2009 International Conference on Industrial Mechatronics and Automation

Application and Implementation of CAN Bus Technology in Industry Real-time Data Communication

Xiao-feng WAN,Yi-si XING, Li-xiang CAI
College of Information Engineering
Nanchang University
Nanchang, China
e-mail: xfwan@ncu.edu.cn

Abstract—**The real-time data communication of real-time operation system is a difficult point. This paper analyzes the functions and characteristics of embedded real-time operation system VxWorks and CAN field bus, and makes use of CAN field bus to realize the data communication between embedded real-time operation system VxWorks and work-site MCU (Micro-Controller Unit）. The applications of CAN bus can satisfy the reliability and real-time request of data communication completely. The general steps of CAN bus applications offer reference for CAN bus applications in real-time operation system.**

Keywords-CAN; VxWorks; Communication

I. FOREWORD

Controller Area Network (CAN) is a kind of multi-host local network that leads to be offered by Bosch Company in modern automobile technology. CAN protocol is stipulated as international standards by 150 International Standard Organization. CAN protocol is a high-speed serial bus which follows 150/051 model and contains three-tier protocols: physical layer, data link layer and application layer. In the practical application system of CAN bus, various field intelligent devices represent a node separately. By means of CAN field bus, we can realize the information transmission and communication among all nodes, as well as between the field nodes and process control management, meanwhile, various complicated automatic control function also could be achieved.

VxWorks, developed by WindRiver Company, is a high qualified real-time operation system with leading industrial position. Simultaneously, it is suitable for all mainstream CPU target platform with retractility, easy-clipped and high reliability. VxWorks real-time operation system is composed of more than 400 relatively independent and brief modules, the user may choose the appropriate modules according to the need to tailor and dispose system, therefore, the system's security and reliability are guaranteed effectively. System linker may link some object modules automatically on the basis of applications need. Through the on-demand combination among the object modules, the user may obtain lots of applications which could meet the functions need.

II. OVERALL DESIGN SCHEME

The Xin Hualong C8051F046 device is fully integrated mixed-signal system-on-a-chip MCU with on-chip VDD monitor, 64 digital I/O pins, watchdog timer, clock oscillator and an on-chip CAN 2.0B controller. The C8051F046 MCU is selected as a network node. Its on-chip integrated CAN controller could realize the data exchange between MCU and VxWorks operation system through CAN transceiver and CAN field bus. Embedded operation system VxWorks has the high performance mechanism such as: preemption scheduling, multitask and so on, so it could be in the first time to collect mass data from different types and buses by means of roll polling. VxWorks real-time system based on the AT91RM9200 embedded kernel board could communicate with American Microchip MCP2510 Can controller by the industry-standard serial peripheral interface (SPI), and also could realize the data exchange with Can bus using the CAN controller.

System structure diagram in Fig. 1:

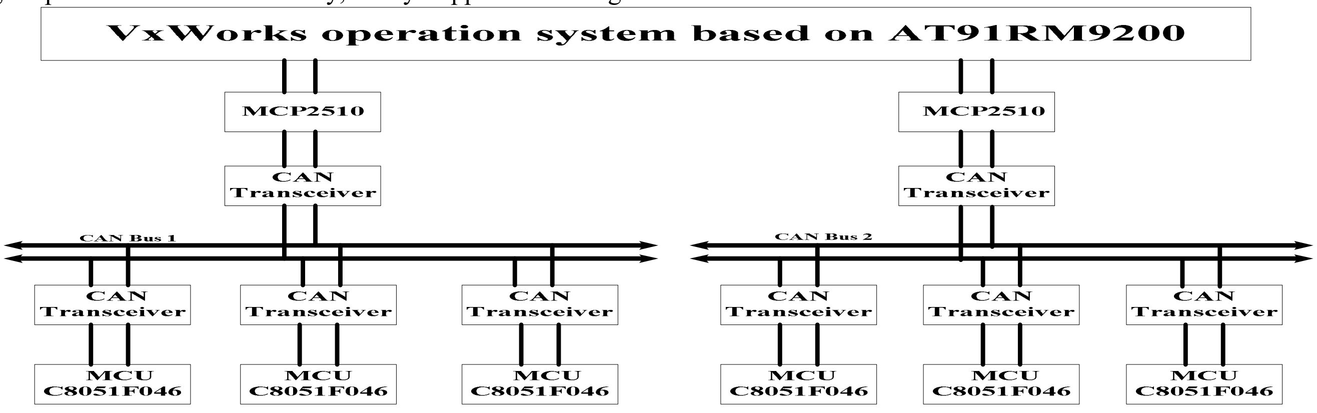

Figure 1. System structure diagram

A. Implementation of the data exchange between VxWorks real-time system and CAN bus:

1) Hardware connection between embedded kernel board and CAN controller:

Serial peripheral interface (SPI) is a kind of for-wire synchronous serial interface. The data communication would be valid when the chip-select signal (CS) is low. The data would be transmitted by the three-wine interface composed of serial data input (SI), serial data output (SO) and serial clock (SCK). Each SPI system is made up by a host and one or more slave computers. The host is the microcontroller to provide the clock signal of SPI, and the slave computer is the any integrated circuit to receive the SPI signal.

ATMEL Company AT91RM9200 takes the ARM920TARM/Thumb processor as the foundation to construct its system. It is rich in system, application peripheral device and standard interface. It also provides serial peripheral interface (SPI) with a length of 8~16 programmable data bits and may connect 4 peripheral devices. And, Peripheral Data Controller (PDC) provides the DMA channel to all serial peripheral devices. So it doesn't have to transmit data with on-chip or off-chip memory through processor. This would reduce the processor's cost when transmitting continual data stream. This scheme selects the AT91RM9200 as the host of SPI system. Microchip MCP2510 CAN protocol controller with the capability of transmitting and receiving the standard or expansion message is selected as the slave computer. Simultaneously, it also has the acceptance filtering mechanism as well as message management function. It contains three transmission buffers and two receiver buffers, reduces the management burden of the host.

Structure diagram of data transceiver in Fig. 2:

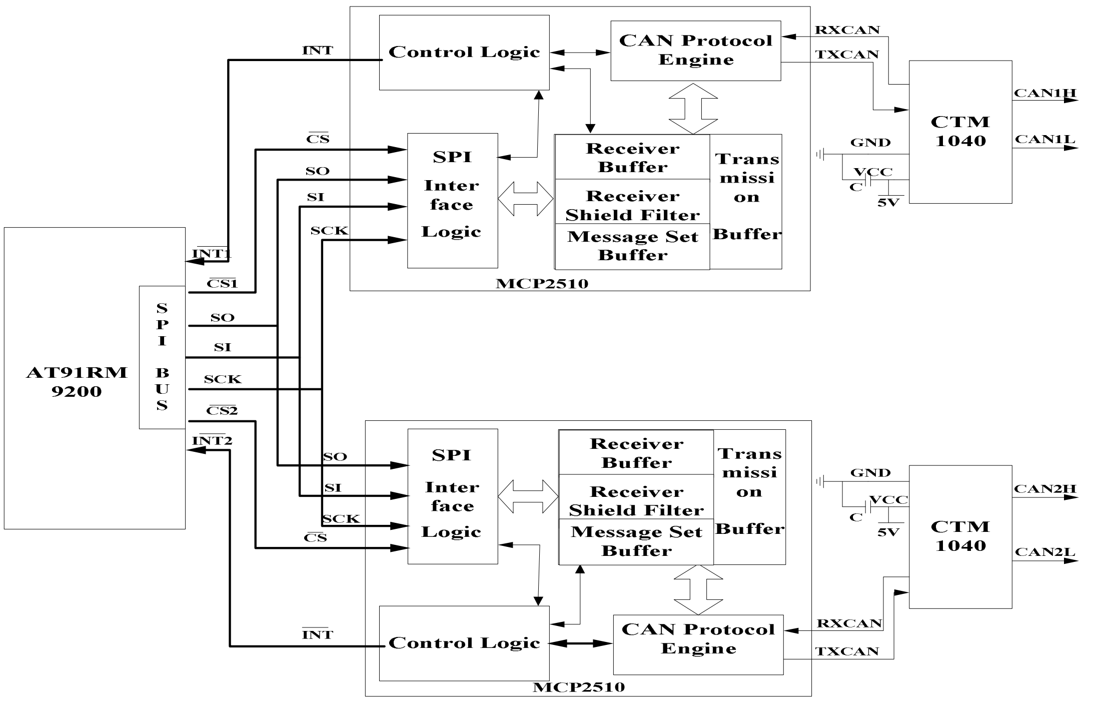

Figure 2. Structure diagram of data transceiver

As shown: embedded operation system VxWorks is carried on the data communication with two groups of CAN bus. The message transmission and receiver are treated by the protocol engine. When message transmitting, the first step is to load the message to the correct message buffer and the control register. Using control register bit, the operation for transmission and receiver would be started through SPI. The status of communication and mistakes may be checked by reading the corresponding register. Any message detected on CAN would be carried on the error detection, then, would be sent to match with the user definition's filter, so as to determine whether to transmit it to one of tow receiver buffers.

2) The data flow of VxWorks system and CAN bus:

Data exchange flow chart of VxWorks and CAN bus in Fig. 3:

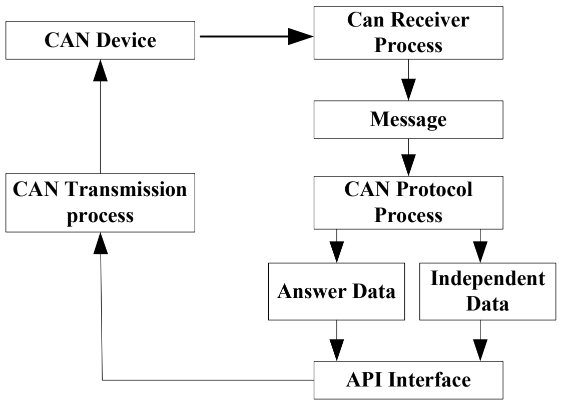

Figure 3. Data exchange flow chart of VxWorks and CAN bus

#define AT91C_BASE_SPI ((AT91PS_SPI) 0xFFFE0000) /* (SPI) Base address */

typedef volatile unsigned int AT91_REG; /* Register definition */

typedef struct _AT91S_SPI

{

AT91_REG SPI_CR; /* Control register */

AT91_REG SPI_MR; /* Model register */

AT91_REG SPI_RDR; /* Receiver data register */

AT91_REG SPI_TDR; /* Transmission data register */

} AT91S_SPI, *AT91PS_SPI;

oxFFFE000 is the SPI bus register address of AT91RM9200 kernel board. Corresponding the register with the structure, assign the value to the structure body is equivalent to complete the register's operation.

CAN receiver process LOCAL void bcanGetDataTask (AT91_CAN_IO_CHAN *pChan // CAN equipment structure body) makes a real-time monitor upper CAN controller MCP2510. When the SPI bus monitors that the controller has received the new data, call the function

LOCAL STATUS spiCanRead (AT91PS_SpiDesc pDesc,

// CAN equipment description table structure body object

UCHAR address, // Address

UCHAR *buf, // Data storage address pointer

UCHAR size // Data sizes);

to read the data and transmit it to the upper message queue through the message.

msgQSend(pChan->canTaskInfo.msgCanRecDealId, // CAN message ID NO.

(char*)&canRecData, // CAN data storage address pointer

sizeof(canRecData) // Data sizes) ;

CAN protocol process takes out the message from message queue, unpacks the data bale according to the request of CAN protocol, and distinguishes whether the received data is the answer data or the independent data. Though the API interface, transmit the data to the application layer for data processing.

When VxWorks operation system has to transmit the data to MCU though CAN bus, it would use the API function to send the data to the CAN transmission process, and transmit the data from CAN controller to CAN bus though the transmission process.

LOCAL STATUS spiCanWriteBit (AT91PS_SpiDesc pDesc, // CAN equipment description tables structure body object

UCHAR address, // Address

UCHAR mask, // Shiled

UCHAR data); // Data

B. The implementation of MCU data transceiver by CAN bus:

This example selects the Xin Hualong C8051F046 MCU because of its CAN controller and CAN protocol with serial communication. Silicon Labs CAN controller fully realized all the functions of Bosch CAN module and completely conforms to CAN standard 2.0B. CAN controller contains a CAN nuclear, message RAM (independent of the MCU RAM), information processing status machine and control register. The CAN nuclear provides shifting (CANTX and CANRX), the message serial/parallel conversion and other tasks related to the protocol .Message RAM may save 32 objects which could receive or transmit information on CAN network. Input data, message object and mark mask code are stored in CAN message RAM. All protocol of the data transmission and receiver filtration would be treated by CAN controller without the intervention of MCU. This makes the CPU bandwidth for CAN communication minimum. CAN register and message processor provide the interface for data transmission and status notification between CAN controller and MCU. Silicon Labs CAN is a protocol controller without the physical layer driver (i.e. transceiver).Thus, if the user want to use CAN bus, the CAN transceiver must be connected to MCU for electrical conversion in order to convert the logic signal to the balance difference code.

Design flow of CAN bus data transceiver program in Fig. 4:

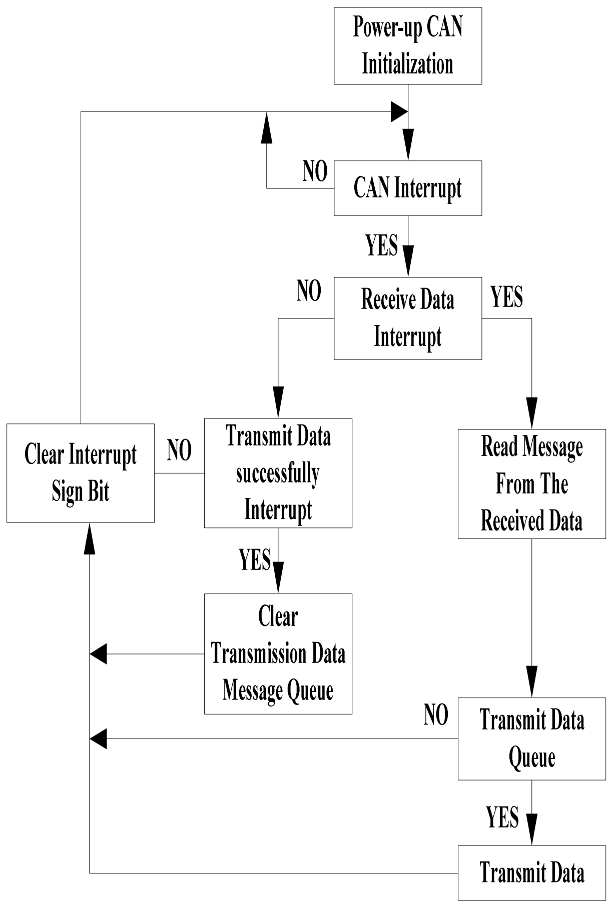
Power-up CAN Initialization
CAN Interrupt
NO
YES
Receive Data Interrupt
NO
YES
Transmit Data successfully Interrupt
Read Message From The Received Data
Clear Interrupt Sign Bit
NO
YES
Clear Transmission Data Message Queue
NO
Transmit Data Queue
YES
Transmit Data

Figure 4. Design flow of CAN bus data transceiver program

MCU approximate program as follows:

1）Initialize CAN controller

```
void can_ini()
{
   for (i =1; i<33; i++)
  {
   CAN0ADR = IF1CMDRQST;
   CAN0DATL = i;  /* Clear 32 message objects */
  }
  // Initialize receiver message object 02
    init_msg_object_RX(0x02);
 // Initialize transmission message object 01
    init_msg_object_TX(0x01);
 // Set baud rate of CAN bus
  CAN0ADR = BITREG;
  CAN0DAT = 0x5EC1;
  }
```

2）CAN interrupt program

```
void ISRname (void) interrupt 19
{
   status = CAN0STA;
   if ((status & 0x10) != 0)
    {
     receive_data (0x02); // Read data on CAN bus and
      send data based on protocol request
    }
 CAN0STA = CAN0STA|0x07; // Clear interrupt sign bit
}
```

3) CAN program of transmission data

```
  void transmit (uchar MsgNum,   // Message NO.
        uint CansendId, // Collect numbers of MCU
        uint Cansendrealnum // Transmit data length)
```

III. Conclusion

Combining with embedded Vxworks operation system and CAN Bus characteristic, this paper expounds the application examples about the data transmission based on VxWorks and CAN bus, and also provides the basic mentality for CAN bus application in real-time system.

References

[1] Rao Yuntao, Zou Jijun, Zheng Yongyun, Field Bus CAN Principle and Application Technology, 1st ed. , Beijing: Beijing Aerospace University Press, 2003, pp. 1-19,29.

[2] Cheng Jingyuan, Project Application of VxWorks Software Development [M], Beijing: China Electric Power, 2005, pp. 1-18.

[3] Yu Ming, ARM9 Embedded System Design and Development, Beijing: Electronics Industry Press, 2006.

[4] Jonathan Walker, CAN Bus Instrumentation Uses Available Data, Sim Plifles Installation, Diesel Progress 2004, 23(1): IP .