ГОСУДАРСТВЕННЫЙ НАУЧНЫЙ ЦЕНТР РОССИЙСКОЙ ФЕДЕРАЦИИ

ИНСТИТУТ ФИЗИКИ ВЫСОКИХ ЭНЕРГИЙ

ИФВЭ 2005–13
ОАФ

В.Д. Матвеев, В.П. Новиков

Коммуникационная система для централизованной архитектуры сбора данных и управления по шине CAN-bus

Протвино 2005

УДК 621.39 М-24

Аннотация

Матвеев В.Д., Новиков В.П. Коммуникационная система для централизованной архитектуры сбора данных и управления по шине CAN-bus. Препринт ИФВЭ 2005–13. – Протвино, 2005. – 9 с., 1 рис., 1 табл., библиогр.: 8.

Концептуально описывается упрощенная CANopen коммуникационная система, оптимизированная для централизованного управления сетью на базе шины CAN-bus. Описаны отличительные особенности данной коммуникационной системы и ее реализация в архитектуре унифицированного CAN Slave модуля (CAN-NODE). Не обеспечивая полной совместимости с CANopen спецификациями, предложенная концепция эффективно реализуется в CAN-NODE на низкопроизводительных микропроцессорах, обеспечивая высокие скорости передачи и обработки данных. Работа выполнена в рамках участия ИФВЭ в сотрудничестве LHCb (CERN).

Abstract

Matveev V.D., Novikov V.P. Communication System for the Centralized Architecture of Control and Data acQuisition Based on CAN-bus: IHEP Preprint 2005–13. – Protvino, 2005. – p. 9, fig. 1, tables 1, refs.: 8.

In work the simplified CANopen communication system optimized for centralized management by a network based on CAN-bus is conceptually described. Distinctive features of the communication system and its realization in unified CAN Slave module (CAN-NODE) are described. Not providing complete compatibility with CANopen specifications , the offered concept is effectively realized in CAN-NODE on low-productive microprocessors, providing high speeds of transfer and data processing. Work is executed within the framework of membership IHEP in LHCb collaboration (CERN).

Введение

В настоящее время достаточно широкое распространение в практике реализации небольших локальных высоконадежных систем сбора данных и управления получили системы использующие шину CAN-bus [1] и различные развитые системы управления коммуникациями по стандартной открытой семиуровневой модели. Одной из таких систем является система CANopen [2], поддерживаемая и развиваемая международной организацией CiA (CAN in Automation) [3]. Рассчитанная, как и другие подобные системы DeviceNet и CAN Kingdom [4, 5], на невысокую загрузку шины и, как правило, на не очень высокие требования к обработке и генерации данных и управления в режиме реального времени, система CANopen однако требует для реализации в каждом модуле на шине CAN-bus значительных программно-аппаратных ресурсов для обслуживания специфицированных протоколов. При этом в небольшой централизованной системе управления и сбора данных, в которой все модули на шине CAN-bus взаимодействуют только с одним мастером – мастером сети, значительная часть возможностей CANopen остается невостребованной, а управление большим потоком данных через CAN-bus в рамках специфицированных протоколов невозможно.

В настоящей работе предлагается CANopen ориентированная, но упрощенная и измененная концепция управления коммуникациями в централизованной сети, дополненная механизмом простого управления сетевым временем и временного маркирования данных, набором мод интерпретации данных, передаваемых в PDO объектах, и микропрограммным управлением (в CAN-SLAVE модуле сети – CAN-NODE) быстрым блочным обменом. Работа выполнена в рамках участия ИФВЭ в сотрудничестве LHCb (CERN).

1. Основные используемые положения CANopen спецификаций

1.0 Используются рекомендованные CANopen спецификации:

- на используемый CAN-bus разъем (9pin Dsub : DIN 41652);
- на используемые скорости передачи на CAN-bus и программирование битовой синхронизации на CAN-bus [1] (bit timing спецификация);
- на используемый CAN-bus транссивер (согласно ISO 11898).

1.1 Реализуется усеченное подмножество всех предусмотренных CANopen типов сервисов и соответствующих им протоколов, коммуникационных объектов и CAN-bus сообщений:

- протоколы сетевого управления (NMT– Network Management Protocols);
- протоколы для сервисных объектов (SDO – Service Data object Protocols);
- протоколы для процессных объектов (PDO – Process Data object Protocols);
- протоколы для реализации специальных функций (объектов) (Special object Protocols).

1.2 В CAN-NODE реализуются все 4 состояния, предусмотренные для CANopen устройства (инициализация – Initialization, предрабочее – Pre-Operational, рабочее – Operational и подготовительное – Prepared), и переходы между ними.

1.3 Используется предусмотренное CANopen распределение идентификаторов CAN-bus сообщений с выделением 4-битного функционального кода, приписанного определенному типу объекта или функции передаваемой в сообщении, и 7-битного идентификатора CAN модуля – адресата сообщения.

1.4 Описание CAN-NODE реализуется в форме, совместимой с описанием CANopen устройства, т.е. через CANopen словарь объектов (object dictionary). Все непредусмотренные CANopen объекты специфицируются в определяемом производителем специальном профиле (Manufacture Specific Profile – MSP), занимающем область индексов, начиная с 2000h в словаре объектов.

2. Ограничения в реализации CANopen спецификаций и протоколов

2.1 Введено ограничение на используемые типы данных оконечных устройств – только байтовый формат и переменные типа Unsigned8 и Octet String (массив Unsigned8).

2.2 Из определенных в CANopen SDO протоколов реализуется только один протокол – протокол инициации сегментной передачи данных (Initiate Domain Down/Upload protocol), предусматривающий так называемую завершенную (expedited) передачу до 4 байтов данных, включенных в SDO сообщение. С ориентацией на полную централизацию потоков данных в сети (клиентом всегда выступает единственный мастер сети), в каждом CAN-NODE реализуется только один SDO объект (два CAN data frame с отличающимися на единицу функциональными кодами в идентификаторах). Протокол обеспечивает запись и чтение по всем специфицированным входам словаря объектов в CAN-NODE.

2.3 В рамках PDO протокола в каждом CAN-NODE реализуется один PDO объект (PDO-0) асинхронного типа (код типа FFh) с фиксированным нулевым временем ожидания (inhibit-time). Объект реализуется в виде двух PDO сообщений (data frame) – T_PDO (transmit-PDO) и R_PDO (receive PDO), имеющих отличающиеся на единицу функциональные коды в индентификаторах. Определенная в CANopen процедура мультиплексирования данных в PDO объектах (PDO mapping) не используется. Определенное в CANopen PDO протоколе использование PDO remote frame как сигнала запроса данных дополнено возможностью его использования как сигнала подтверждения приема данных или сигнала окончания блочной передачи данных.

2.4 Из предусмотренных в CANopen протоколов специальных функций реализуется только протокол формирования и передачи в сети аварийных сообщений (Emergency protocol). Протоколы установки сетевого времени (Time Stamp protocol) и синхронизации данных (Synchronization (SYNC) protocol) заменены упрощенными процедурами управления сетевым временем и приписки временных меток данным, поступающим от CAN-NODE к мастеру сети.

2.5 Для моделирования работы с CAN-NODE как с полноценным CANopen устройством предусмотрено разбиение словаря объектов для каждого CAN-NODE на часть, хранимую в

мониторе сети (стандартный коммуникационный профиль устройства), и часть, хранимую в CAN-NODE, содержащую специальный профиль устройства (MSP) и дубликаты части стандартного коммуникационного профиля (его изменяемых записей). При этом постоянное соответствие дубликатов частей коммуникационного профиля должно обеспечиваться монитором сети. Для обеспечения компактности и быстрого доступа к записям части словаря, хранимой в CAN-NODE, все записи в ней имеют байтовый формат, а записи коммуникационного профиля – сжатую форму хранения. При дублировании записей монитором сети должна использоваться специальная таблица трансляции записей.

Примечание: в условиях отсутствия требования работы с CAN-NODE по модели CANopen устройства, словарь объектов устройства может ограничиваться только частью, хранимой в самом CAN-NODE.

3. Архитектура унифицированного CAN-NODE

Разработанная архитектура унифицированного CAN-NODE (рис. 1) включает:

- CAN-bus транссивер PCA 82C250 фирмы Philips, удовлетворяющий спецификации ISO11898 ;

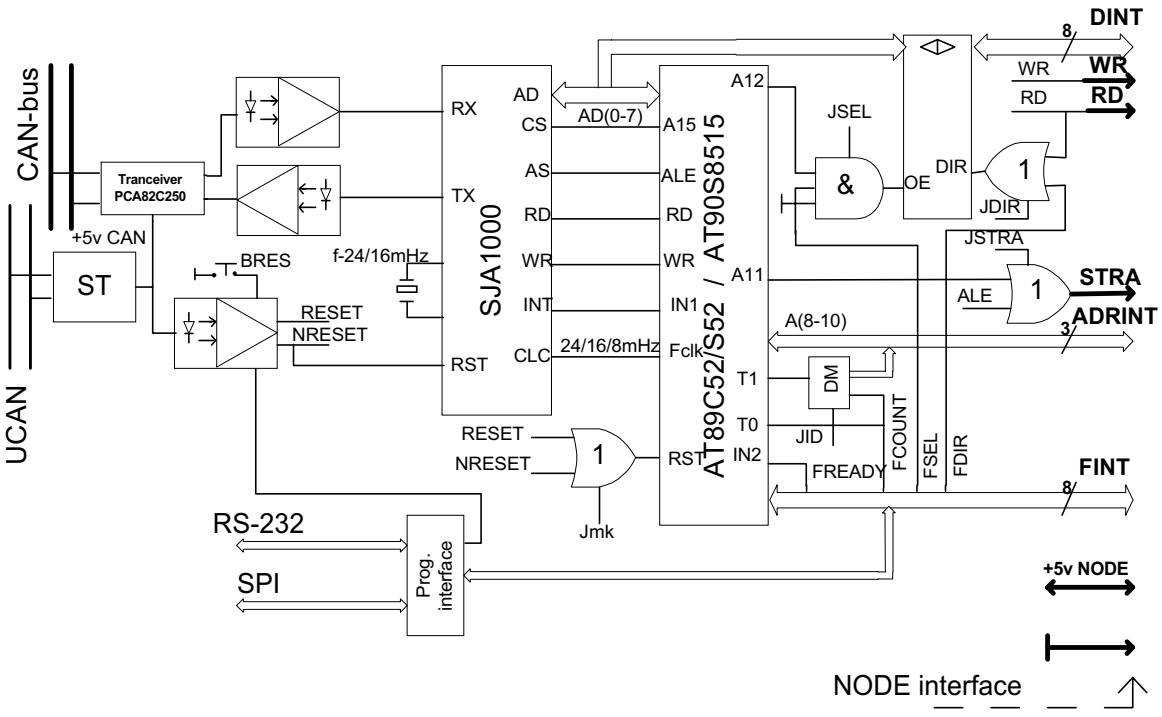

Рис. 1. Схема унифицированного CAN-NODE.

- блок стабилизированного питания транссивера от питания CAN-bus (ST);
- блок аппаратного сброса CAN-NODE по включению питания CAN-bus;
- программируемый CAN-bus контроллер SJA1000 фирмы Philips;
  оптронную развязку CAN-bus контроллера от CAN-bus;
- микроконтроллер Atmel архитектуры C51 AT89C51(52) или AVR AT90S8515 в качестве контроллера коммуникационных протоколов и контроллера интерфейсов;
- интерфейс CAN-NODE с оконечными устройствами через три шины (8-разрядную шину функций – FINT, 8-разрядную шину данных – DINT и 3-разрядную адресную

шину – ADRINT); эти интерфейсные шины ассоциированы с соответствующими тремя объектами в специальном профиле CAN-NODE (FINT, DINT, ADRINT).

4. Специальный профиль (MPS) унифицированного CAN-NODE (табл. 1)

Таблица 1. Специальный профиль CAN-NODE

Индекс (h)	**Имя** переменной (объекта)	**Тип** пере-менной	**Доступ** Раб/Пре сост.	**Дефолт** значе-ние (h)	**Тип** объекта	**Примечание**
2000	FINT	Unsign8	WR/RD	FF	Port MC	Рег. функции интерфейса с устройств.
2001	ADRINT	---“----	WR/RD	80	Port MC	Рег. адреса интерфейса с устройств.
2002	DINT	---“----	WR/RD	--	Port MC	Рег. данных интерфейса с устройств.
2003	ACC_MASK	---“-----	RD/RD	FF	RAM	Acceptance Mask CAN
2004	EVENT_PDO	---“-----	WR/WR	80	---“--	Спецификатор режима PDO
2005	ERREG	---“----	RD/RD	00	---“--	Регистр оши-бок модуля
2006	STATE_NODE	---“----	RD/RD	41	---“--	Рег. состояния CAN модуля
2007	MODE_NODE	---“----	WR/WR	00	---“--	Моды работы
2008	COUNT_D0	---“----	WR/WR	00	---“--	Счетчик данн.
2009	COUNT_D1	---“----	WR/WR	00	---“--	Блочных PDO
200A	ACC_CODE	---“----	RD/RD	00	RAM	Acceptance Code CAN
200B	ACC_PDO_T	---“----	WR/RD	C0	RAM	Код доступа для PDO_T
200C	TRANS_PDO	---“----	RD/RD	FFconst	ROM	Transmission Type for PDO
200D	ACC_PDO_R	---“----	WR/RD	C0	RAM	Код доступа для PDO_R
200E	ID	---“----	RD/RD	Const	RAM	ID модуля
200F	NRAM0	---“----	RD/RD	FFconst	ROM	Спецификатор емкости
2010	NRAM1	---“----	RD/RD	00const	ROM	RAM оконечн. устройства
2011	SERNUM	--“-----	RD/RD	Const	ROM	Serial number of module
2012	TIME	--“-----	WR/WR	00	RAM	
2013	DTIME	---“----	WR/WR	FF	RAM	Time-out mod. Net Time Master

Все объекты специального профиля имеют байтовый формат (переменные типа Unsigned8). Как и предусмотрено CANopen, доступ к объектам профиля разрешен только в предрабочем и рабочем состоянии CAN-NODE. Типы доступа не могут быть изменены. Дефолт статусы объектов (дефолт значения переменных) заносятся при инициализации CAN-NODE.

Объект ID хранит идентификатор CAN-NODE в сети. Он определяется как логическая сумма зашитого 7-битного базового значения и установленного джамперами 3-битного кода. Изменение ID на джамперах вступает в силу только при инициализации CAN-NODE.

Зашитый статус объекта SERNUM определяет аппаратную и программную версии CAN-NODE, его некоторые характеристики и связь с типом CAN модуля. По статусам SERNUM CAN модулей монитор сети определяет текущую конфигурацию и свойства сети.

Объект STATE_NODE хранит текущее состояние CAN-NODE, которое через этот объект можно опросить вне CANopen сторожевого протокола (Node Guarding Protocol) без изменения бита чередования (toggle bit). Кроме состояния CAN-NODE, предусмотренного CANopen, в STATE_NODE хранятся дополнительный бит состояния CAN-NODE (модуль – мастер сетевого времени или нет) и зашитые в CAN-NODE режимы работы CAN контроллера SJA1000 (BasiCAN или PeliCAN [6], разрешенные типы прерываний от SJA1000).

В объекте MODE_NODE устанавливаются (записываются) режимы обработки внешних прерываний в микроконтроллере CAN-NODE (прерывания от CAN контроллера и внешних устройств, связанных с интерфейсом CAN-NODE), режим обработки ошибок, зарегистрированных CAN контроллером SJA1000, текущая мода работы CAN-NODE и бит подтверждения окончания процедур в передаче данных. Мода работы CAN-NODE определяет интерфейс между CAN-NODE и оконечным устройством (устройствами) и интерпретацию данных в PDO сообщениях. На текущий момент реализованы и используются 7 мод, 5 из которых универсальные, а 2 – специализированные для работы с конкретными устройствами. Универсальные моды, определяющие различные типы унифицированных интерфейсов, описаны в разделе 5.

Моды работы CAN-NODE введены как замена громоздкого (в CANopen) механизма работы с прикладными профилями устройств, в том числе и стандартизованными.

В объекте EVENT_PDO устанавливаются режимы обработки (передачи, формирования и синхронизации) данных в CAN-NODE, принимаемых и передаваемых в PDO сообщениях.

Выделяются три режима обработки PDO сообщений в CAN-NODE:

- SLOW – медленный (“0” бита FAST_PDO в объекте EVENT_PDO), в котором информация каждого PDO сообщения обрабатывается или формируется как законченный блок данных (независимый от других PDO до и после данного) с буферизацией данных PDO, возможностью их синхронизации по внешним RTR запросами (remote frame) (разрешение такой синхронизации – бит RTR в объекте EVENT_PDO) и возможностью работы CAN-NODE с данными в режиме приписки или анализа служебной информации.
- FAST – быстрый (“1” бита FAST_PDO в EVEN_PDO), в котором единственной возможной синхронизацией данных является флаг готовности устройства их принимать или выдавать (для этого флага выделен фиксированный старший бит на шине FINT интерфейса – бит READY); разрешение такой синхронизации – бит ZIKL_READY в EVENT_PDO; входящие и выходящие PDO сообщения обрабатываются максимально быстро и без буферизации в CAN- NODE передаются между CAN контроллером и интерфейсом с устройством; приписка или анализ служебной информации в данных невозможны. Этот режим может использоваться для записи или чтения в память с последовательной адресацией, а для аппаратного ограничения на объем таких операций может использоваться зашитый в объектах NRAM0 и NRAM1 максимальный размер RAM.

- MIDDLE – промежуточный, в котором реализуется обработка данных, входящих или выходящих PDO как образующих блочную структуру (“1” бита END_COUNT в EVENT_PDO). Длина блока определяется значениями, записанными в объекты COUNT_D0 и COUNT_D1 (COUNT_D1 – продолжение COUNT_D0); синхронизация данных может осуществляться, в зависимости от состояния бита FAST_PDO в EVENT_PDO, или по внешним RTR запросам (как в режиме SLOW), или по флагу готовности устройства их принимать или выдавать (как в режиме FAST). Данные буферизуются в CAN-NODE с оптимизацией по времени; приписка или анализ служебной информации в данных может осуществляться только на уровне заголовка блока передачи.

Объекты FINT, DINT, ADRINT прямо ассоциированы с шинами интерфейса CAN-NODE с оконечным устройством (устройствами). Через доступ к этим объектам может реализовываться управление устройствами и медленный байтовый обмен данными.

Объекты ACC_MASK и ACC_CODE доступны только для чтения и используются для контроля установленного (в CAN контроллере SJA1000) аппаратного фильтра CAN-bus сообщений. Изменение фильтра могут происходить только при изменении состояния CAN-NODE.

Объекты TIME и DTIME хранят текущее относительное время CAN-NODE, которое может синхронизироваться с сетевым временем по специальному упрощенному протоколу управления сетевым временем (TIME_STAMP протокол).

В объект ACC_PDO_R записываются биты разрешения или запрещения приема CAN_NODE R_PDO и R_PDO_RTR (remote frame) сообщений с CAN-bus.

В объект ACC_PDO_T записываются аналогичные биты, разрешающие или запрещающие генерацию CAN-NODE T_PDO и T_PDP_RTR сообщений в CAN-bus. В этом же объекте записывается бит разрешения включения счета времени в CAN_NODE (TIMER_EN) и бит разрешения приписывания текущего времени CAN-NODE как служебной информации к данным, передаваемым в T_PDO сообщениях (TIME_EN).

Объект TRANS_PDO хранит зашитый в CAN-NODE разрешенный тип синхронизации для PDO сообщений, в соответствии со спецификацией CANopen. В реализованных версиях CAN-NODE используется только асинхронный тип PDO (код FFh), т.е. независимый во времени опрос данных в различных модулях сети без синхронизации с помощью синхронизирующих сообщений-объектов (Synchonnization Object), предусмотренных CANopen протоколом синхронизации (Sync Protocol).

Объект ERREG хранит флаги ошибок, зафиксированных в работе CAN-NODE, по которым в соответствии с CANopen Emergency Protocol формируются и передаются в сеть аварийные сообщения.

Объекты SERNUM, ID, ACC_PDO_R, ACC_PDO_T и ERREG являются дубликатами изменяемых частей соответствующих объектов стандартного CANopen коомуникационного профиля устройства, как указывалось в разделе 2.5.

5. Универсальные моды реализации интерфейсов в CAN-NODE

Разработано 5 универсальных мод, ориентированных на различные группы устройств:

- Прозрачная (TRANSPARENT) мода – ориентирована на быстрые устройства типа RAM с безадресным обращением и параллельной байтовой записью и чтением по соответствующим циклам обращения микроконтроллера CAN-NODE к внешней памяти; данные передаются только через шину DINT интерфейса; никакого кодирования, контроля и преобразования данных нет; прозрачная мода допускает все режимы обработки PDO сообщений (SLOW, FAST, MIDDLE).

- Открытая (OPEN) мода – ориентирована на медленные устройства, управление которыми осуществляется кодированием состояний на шине функций (FINT) и передачей данных или через биты шины FINT, или через шину данных (DINT) в режиме записи/чтения портов микроконтроллера. Работа в этой моде выглядит как посылка в CAN-NODE и прием от него в ответ идентичных по формату PDO сообщений, обрабатываемых в режиме SLOW; первый байт всегда служебный, не передаваемый устройству на интерфейсе; при передаче в CAN-NODE в служебном байте записывается бит режима управления (Simple Control – да/нет) и 7-битный код временной паузы (Time-out) (формируемой в CAN-NODE). Квант Time-out зависит от типа используемого микроконтроллера и имеет порядок долей миллисекунды. В режиме управления Simple Control байты данных записываются (передаются) последовательно в одну и ту же шину FINT с паузами, равными Time-out. Все 8 бит шины FINT могут использоваться для свободного кодирования управления и данных. В режиме NO Simple Control байты начиная со второго записываются в шину FINT или DINT по правилу: первый – всегда в FINT, байт, записываемый в FINT, определяет по служебному биту (флагу FADR) следующего получателя (FINT или DINT), байт, следующий за записываемым в DINT, всегда посылается в FINT; другой служебный бит в байте, посылаемом в FINT, – флаг ожидания (FWAIT) (да/нет) сигнала готовности от устройства в уже известном фиксированном бите READY; на ожидание накладывается заданный в первом служебном байте Time-out, по истечении которого протокол передачи прерывается и в CAN-NODE обрабатывается ситуация ошибки протокола. В режиме NO Simple Control свободное кодирование на шине FINT имеет поле 5 бит, а на шине DINT – все 8 бит; реализация алгоритма управления устройством в OPEN моде целиком ложится на прикладную программу.
- Закрытая мода микропрограммной реализации Hand Shake интерфейса между микроконтроллером CAN-NODE и микроконтроллером исполнительного устройства (CLOSE HAND SHAKE). Эта мода реализует асинхронный байтовый обмен данными между двумя микроконтроллерами в обоих направлениях блоками до 7 байт в режиме Master-Slave; микроконтроллер, передающий данные, всегда Master, а принимающий данные – всегда Slave. Для интерфейса используются 5 протокольных линий на шине FINT и 8-разрядная шина данных DINT; ситуации столкновения встречных запросов на передачу разрешаются в микроконтроллере CAN-NODE через протокол отложенных передач; протокол интерфейса охвачен фиксированным аппаратным Time-out на все Hand Shake сигналы; истечение Time-out прерывает протокол и в CAN-NODE обрабатывается ситуация ошибки протокола.

  PDO сообщения, поступающие в и из CAN-NODE, обрабатываются в режиме SLOW.
- Закрытая мода микропрограммной реализации записи и чтения по стандартной двухпроводной шине I2C (CLOSE I2C); для интерфейса с транссиверами шины I2C используются 3 протокольные линии на шине FINT.

  Реализуются 4 процедуры обмена данными по шине I2C, которые задаются в первом служебном байте R_PDO сообщения:
  - I2C_SEND – реализует захват шины, соединение с прибором на шине, имеющим заданный в процедуре адрес, и посылку в прибор N байт данных (0–6), зависит от количества байтов данных в принимаемом R_PDO сообщении, задающем данную процедуру;
  - I2C_RECEIVE – реализует все, как в процедуре I2C_SEND, и дополнительно чтение из прибора M байт данных (0–7), M задается в первом служебном байте вместе с типом процедуры;
  - I2C_WR – реализует запись в выбранный ранее прибор до 7 байтов данных из принятого R_PDO сообщения, задающего данную процедуру;

- I2C_RD – реализует чтение из ранее выбранного прибора до 7 байтов; количество байтов M задается как в процедуре I2C_RECEIVE.

Примечание: для CAN-NODE на микроконтроллере AVR AT90S8515 реализованы две моды спецификации I2C-BUS [7] – стандартная (до 100 кбит/с) и быстрая (до 400 кбит/с), а на более медленных микроконтроллерах AT89C52 (S52) – только стандартная мода.

- Закрытая мода микропрограммной реализации записи и чтения по трехпроводной шине (CLOSE 3-WIRE) (SELi , SCLK и SDA); количество сигналов позиционной выборки устройств на шине (SELi) – до 8; для интерфейса со схемой управления шиной 3-WIRE используются шина ADRINT (3 линии) и 3 линии шины FINT.

  Предусмотрены следующие операции:

  - SERIAL_WR – запись заданного количества байтов данных (0–6 байтов) в заданное устройство на шине;
  - SERIAL_RECIEVE – запись заданного количества байтов данных (0–6 байтов) в заданное устройство и чтение (0–7 байтов) данных из устройства;
  - SERIAL_RD – чтение 0–7 байтов данных из заданного устройства.

Примечание: сигнал выборки устройства может программироваться как перекрывающий все время операции (FULL-SEL), или как фиксированный одиночный импульс (длительность 5 мкс) выборки-запуска до или после операции, в зависимости от типа операции; предусмотрено программирование возможности приема внутри протокола записи сигнала подтверждения записи от устройства.

Обе закрытые моды обмена по шине I2C и по шине 3-WIRE используют обработку PDO сообщений в режиме SLOW и имеют схожие форматы управления.

6. Управление сетевым временем

В отличие от CANopen Time Stamp процедура выполняет не запись текущего физического времени в CAN-NODE (8 или 6 байт), а просто сбрасывает внутренние счетчики относительного времени в каждом CAN-NODE , в котором в данный момент разрешена работа с сетевым временем. Зная таймерные характеристики всех модулей в сети (они отражаются в их серийных номерах), мастер сети назначает мастером сетевого времени (Time Master сети) (генератором синхронизирующих TIME-STAMP сообщений) один из модулей в сети, записывает ему в объект DTIME квант времени между Time Stamp процедурами (0,04 – 10 с) и выполняет первичную синхронизацию сетевого времени посылкой своего TIME-STAMP сообщения. Далее мастер сети корректирует абсолютное сетевое время принимая поступающие из сети TIME-STAMP сообщения, генерируемые мастером сетевого времени, а CAN модули сети отсчитывают свое синхронизированное относительное время, которое они могут приписывать к данным в соответствии с протоколом приписки сетевого времени. Преобразование относительного времени событий в CAN модулях в абсолютное сетевое время осуществляется на уровне монитора сети или прикладной программы.

CAN модуль, поддерживающий работу с сетевым временем и функцию Time Master сети, имеет три флага управления (см. также раздел 4):

- флаг TIMER_EN – разрешающий работу встроенного аппаратно-программного таймера от момента синхронизации сетевого времени;
- флаг TIME_EN – разрешающий включение показаний сетевого времени (два байта) в сообщения с состоянием модуля (NMT STATE) и в сообщения с данными от модуля (T_PDO);

- флаг TIME_STAMP – разрешающий генерацию модулем TIME-STAMP сообщений (собственно функция Time Master сети), если включен встроенный аппаратно-программный таймер.

Описанная процедура работы с сетевым временем значительно разгружает CAN-NODE от программного контроля за временем, сохраняя высокую точность привязки событий к единому сетевому времени.

Заключение

Описанная в данной работе концепция построения коммуникаций для централизованной системы управления и обработки данных, использующей шину CAN-bus, успешно реализована и прошла испытания в стендах цезиевой калибровки модулей адронного калориметра (HCAL) [8] для установки LHCb (CERN). Для всех CAN модулей, разработанных для этих стендов, написанные на языке С микропрограммы CAN-NODE со всеми необходимыми модами работы не превысили 4 Кбайт откомпилированного микрокода.

Подготовка и отладка микропрограмм проводились на системе Proview32 V.3.34 (Franklin Software Inc.) для микроконтроллеров с архитектурой C51 и на системе IAR Embedded Workbench V.2.31C (IAR Systems Inc.) для микроконтроллеров с AVR архитектурой.

Для CAN-NODE на сравнительно низкопроизводительном микроконтроллере AT89C51 максимальная скорость обработки и передачи данных в PDO сообщениях, измеренная на тестовых программах, составила 40 Кбайт/с. для режима FAST, 30 Кбайт/с. для режима MIDDLE и 5 Кбайт для режима SLOW обработки PDO сообщений.

Список литературы

[1] CAN Specification Version 2.0. 1991, Robert Bosch GmbH ,Postfach 50, D-7000 Stuttgart 1.

[2] CiA DS-301,V3.0 CANopen Communication Profile. October 1996.

[3] Www.can-cia.org

[4] Www.odva.org

[5] Www.canKingdom.org

[6] SJA1000 Stand-alone CAN controller . Product specification 2000 Jan.04. Philips Semicondu ctors.

[7] The I2C-BUS Specification Version 2.1. Jan.2000, Philips Semiconductors.

[8] Design and integration of HV, LED monitoring and radioactive -source system for HCAL. Y.Gouz et al., LHCb 2003-147.

Рукопись поступила 13 мая 2005 г.

В.Д. Матвеев, В.П. Новиков.
Коммуникационная система для централизованной архитектуры сбора данных и управления по шине CAN-bus.

Оригинал-макет подготовлен с помощью системы Word.
Редактор Н.В. Ежела.

Подписано к печати 18.05.2005. Формат 60 × 84/8. Офсетная печать.
Печ.л. 1,25. Уч.– изд.л. 1. Тираж 130. Заказ 50. Индекс 3649.
ЛР №020498 от 17.04.97.

ГНЦ РФ Институт физики высоких энергий,
142284, Протвино Московской обл.

Индекс 3649

П Р Е П Р И Н Т 2005–13, ИФВЭ, 2005