309
Общий режим: нуль 308, 309
Объявление — Modbus 413
Объявление — таблица данных 540
Объявление констант 539
Объявление переменной 538
Объявление 129, 140, 537
Ограничитель бросков напряжения 101
Оконечный узел 394, 395
Ом 521
Онлайновая передача данных 522
Оператор AND (И) 200, 565
Оператор OR (ИЛИ) 200, 565
Оператор 563, 565
Операторы — побитовый сдвиг 563
Операторы битового сдвига 200, 563
Операционная система 603
Организация вручную 33
Отладка 485
Отправить(Send) 527
Очистить память 603
Ошибка — аналоговое измерение 108, 109, 482
Ошибка — программирование 481
Ошибка — температура почвы термопара 109
Ошибка — термопара 327, 331, 334, 336
Ошибка нулевого потенциала 109
Ошибка 311, 319, 329, 330, 337, 482, 483, 491
Ошибки времени выполнения 481, 486, 488
Ошибки коммуникационной памяти 491, 603
Ошибки компиляции 481, 486, 488
Ошибки программ 486, 488, 603
Ошибки схемы безопасности 167, 372, 485, 488, 491, 533, 603, 604

П

Пакет с сигнатурой 88
Пакетный режим 229
Память — размер 603
Память — свободная 603
Память окончательных данных 515
Память 87, 162, 370
Параметр 522, 523
Пароль PPP 603
Пароль 92, 603
Перевод байтов 253
Переключаемый порт 12 В пост. тока (SW12) 79, 97, 105, 552, 603
Переключение 531
Перекос временной диаграммы 273, 326, 526
Перем. ток 507
Перем. ток, возбуждение 104, 341
Перем. ток, мощность 556
Перем. ток, подавление шумов 316
Перем. ток, синусоидальная волна 69, 352
Переменная DNP 409
Переменная вне допустимого диапазона 603
Переменная 129, 162, 532
Переменная-триггер 195
Переполнение таблицы 485
Переходный режим 80, 94, 101, 485, 514, 533
Периферийный порт 81
Печатающее устройство 524
Питание 12 В 104
Питание 5 В 104
Питание датчика 85, 103
Питание датчиков 85, 103
Питание от бортовой сети автомобиля 102
Питание 44, 85, 100, 101, 281, 494, 495
Планировщик сети 112
Плата памяти -- 10 30 ..... 89, 143, 205, 386, 461, 509
Платиновый термометр

сопротивления 234, 235, 570
ПЛК 413
ПО для начинающих 48, 50
ПО начального уровня 48, 50
ПО поддержки ПК 95, 512
ПО поддержки регистратора данных 95, 512
ПО поддержки 529
Побитовое сравнение 200
Погрешность успокоения 319
Подавление 50 Гц 100, 316
Подавление 60 Гц 100, 316
Поддерживаемые протоколы 97
Поддержка PDA 654
Поддержка датчика 303
Подключение батареи 47, 103
Подключение к COM-порту 47
Подключение 43, 47, 76
Подпрограмма 147, 288
Поиск и устранение неисправностей — SDI-12 267
Поиск и устранение неисправностей — сеть PakBus 397
Поиск и устранение неисправностей — солнечная панель 496
Поиск и устранение неисправностей — электропитание 494
Поиск и устранение неисправностей 479, 603
Полином — термопара 334
Полный дуплекс 516
Полный мост 337
Полный сброс памяти 603
Полудуплекс 517
Полумост 337
Получение данных 55, 57
Пользовательские функции 602
Пользовательский дисплей 455
Пользовательское меню 84, 581
Поляризованный датчик 341
Полярность 47
Полярный датчик 109
порт CS I/O 81, 511, 633
Порт RS-232 81
Порт SW-12 ..... 79, 97, 105, 552, 603
Порт TCP 603
Порт ввода/вывода 78
Порт управления 78, 603
Порт 78, 462
Порядок следования байтов 248, 249, 643
Последовательность — веб-страница 151
Последовательность — вспомогательная 150
Последовательность — выключение 151
Последовательность — набор 151
Последовательность — разъединение модемного соединения 151
Последовательность автоматической калибровки 153
Последовательность веб-страницы 151
Последовательность выключения 151
Последовательность набора 151
Последовательность разъединения для модема 151
Последовательность 140
Последовательный — ввод/вывод 245, 364, 583
Последовательный — вход 245
Последовательный — датчик 364
Последовательный — номер 603
Последовательный — подключение к порту 47
Последовательный — порт 246, 633
Последовательный — расширение входа 367
Последовательный — режим «анализатор пакетов» 501
Последовательный — режим ретрансляции 501
Последовательный — сервер 295
Последовательный режим 105, 153
Последовательный 62, 527
Пост. ток 513
Пост. ток, возбуждение 104, 341
Посылки и паузы 248
Превышение 485, 603
Предварительно определенная константа 138
Предупреждающее сообщение 486
Преобразование констант 163
Преобразование 163
Преобразователь сигнала 109
Прерывание 78
Пример программы 196, 255, 260
Пример 165, 388, 389, 400
Приоритет задачи 151
Приоритет 116, 151, 156
Проверить интервал 603
Провод 317
Проводимость воды 341
Программа — Modbus 413
Программа — SubScan 156

Программа — арифметика чисел с плавающей запятой ... 161
Программа — библиотека ресурсов ... 210
Программа — большая ... 391
Программа — выполнение ... 151
Программа — выражение ... 160
Программа — документирование ... 126
Программа — единица измерения ... 138
Программа — имя в параметре ... 159
Программа — калибровка в полевых условиях ... 211
Программа — команда DataInterval ... 144
Программа — команда DataTable() ... 143
Программа — команда измерения ... 158
Программа — команда обработки для сохранения данных ... 158
Программа — константа ... 137
Программа — массив ... 135
Программа — математическая операция ... 161
Программа — медленная последовательность ... 155
Программа — обработка выхода ... 145
Программа — объявление ... 129, 140, 537
Программа — ошибки времени выполнения ... 481, 486, 488
Программа — ошибки компиляции ... 481, 486, 488
Программа — переменная ... 129
Программа — подпрограмма ... 147, 288
Программа — последовательный режим 153
Программа — превышение ... 485, 603
Программа — приоритет задачи ... 151
Программа — приоритет сканирования 156
Программа — размерность ... 134
Программа — режим поточной обработки ... 152
Программа — сканирование ... 154
Программа — структура ... 123
Программа — таблица данных ... 140
Программа — тип данных ... 130
Программа — тип параметра ... 159
Программа — учет времени ... 154
Программа CRBasic ... 48, 55, 126
Программа для ПК ... 491
Программа пользователя ... 126, 533
Программа— псевдоним ... 138
Программа ... 86
Программирование — массив с масштабированием ... 177
Программирование — многократные сканирования ... 179
Программирование — скользящее среднее ... 192
Программирование — тестирование насоса для грунтовых вод ... 171
Программирование — условный вывод 170
Программирование — фиксация событий ... 169
Программирование ... 48, 55, 86, 126
Программное обеспечение — для начинающих ... 48, 50
Программное обеспечение ... 95
Продолжение линии ... 128
Прозрачный режим — SDI-12 ... 501
Производительность канала ... 398
Промежуточная память ... 145
Промежуточное хранение ... 512
Пропуск ... 144
Пропускная способность ... 531
Пропущенное медленное сканирование ... 603
Пропущенное сканирование системы ... 603
Пропущенное сканирование ... 144, 485, 528, 603
Пропущенные записи ... 603
Просмотр данных ... 48, 55
Просмотр файла ... 459
Пространственная обработка ... 571
Процессор ... 97
Прямой порядок байтов ... 248, 249, 643
Прямой термоэлектрический эффект ... 527
Прямоугольный сигнал ... 69
Пьезометр ... 42

P

Размер буфера ... 249
Размер записи данных ... 603
Размер пакета ... 603
Размерность ... 134, 513
Разомкнутые входы ... 309
Разрешение — концепция ... 533
Разрешение — определение ... 130, 525, 533
Разрешение — разрешающая способность по времени ... 62
Разрешение — среднее за период ... 62
Разрешение — термопара ... 331

Разрешение — тип данных 130, 525, 533
Разрешение по частоте 353
Разрешение 97
Разъем питания в автомобиле 102
Разъемы с 9 контактами 246, 633
Разъемы с 9 штекерами 246, 633
Расчет деформации 342
Расширение аналогового входа 97, 366
Расширение импульсного входа 367
Расширение управляющего выхода 368
Расширенная команда SDI-12 278
Расширенные команды — SDI-12 278
Регистр временного хранения 413
Регистры в обратном формате – Modbus 415
Регулятор 525
Редактировать программу 459
Редактировать файл 459
Редактор — ярлык 125
Редактор 50
Редактор CRBasic 125
Редактор программ 50
Режим поточной обработки 105, 152, 153
Резервирование батареи 45, 94
Резервная батарея 45, 94, 474
Резистивный мост 97, 337
Резистор 525
Результаты компиляции 603
Реле 369, 370
Руководство для начинающих 41

C

С защитой от атмосферных воздействий 93, 533
Сбор данных 55, 57
Сбор данных 55, 57
Сбор данных 88, 391
Сброс памяти 381
Сброс числа импульсов 177
Сброс 381, 603
Свободная коммуникационная память 491, 603
Свободное место на накопителе USR 603
Свободное место на накопителе ЦП 603
Связь 47, 55, 90, 391, 408, 490, 491
Сдвиг мирового координированного времени 603
Семафор 527
Сервер 655
Сеть беспроводных датчиков 558
Сеть 394
Сжатие Gzip 117
Сжатие файла 117, 463
Сжатие 117, 463
СИ, международная система единиц 528
Сигнатура OS 603
Сигнатура программы 603
Сигнатуры — время выполнения 603
Сигнатуры — программа 178, 603
Сигнатуры — система 169
Сигнатуры времени выполнения 603
Символ NULL 264
Символ завершения 264
Синусоидальная волна 352
Синхронизация часов 55
Синхронный 530
Система сбора данных — датчик 42
Системное время 154, 530
Сканирование (интервал выполнения) 97, 526
Сканирование 97, 154
Скользящее среднее 192
Скорость передачи в бодах 248, 249, 583, 588, 599, 603
Скрытые файлы 92
Смещение опорного заземления 326
Совместимость с требованиями CE 97
Сокращения 168
Солнечная панель 496
Сообщения 603
Сопротивление 525
Сосед 395, 603
Соседнее устройство 521
Соседний фильтр 396
Состояние платы 603
Состояние 462
Состояние 79, 528, 634
Состояния электропитания RS-232 634
Сохранение данных 385
Сохранение памяти 126, 144, 162, 253
Сохранить данные 127, 385
Сохранить настройки 603
Спутник 599
Среднее за период 62, 97, 360, 361, 523, 554
Среднеквадратичное значение 526
Стандартное отклонение 300
Стартовый бит 249
Стоповые биты 249
Сторожевой таймер 533

Строка набора 603
Строковая операция 282, 574
Строковая функция 574
Строковое выражение 166
Структура — программа 123
Струнный входной модуль 367
Суммарное испарение 544
Схема ИЛИ, диодная цепь 102
Схема многоуровневого подчинения 399
Счетчик низкого напряжения 12 В 603
Счетчик цикла 519

Т

Таблица — заголовок данных 165
Таблица ввода параметров 4 звезды (*4) 524
Таблица данных 48, 140, 141,142, 167, 197, 457,466, 540
Таблица состояния 604
Таблица 48
Таблицы окончательных данных 457
Телекоммуникации 48, 55, 89, 90, 391, 408
Температура панели 328, 329, 330, 335, 337, 603
Термистор 235, 305, 530, 556
Термопара 46, 329, 330, 331, 334, 336, 337
Технические единицы измерения 514
Технические характеристики 97
Техническое обслуживание 93, 473
Тип данных — Bool8 198
Тип данных — LONG 519
Тип данных — NSEC 202, 521
Тип данных — UINT2 531
Тип данных Bool8 ..... 198, 200
Тип данных BOOLEAN (логический) 130, 508
Тип данных NSEC 130, 202, 521
Тип данных 130, 131, 162,200
Тип данных: формат 249
Тип параметра 159
Ток 103
Точка данных 512
Точность часов 97
Точность 523, 533
Точность 97, 337, 507, 533,

См. Подавление 50 Гц

Трансформатор 85, 498
Требование — электропитание 101
Триггер — выход 195
Триггеры 195
Тригонометрические функции 568

У

Узел 394, 521, 603
Управление модемом 597
Управление переменными 589
Управление таблицы данных 592
Управление файлами 382, 589
Управление файлами 603
Управление файлом 382, 515
Управление 79, 105, 367, 545, 548
Условная компиляция 180, 181
Установка 43
Учебное пособие 41
Учебное упражнение 46

Ф

Файлы CAL 210
Фиксированный диапазон напряжений . 309
Фильтр маршрута 603
Флажок 135, 414
Фоновая калибровка 153, 323, 326, 344, 603
Формат — числовой 139
Формат данных 90, 641
Фрагментация 373
Функция часов 578

Х

Хранение данных — триггер 196
Хранение данных 87, 144, 370,541, 542
Хранение 541

Ц

Целое 163, 518
Цепь зарядки 498, 499
Цепь нормирования 361
Цепь 339, 361, 370
Цикл 519
Цифровой ввод/вывод 62, 73, 78,

97, 368,
554
Цифровой регистр 413
ЦП 374, 511

Ч

Часовой пояс 202
Частота выборки 526
Частота 69, 349
Часы 75
Четвертьмостовый нуль 227
Четвертьмостовый шунт 226
Четвертьмостовый 223, 337
Число импульсов 97, 349
Число с плавающей точкой 161
Число свободных байтов на плате 603
Числовой формат 139

Ш

Шестнадцатеричный 139
Широтно-импульсная модуляция 79, 368
Шифрование 92, 406
Штекер 5 В 633
Шум 100, 311,
315, 316,
317
Шунт на базе RC-резистора 225

Э

ЭДС 311
Электронная почта 289, 593
Электропитание 48, 80,
97,
102, 103,
105
Электропроводка 43, 47,
76,
364
Эмулятор терминала 501
Энергопотребление 101
Эталонная температура 328, 329,
330, 335,
336, 337
Эталонная температура 328, 329,
330, 335,
336, 337
Эталонное напряжение 108
Эталонный контакт 336

Компании группы Campbell Scientific

Campbell Scientific, Inc. (CSI)
815 Уэст 1800 Норт
Юта, г. Логан, 84321
США
www.campbellsci.com • info@campbellsci.com

Campbell Scientific Africa Pty. Ltd. (CSAf)
п/я 2450
Somerset West 7129
ЮАР
www.csafrica.co.za • cleroux@csafrica.co.za

Campbell Scientific Australia Pty. Ltd. (CSA)
п/я 8108
Garbutt Post Shop QLD 4814
АВСТРАЛИЯ
www.campbellsci.com.au • info@campbellsci.com.au

Campbell Scientific (Beijing) Co., Ltd.
8B16, Floor 8 Tower B, Hanwei Plaza
7 Guanghua Road
Chaoyang, Beijing 100004
КИТАЙСКАЯ НАРОДНАЯ РЕСПУБЛИКА
www.campbellsci.com • info@campbellsci.com.cn

Campbell Scientific do Brasil Ltda. (CSB)
Rua Apinagés, nbr. 2018 — Perdizes
CEP: 01258-00 — Sao Paulo — SP
БРАЗИЛИЯ
www.campbellsci.com.br • vendas@campbellsci.com.br

Campbell Scientific Canada Corp. (CSC)
14532 - 131 Avenue NW
Edmonton AB T5L 4X4
КАНАДА
www.campbellsci.ca • dataloggers@campbellsci.ca

Campbell Scientific Centro Caribe S.A. (CSCC)
300 N Cementerio, Edificio Breller
Santo Domingo, Heredia 40305
КОСТА-РИКА
www.campbellsci.cc • info@campbellsci.cc

Campbell Scientific Ltd. (CSL)
Campbell Park
80 Hathern Road
Shepshed, Loughborough LE12 9GX
ВЕЛИКОБРИТАНИЯ
www.campbellsci.co.uk • sales@campbellsci.co.uk

Campbell Scientific Ltd. (CSL France)
3 Avenue de la Division Leclerc
92160 ANTONY
ФРАНЦИЯ
www.campbellsci.fr • info@campbellsci.fr

Campbell Scientific Ltd. (CSL Germany)
Fahrenheitstraße 13
28359 Bremen
ГЕРМАНИЯ
www.campbellsci.de • info@campbellsci.de

Campbell Scientific Spain, S. L. (CSL Spain)
Avda. Pompeu Fabra 7-9, local 1
08024 Barcelona
ИСПАНИЯ
www.campbellsci.es • info@campbellsci.es

Чтобы получить контактную информацию ближайшего к Вам локального или международного представителя Campbell Scientific, пройдите по ссылке www.campbell