AVR Freaks

DESIGN NOTE #040

AUTHOR: CHRIS

KEYWORDS: XXXXXXXXX

Connecting an AVR Controller to CAN

Introduction

Controller Area Network (CAN) was initially created by German automotive system supplier Robert Bosch for automotive applications as a method for enabling robust serial communication. The goal was to make automobiles more reliable, safe, and fuel-efficient while decreasing wiring harness weight and complexity. Since its inception, the CAN Protocol has gained widespread popularity in industrial automation and automotive/truck applications.

Other markets where networked solutions can bring attractive benefits like medical equipment, test equipment and mobile machines are also starting to utilize the benefits of CAN.

Description

The goal of this design note is to explain how to connect the MCP2510 CAN Controller via the Serial Peripheral Interface (SPI) to an AVR 8515.

As C-compiler we will use the free AVRGCC.

In detail this note shows how to

- Connect the MCP2510 to the AVR8515 using the SPI.
- Initialize the Serial Peripheral Interface.
- Make a Software Reset.
- Write to the internal registers of MCP2510.
- Read from the internal registers of MCP2510.
- Find out that you have access to the MCP2510 CAN Controller.

CAN Controller MCP2510

CAN Controller make it “easy” to use CAN. The MCP2510 is a stand-alone CAN Controller with SPI and implements Full CAN V2.0A and V2.0B at 1Mb/s.

Features

- **0-8 Byte Message Length**
- **Standard and Extended Data Frames**
- **Programmable Bit Rate up to 1Mb/s**
- **Support for Remote Frames**
- **Two Recieve Buffers with Prioritized Message Storage**
- **Six Full Acceptance Filter Masks**
- **Three Transmit Buffers with Prioritization and Abort Features**
- **Loop-back Mode for Self Test Operation**

That means the Controller handles the whole bus activities to transmit and receive messages. All what you have to do is to send the right byte in the right register of the MCP.

While the AVR’s include a Serial Peripheral Interface it is “easy” to get access to the controller.

CAN Controller Interface PCA82C250

The PCA82C250 is the interface between the CAN Protocoll Controller and the physical bus. How to fit all together is shown in Figure 1.

Figure 1. CAN Interface

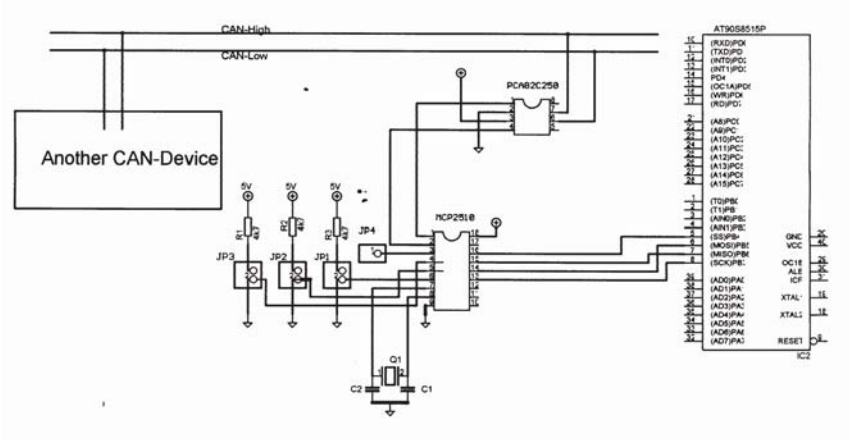

Figure 1 shows how to connect the CAN Controller to the AVR and to the PCA28C250. The MCP2510 also provides three digital inputs, e.g., to use with switches. This inputs are pulled high using R1 to R3. To simulate a closed switch you can short circuit the JP3,JP2, or JP1 pins. The MCP2510 also provide a clock output with programable prescaler on pin number three, accessable through JP4.

How to Initialize the SPI Interface

The initialization routine for the SPI is shown below.

void InitSpi (void)

```
{
  outp(0x70,PORTB);//init PortB
  outp(0xB0,DDRB);//init DDRB
  outp(0x5F,SPCR);//SPI controll register, master, enable SPI ,
// clock=mainclock/128; mode 1,1
}
```

The SP Interface

The MCP2510 is designed to interface directly with the Serial Peripheral Interface port available on many microcontrollers and supports Mode0,0 and Mode1,1. Five different operations are available.

Table 1 shows the three important instructions to communicate with the MCP2510.

Table 1. SPI-Instruction Set

Instruction Name	Instruction Format	Instruction Value	Description
RESET	1100 0000	0xC0h	Resets internal registers to default state, set configuration mode.
READ	0000 0011	0x03h	Read data from register beginning at selected address.
WRITE	0000 0010	0x02h	Write data to register beginning at selected address.

How to Make a Software Reset

The Reset command is an one-byte command. This command Resets the Internal Registers to default state. The C-code is shown below.

void ResetMcp (void)

```
{
  unsigned char counter;
  cbi(PORTB,PB4);//clear bit Chipselect -> that means pull it low
  outp(0xC0,SPDR);//send reset command
  loop_until_bit_is_set(SPSR,SPIF);//wait until byte is transmitted
sbi(PORTB,PB4);//set bit Chipselect
for (zaehler=0;counter<128;counter++);//wait a few cycles until oscillator
//is running
}
```

How to Write a Byte to the MCP2510

The Write command is three byte long. At first you have to send the write-command as shown in Table 1 (0x02h). Next you have to send the address where you want to write. Last the new value has to be written. The C-code is shown below.

void WriteMcp (Unsigned Char Value, Unsigned Char Address)

```
{
  cbi(PORTB,PB4);//set bit chipselect select MCP2510
  outp(0x02,SPDR);//send bitcombination for write
  loop_until_bit_is_set(SPSR,SPIF);//wait until byte is transmitted
  outp(adress,SPDR);//send address
  loop_until_bit_is_set(SPSR,SPIF);//wait until byte is transmitted
  outp(value,SPDR);//send value
  loop_until_bit_is_set(SPSR,SPIF);//wait until byte is transmitted
  sbi(PORTB,PB4);//set bit chipselect
}
```

How to Read a Byte From the MCP2510

The Read command is three byte long. At first you have to send the read instruction as described in Table1 (0x03h). Then the address from where you want to read must be sent. After that you have to send a byte just what you like. During this last byte the MCP2510 will move the addressed byte into the SPI Data Register (SPDR).

The C-code is shown below.

unsigned char ReadMcp (Unsigned Char Address)

```
{
  unsigned char SpiReceived;
  cbi(PORTB,PB4);//clear bit chipselect select MCP2510
  outp(0x03,SPDR);//send read-command
  loop_until_bit_is_set(SPSR,SPIF);//wait until byte transmitted
outp(Adresse,SPDR);//send adress from where you wana read
  loop_until_bit_is_set(SPSR,SPIF);//wait until byte is transmitted
  outp(0xFF,SPDR);//send any byte (just whatt you like)
  loop_until_bit_is_set(SPSR,SPIF);//wait until byte is transmitted
  SpiReceived=inp(SPDR);
  sbi(PORTB,PB4);//set bit Chipselect
  return (SpiEmpfang);

}
```

How to Find Out that Everything is Working Right

To show if there is a communication between AVR Controller and CAN Controller you can change the clockout frequency which appear on pin three (you need something like an oscilloscope). After Reset the clockout frequency is: System clock / 8, e.g., 8 MHz / 8 = 1 MHz.

You can change that frequency by writing different values to the CANCTRL Register (adress: 0x0Fh). Bit 0 and 1 are responsible for the frequency as shown below

Table 2. Prescaler

Bit1	Bit0	Prescaler	
0	0	1	
0	1	2	
1	0	4	
1	1	8	After Reset

Conclusion

Its a far way from configuration to reveive and transmit messages but I hope this note will help. For more information please have a look to the microchip homepage: www.microchip.com, and Especially to DS21291 data sheet MCP2510

Interesting application notes are AN215, AN713, AN212, AN739