Драйвер CAN-устройств на основе архитектуры многопотокового менеджера ресурсов ОС РВ QNX6

С.В. Криволапов, И.В. Сорочинский (РГУ ПС)

Представлен менеджер ресурса (драйвер), предназначенный для работы в ОС QNX 6.x с адаптерами полевой последовательной шины CAN (Controller Area Network).

Менеджер ресурса (драйвер) предназначен для работы в ОС QNX 6.x с адаптерами полевой последовательной шины CAN (Controller Area Network).

Особенности решения технологических задач потребовали реализации дополнительных возможностей по сравнению с известными на сегодняшний день драйверами. Необходимы были дополнительные требования к его программному интерфейсу с прикладными клиентскими задачами:

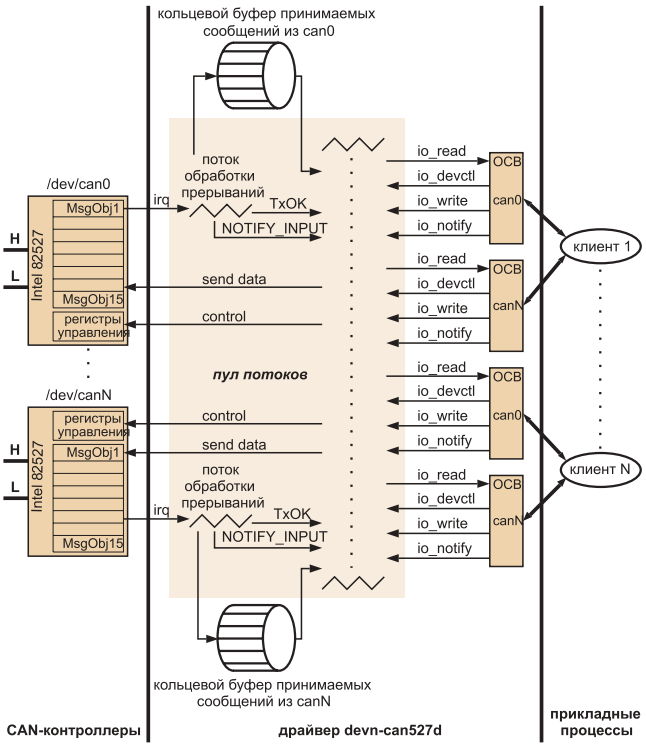
• возможность одновременного подключения нескольких клиентов, с различными способами взаимодействия (с блокировкой или без нее) и правами доступа;

• необходимость получения всеми клиентами всех принятых сообщений.

А также особенности аппаратной реализации используемых адаптеров привели к необходимости написания собственного драйвера.

Аппаратная реализация адаптера

Адаптеры CAN104D и CANPC527D производства фирмы Каскод (С.-Петербург) построены по двухканальной схеме на базе CAN-контроллера Intel 82527, реализующего физический интерфейс ISO 11898, и имеют гальванически изолированный физический интерфейс.



Данные адаптеры функционально полностью аналогичны и отличаются только используемой для подключения к компьютеру шиной: PC104 и ISA соответственно.

Наиболее важные особенности аппаратной реализации адаптера:

• наличие на плате двух независимых CAN-контролеров;

• поддержка 11-и 29-битных идентификаторов сообщения;

• отображение регистров CAN-контроллеров в основную память;

• возможность конфигурирования базового адреса диапазона используемой памяти и номеров используемых прерываний с помощью установленных на плате адаптеров переключателей.

Структурно каждый CAN-контроллер Intel 82527 состоит из регистров управления и 15 маскируемых регистров данных размером по 8 байт. 14 из них – общего назначения и могут конфигурироваться для приема/передачи в зависимости от требований задачи. 15-й специализирован только на прием, имеет теневой буфер и может маскироваться отдельно.

Как показал опыт, не всякий адрес, задаваемый переключателями на плате адаптеров из допустимого диапазона, будет работать корректно. Все зависит от объема памяти, установленной на графическом адаптере, использующемся в компьютере. При этом значение начального базового адреса должно устанавливаться в старшем диапазоне адресов. Эти издержки отображения регистров CAN-контроллера в основную память с лихвой окупаются простотой и скоростью работы с внутренними регистрами платы адаптера.

Поскольку со стороны ПО нет никаких различий при работе с вышеуказанными адаптерами, в дальнейшем будет идти речь только об одном устройстве.

Особые требования к драйверу

Особенности решения технологических задач в области автоматизации потребовали реализации дополнительных возможностей по сравнению с известными на сегодняшний день драйверами.

В процессе разработки драйвера были реализованы дополнительные требования к его программному интерфейсу с прикладными клиентскими задачами:

• возможность одновременного подключения нескольких клиентов с различными способами взаимодействия (с блокировкой или без нее) и правами доступа;

• необходимость получения клиентами всех принятых сообщений.

Программная организация драйвера

При создании драйвера использовался высокоуровневый механизм обработки сообщений микроядра, известный под названием менеджер ресурсов. Внутреннее устройство драйвера представлено на рисунке.

Так как к драйверу может быть подключено несколько процессов-клиентов, взаимодействие с которыми должно происходить одновременно и независимо, наиболее подходящей для применения была технология многопотокового менеджера ресурсов с использованием динамического пула потоков.

Потоки обработки взаимодействия с клиентами и обработки прерываний от адаптера образуют пул потоков драйвера, который создается с помощью функции **thread_pool_create()**.

При открытии клиентом одного из файлов CAN-устройства функцией **open()**, драйвер вносит его в динамический список подключенных клиентов.

При вызове клиентом функции **write()** для передачи сообщения в CAN-устройство в одном из потоков пула вызывается функция обработки **io_write()**, которая заносит сообщение в свободный для передачи регистр CAN-контроллера и блокирует поток функцией **pthread_cond_timedwait()** в ожидании прерывания от CAN-контроллера. Функция обработки прерывания после проверки соответствующих регистров разблокирует клиента и передает ему признак успешной или неудачной передачи сообщения.

При приеме сообщения CAN-контроллером возникает прерывание, которое обрабатывается внутри микроядра. При этом одним из потоков пула будет получен импульс, приоритет которого позволяет обработать прерывание в первую очередь по сравнению с остальными потоками, обслуживающими запросы клиентов. Этот поток заносит сообщение в приемный кольцевой буфер и посылает импульс (но уже с обычным приоритетом) для вызова в другом потоке пула функции обработчика принятых сообщений. Обработчик посылает уведомление всем клиентам, ожидающим получения нотификации после вызова ими функции **ionotify()**, а затем выбирает из динамического списка подключенных клиентов тех, которые были блокированы на функции **read()** и отсылает им полученные сообщения.

Пример 1

devn-can527d [опции]* [ioaddr[,irq0,irq1[,brate0,brate1]]]* &

Опции:

- q num (decimal) размер приемного буфера (по умолчанию 256)
- c num (decimal) 1=только /dev/can0 , 2=только /dev/can1, 0=оба выключены (по умолчанию оба включены)

Параметры:

ioaddr (hex)	базовый адрес физической памяти
irq0 (decimal)	номер прерывания для устройства /dev/can0
irq1 (decimal)	номер прерывания для устройства /dev/can1
brate0 (decimal)	скорость шины CAN в Кбит/с устройства /dev/can0
brate1 (decimal)	скорость шины CAN в Кбит/с устройства /dev/can1

Скорость шины CAN может принимать только значения: 500, 250, 125, 100, 50, 20, 10 Кбит/с

При запуске драйвера с опциями по умолчанию: ***devn-can527d &***

- будут смонтированы два устройства */dev/can0* и */dev/can1* с базовыми адресами физической памяти *0xD8000* и *0xD8100* (адрес второго устройства в каждом адаптере всегда смещен на 0x100 относительно первого из-за особенностей аппаратной реализации), номерами прерывания *5* и *7* соответственно. Скорость общения по CAN шине будет составлять у каждого из устройств *125 Кбит/с*, а размер приемного буфера *256* сообщений по 8 байт каждый.

Если запустить драйвер со следующими параметрами:

devn-can527d DC000,10,11 -q64 DC200,10,7,100,50 -c2 DC400 &

- будет смонтировано пять устройств. У первых двух будут адреса *0xDC000* и *0xDC100*, как это указанно в первом параметре командной строки. Следующие два параметра говорят о номерах прерывания для них: *10* и *11* соответственно. Четвертый параметр - *q64* определяет размер приемного буфера (64 сообщения) для второй пары устройств: */dev/can2 и /dev/can3*. Восьмой и девятый параметры показывают скорость CAN шины для этой пары: *100 Кбит/с* и *50 Кбит/с* соответственно. Десятый параметр отключает первое устройство третьей пары, таким образом остается только */dev/can5* с адресом *0xDC500* (одиннадцатый параметр). Все пропущенные опции означают установку их по умолчанию.

Ниже показаны устройства, которые будут смонтированы, и их основные параметры.

```
/dev/can0 is addr=0xDC000, irq=10, bit rate=125, queue=256
/dev/can1 is addr=0xDC100, irq=11, bit rate=125, queue=256
/dev/can2 is addr=0xDC200, irq=10, bit rate=100, queue=64
/dev/can3 is addr=0xDC300, irq=7,  bit rate=50,  queue=64
/dev/can5 is addr=0xDC500, irq=7,  bit rate=125, queue=256
```

Запуск драйвера

В QNX 6.x драйвер может быть запущен и остановлен в любое время. При запуске драйвер регистрирует префиксы в пространстве имен путей **/dev/can0 ... /dev/canN**, в зависимости от опций командной строки. Опционально задается, сколько плат должен обслуживать драйвер, какие контроллеры надо активировать и какие у них будут имена, базовые адреса памяти, номера прерываний, скорость CAN шины и размеры приемных буферов. Пример 1 иллюстрирует возможности задания параметров в командной строке.

Организация взаимодействия с "клиентами" драйвера

Интерфейс общения с клиентами построен на стандартных POSIX функциях таких, как: **open(), close(), read(), write(), ionotify()**. Размер посылаемого сообщения должен быть кратен размеру структуры **can-msg_t**. Процессы пользователей могут открывать фай-

Пример 2

```
#pragma pack(1)

#define CAN_MSG_LENGTH  8

typedef struct                  //Структура идентификатора
                                //сообщения
{unsigned char id0;
 unsigned char id1;
 unsigned char id2;
 unsigned char id3;
}Devid;

typedef struct                  //Структура сообщения,
                                //пересылаемого
                                //между драйвером и клиентом
{unsigned int flags;            //Дополнительная информация
 struct timespec timestamp;//Дата и время прихода сообщения
 Devid          dev;            //Идентификатор сообщения
unsigned char   lendata;        //Длина данных, передаваемых
                                //по шине CAN
 unsigned char  data[CAN_MSG_LENGTH]; //Данные
                                //передаваемые по шине CAN
}canmsg_t;

int main(int argc, char **argv)
{
 int fd, size, i;
 canmsg_t msg;
 //открытие устройства /dev/can0 на чтение
 if((fd = open("/dev/can0", O_RDONLY)) == -1)
  {
    printf("error open /dev/can0\n");
    exit(EXIT_FAILURE);
  }
 for(;;)
  {
    //чтение из устройства /dev/can0 с блокировкой
    if((size = read(fd, &msg, sizeof(msg))) == -1)
      printf("error read /dev/can0\n");
    else
     {
      //печать идентификатора сообщения
      printf("%02X%02X %02X%02X:",
        msg.dev.id0 & 0x1F, msg.dev.id1,
        msg.dev.id2, msg.dev.id3);

      //печать данных
      for(i=0; i<msg.lendata; i++)
        printf(" %02X", msg.data[i]);

      //печать даты и времени прихода сообщения
      printf(" %s,%d", ctime(&msg.timestamp.tv_sec),
            msg->timestamp.tv_nsec);
      printf("\n");
     }
  }

 exit 0;
}
```

лы CAN-устройств с блокировкой, без блокировки или с использованием нотификации, при этом с любой формой доступа, в произвольной комбинации.

Для иллюстрации приведен пример 2, который выводит на экран все сообщения, пришедшие по шине в **/dev/can0** устройство. В конце печатается дата и время прихода сообщения, которое драйвер сохраняет в структуре **canmsg_t**.

Для задания параметров работы CAN контроллера и внутренних модулей драйвера используется функция **devctl()**, с помощью которой можно динамически изменять следующие параметры:

- отключение/подключение устройства к шине CAN;
- скорость передачи данных шины CAN;
- размер циклических приемных буферов.

Особенности программной архитектуры драйвера

Отличительной особенностью описываемого драйвера является процедура общения с клиентами – сообщения из приемного буфера одновременно рассылаются всем открытым на чтение клиентам. При этом не имеет значения, каким способом подключился клиент (с блокировкой, без блокировки или по уведомлениям), расположен он с другими клиентами в одном процессе или даже узле, подключен к одному или нескольким устройствам.

Для обработки прерываний от CAN устройства используются потоки из динамического пула. Вызов функции обработчика прерывания происходит при получении импульса от микроядра, который повышает приоритет потока по сравнению с приоритетами остальных потоков пула. Таким образом, нет необходимости в создании статических специализированных, предназначенных только для ожидания и обработки прерываний потоков (по числу подключенных устройств), которые существуют на всем протяжении работы драйвера. В случае использования динамического пула, потоки создаются и уничтожаются по мере необходимости, причем не при возникновении обрабатываемых событий, а заранее, что обеспечивает минимальное время задержки до начала обработки события.

Отладка и перенос проекта в QNX Momentics 6.3.0

На этапе написания драйвера выявились проблемы со стабильностью работы отладчика в интегрированной среде разработки QNX Momentics 6.2.0. Так, в отладчике практически невозможно было просмотреть содержимое регистров контроллера Intel 82527 напрямую, т.е. структуру, указывающую на адрес, полученный от функции mmap_device_memory(). Спасал положение просмотр дампа памяти. Ситуация кардинально изменилась после перехода на QNX Momentics 6.3.0. Просмотр содержимого регистров контроллера I82527, расположенного на плате адаптера, не вызывало зависания IDE, да и общая стабильность как отладчика, так и самой IDE значительно возросла. Перенос проекта выполнялся без помощи встроенных средств переноса. Был создан новый пустой проект в IDE 6.3.0, в который при помощи команды Import были перенесены файлы с исходными текстами драйвера.

Окончательная отладка драйвера производилась на одноплатном промышленном компьютере с установленным на нем адаптером CAN104D формата PC104. Целевая система была соединена локальной сетью с компьютером разработчика по протоколу TCP/IP. Хочется отметить, что удобство работы в интегрированных средствах разработки при таком расположении

адаптера, практически не отличалось от работы, когда отладка производилась на одном компьютере.

Тестирование драйвера

Проверка работоспособности драйвера производилась на маломощных, по современным меркам, малогабаритных встраиваемых промышленных компьютерах с процессором Cyrix 300 МГц и 128 Мбайт оперативной памяти. На этих компьютерах все радиаторы без охлаждающих вентиляторов! Скорость CAN шины бала установлена 125 Кбит/с. В этих условиях драйвер показал уверенную и стабильную работу. В процессе проверки выяснилось, что при 5...6 активных клиентах и генерируемом трафике почти на пределе пропускной способности шины, наблюдалась потеря небольшого числа сообщений. Увеличение размера приемных буферов с 32 до 256 сообщений, позволило решить эту проблему.

Заключение

Предлагаемая программная архитектура при внесении минимального числа изменений, связанных с аппаратными особенностями может использоваться при разработке драйверов для любых последовательных устройств, предназначенных для приема и передачи сообщений.

Список литературы

1. ISO 11898 Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication. 1993-11-15.
2. 82527 Serial Communications Controller. Architectural Overview. – Intel Corporation, 1996.
3. *Роб Кертен.* Введение в QNX Neutrino 2. С.-Петербург. Петрополис. 2001.
4. QNX Neutrino RTOS. System Architecture – QNX Software System Ltd. 2002.
5. QNX Neutrino RTOS. Programmer's Guide. – QNX Software System Ltd. 2002.

Сорочинский Игорь Владиславович *– инженер,*

Криволапов Сергей Владимирович *– научный сотрудник, научно-исследовательской лаборатории "Системы диспетчерского контроля и управления" Ростовского государственного университета путей сообщения.*

Контактный телефон (8632) 59-49-74.

ISaGRAF 5.0 – новый шаг в развитии ПО для систем автоматизации

Компания ICS Triplex ISAGRAF Inc (www.icstriplex.ca) с гордостью объявляет о прорыве в развитии ПО для систем автоматизации. 30 мая 2005 г. компания планирует официальный выпуск бета версии ISaGRAF 5.0, первого в мире коммерческого программного пакета для систем контроля и управления, который соответствует стандарту IEC 61499. Стандарт IEC 61499, разработанный Международной Электротехнической Комиссией (МЭК), – это пакет рекомендаций по использованию функциональных блоков в распределенном промышленном процессе, процессе измерения и системах контроля и управления.

ISaGRAF 5.0 представляет собой естественное развитие технологии ISaGRAF, объединяя стандарты IEC 61131-3 и IEC 61499 новый продукт предлагает беспрецедентные функциональные возможности и преимущества. Используя продукт ISaGRAF 5.0, пользователи будут иметь возможность создавать традиционные системы контроля и управления, где итерации между устройствами автоматически регулируются и синхронизируются согласно функциональной блок-схеме стандарта IEC 61499, а не с помощью алгоритмов, созданных вручную.

ISaGRAF 5.0 включает все преимущества известной технологии TIC(Target Independent Code), которая дает возможность использовать исполнительную систему на любой аппаратной платформе (embedded, Motorola, Intel, ARM и т.д.) под управлением любой ОС (Win XP, CE, 2000, Linux, VxWorks, QNX и т.д.), а также может быть использована для создания законченных устройств автоматизации типа миниконтроллеров, RTUs, PLCs, DCS и систем противоаварийной защиты.

Фактически, при использовании ISaGRAF 5.0, любой производитель полевых устройств будет иметь способность превратить свое оборудование в продукт, который соответствует стандарту IEC 61499. Например, интеллектуальные измерительные устройства могут работать как устройства регулирования и управления клапанами и одновременно быть частью интегрированной РСУ.

Бета версия ISaGRAF 5.0 первоначально будет предлагаться ограниченному набору партнеров компании ICS Triplex ISaGRAF, которые, принимая участие в тестировании бета версии ISaGRAF 5.0, будут активно сотрудничать с группой поддержки ICS Triplex ISaGRAF.

Официальным поставщиком решений компании ICS Triplex ISaGRAF в России и странах СНГ является компания ФИОРД.

Http://www.fiord.com

Технический семинар, посвященный системам машинного зрения

20 апреля 2005 г. специалисты компании "ПБ "Экстера" провели технический семинар, посвященный системам машинного зрения, где рассматривались вопросы, связанные с применением цифровых видеокамер в современных технологиях машинного зрения. На семинаре прозвучали доклады, посвященные: обзору рынка промышленных цифровых видеокамер; перспективным направлениям в развитии систем автоматизации производства и технологии National Instruments; специализированным цифровым видеокамерам для научных исследований и автоматизации эксперимента; примерам применения цифровых видеокамер в промышленности и науке. Демонстрировались реализованные компанией проекты по применению систем машинного зрения в различных отраслях промышленности, а также работа отдельного оборудования для создания систем машинного зрения. В частности, участники семинара смогли убедиться в простоте создания приложений на базе продукта NI CVS-1454.

Речь шла и о скоростных цифровых видеокамерах. Участники семинара познакомились с различными системами регистрации быстропротекающих процессов и возможными областями их применения. На семинаре подробно освещались вопросы, связанные с особенностями технической реализации систем скоростной съемки, работы скоростных видеокамер. Демонстрировался аппаратно-программный комплекс скоростной съемки, разработанный специалистами компании "ПБ "Экстера".

Http://www.extera.ru