International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 8, August - 2013

Hardware Implementation of Mbed to Mbed Through Controller Area Network Using ARM Cortex Core

Nulu Sudheer

PG Student

Department of Electronics and Communication Engineering, Sree Nidhi Institute of Science and Technology, Hyderabad.

Abstract

The communication between two mbed 's is achieved by placing a Controller Area Network (CAN) in between them. The CAN is another type of serial communication protocol that was developed within the automotive industry to allow a number of electronic units on a single vehicle to share essential control data. A vehicle nowadays uses many microcontrollers for autonomous control systems. CAN was developed by the German company Bosch. The CAN standard has a high level of data security. In this paper, a hardware implementation of CAN on the mbed is proposed. This uses the ARM Cortex core. In very broad terms, the mbed takes a microcontroller and surrounds it with some very useful support circuitry. It places this n a conveniently sized little printed circuit board (PCB) and supports it with an online compiler, program library and handbook. This gives a complete embedded system development environment, allowing users to develop and prototype embedded systems simply, efficiently and rapidly. Fast prototyping is one of the key features of the mbed approach.

1. Introduction

1.1 About mbed:

In very broad terms, the mbed takes a microcontroller and surrounds it with some very useful support circuitry. It places this in a conveniently sized little printed circuit board (PCB) and supports it with an online compiler, program library and handbook. This gives a complete embedded system development environment, allowing users to develop and prototype embedded systems simply, efficiently and rapidly. Fast prototyping is one of the key features of the mbed approach.

The mbed takes the form of a 2 inch by 1 inch (53mm by 26mm) PCB, with 40 pins arranged in two rows of 20, with 0.1 inch spacing between the pins.

This spacing is a standard in many electronic components .The mbed is based on around the LPC1768 microcontroller. This is made by a company called NXP semiconductors, and contains an ARM cortex-M3 core. Program download to the mbed is achieved through a universal serial bus (USB) connector; this can also power the mbed. Usefully, there are five light-emitting diodes (LEDs) on the board, one for status and four that are connected to four microcontroller digital outputs. This allows a minimum system to be tested with no external component connections needed. A reset switch is included, to force restart of the current program.

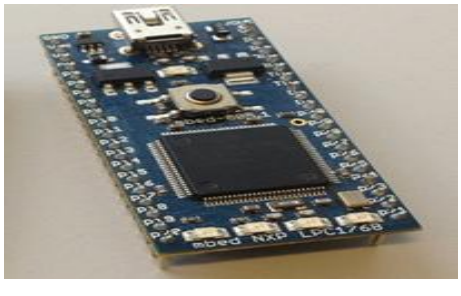

Figure 1: The ARM mbed

The mbed pins are clearly providing a summary of what each does. In many instances the pins are shared between several features to allow a number of design options. Top left we can see the round and power supply pins. The actual internal circuit runs from 3.3V. However, the board accepts any supply voltage within the range 4.5 to 9V, while an onboard voltage regulator drops this to the required voltages. A regulated 3.3V output voltage is available on the top right pin, with a 5V output on the next pin down. The remainder of the pins connects to the mbed peripherals. There are no fewer than five serial interface types on the mbed: I^2C, SPI, CAN, USB and Ethernet. Then there is set of analog inputs, essential for reading sensor values, and a set of PWM outputs useful for control of external devices, for external DC motors. While not immediately evident from the figure, pins 5 to 30 can also be configured for general digital input/output.

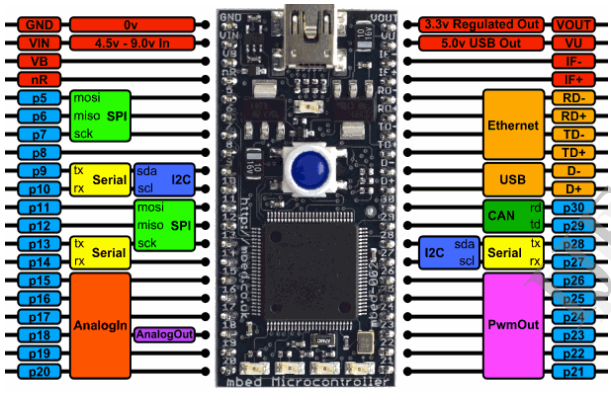

Figure 2: Pin description of mbed

The mbed is constructed to allow easy prototyping, which is of curse its very purpose. While the PCB itself is very high density, interconnection is achieved through the robust and traditional dual-in-line pin layout.

1.2 The mbed Architecture

A block diagram representation of the mbed architecture is shown in Figure 2 . It is possible and useful, to relate the blocks shown here to the actual mbed.

At the heart of the mbed is the LPC1768 microcontroller, clearly seen in Figures. The signal Pins of the mbed as connect directly to the microcontroller. Thus, when in the coming chapters we use an mbed digital input or output, or the analog input, or any other of the peripherals; we will be connecting directly to the microcontroller within the mbed, and relying on its features. An interesting aside to this, however, is that LPC1768 has 100 pins, but the mbed has only 40. Thus, when we get deeper into understanding the LPC1768, we will find that there are some features that are simply inaccessible to us as mbed users. This is, however, unlikely to be a limiting factor.

There is a second microcontroller on the mbed, which interfaces with the USB. This is called the interface microcontroller in Figure 2, and is the largest integrated circuit(IC) n the underside of the mbed PCB. The cleverness of the mbed hardware design is the way which this device manages the USB link and acts as a USB terminal to the host computer. In most common use it receives program code files through the USB, and transfers those programs to a 16M bit memory, which acts as the 'USB disk'. When a program 'binary' is downloaded to the mbed, it is placed in the USB disk. When the reset button is pressed, the program with the latest timestamp is transferred t the flash memory of the LPC1768, and program execution commences. Data transfer between interface microcontroller and LPC1768 goes as a serial data through the UART port of the LPC1768.

The 'power management' unit is made up of two voltage regulators, which lie either side of the status LED. There is also a current-limiting IC, which lies at the top left of the mbed. The mbed can be powdered from the USB; this is a common way to use it, particularly for simple applications. For more power-hungry applications or those that require a higher voltage, it can also be powered from an external 4.5 to 9.0 V input, supplied to pin 2(labeled respectively). The VU connection supplies 5V, taken almost directly from the USB link; hence it is only available if the USB is connected. The VOUT pin supplies a regulated 3.3 V, which is derived either from the USB or from the VIN input.

Figure 3: shows how mbed works

2. CAN IC MCP2551

2.1 Device Overview

The MCP2551 is a high-speed CAN, fault-tolerant device that serves as the interface between a CAN protocol controller and the physical bus. The MCP2551 provides differential transmit and receive capability for the CAN protocol controller and is fully compatible with the ISO-11898 standard, including 24V requirements. It will operate at speeds of up to 1 Mb/s .

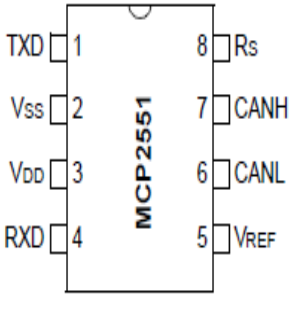

Figure 4: MCP 2551 IC

Typically, each node in a CAN system must have a device to convert the digital signals generated by a CAN controller to signals suitable for transmission over the bus cabling (differential output). It also provides a buffer between the CAN controller and the high-voltage spikes that can be generated on the CAN bus by outside sources (EMI, ESD, electrical transients, etc.).

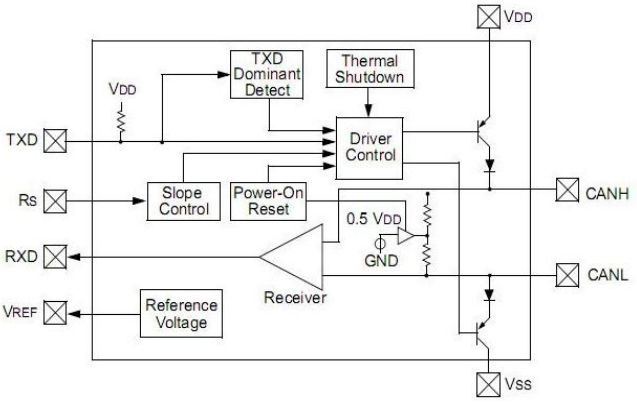

Figure 5: BLOCK DIAGRAM OF MCP2551

2.2 Transmitter Function

The CAN bus has two states: Dominant and Recessive. A dominant state occurs when the differential voltage between CANH and CANL is greater than a defined voltage (e.g., 1.2V). A recessive state occurs when the differential voltage is less than a defined voltage (typically 0V). The dominant and recessive states correspond to the low and high state of the TXD input pin, respectively. However, a dominant state initiated by another CAN node will override a recessive state on the CAN bus.

The MCP2551 CAN outputs will drive a minimum load of 45Ω, allowing a maximum of 112 nodes to be connected (given a minimum differential input resistance of 20KΩ) and a nominal termination resistor value of 120KΩ.

2.3 Receiver Function

The RXD output pin reflects the differential bus voltage between CANH and CANL. The low and high states of the RXD output pin correspond to the dominant and recessive states of the CAN bus, respectively.

2.4 Internal Protection

CANH and CANL are protected against battery short circuits and electrical transients that can occur on the CAN bus. This feature prevents destruction of the transmitter output stage during such a fault condition. The device is further protected from excessive current loading by thermal shutdown circuitry that disables the output drivers when the junction temperature exceeds a nominal limit of 165°C. All other parts of the chip remain operational and the chip temperature is lowered due to the decreased power dissipation in the transmitter outputs. This protection is essential to protect against bus line short-circuit-induced damage.

2.5 Operating Modes

The RS pin allows three modes of operation to be selected:

- High-Speed
- Slope-Control
- Standby

These modes are summarized in Table 1-1. When in High-speed or Slope-control mode, the drivers for the CANH and CANL signals are internally regulated to provide controlled symmetry in order to minimize EMI emissions. Additionally, the slope of the signal transitions on CANH and CANL can be controlled with a resistor connected from pin 8 (RS) to ground, with the slope proportional to the current output at RS, further reducing EMI emissions.

2.5.1 High-speed

High-speed mode is selected by connecting the RS pin to VSS. In this mode, the transmitter output drivers have fast output rise and fall times to support high-speed CAN bus rates.

2.5.2 Slope-control

Slope-control mode further reduces EMI by limiting the rise and fall times of CANH and CANL. The slope, or slew rate (SR), is controlled by connecting an external resistor (REXT) between RS and VOL (usually ground). The slope is proportional to the current output at the RS pin. Since the current is primarily determined by the slope-control resistance value REXT, a certain slew rate is achieved by applying a respective resistance. typical slew rate values as a function of the slope-control resistance value.

2.5.3 Standby mode

The device may be placed in standby or "SLEEP" mode by applying a high-level to RS. In SLEEP mode, the transmitter is switched off and the receiver operates at a lower current. The receive pin on the controller side (RXD) is still functional but will operate at a slower rate. The attached microcontroller can monitor RXD for CAN bus activity and place the transceiver into normal operation via the RS pin (at higher bus rates, the first CAN message may be lost).

Mode	Current at R_s Pin	Resulting Voltage at Rs Pin
Standby	$-I_{RS} < 10\ \mu A$	$V_{RS} > 0.75\ V_{DD}$
Slope-control	$10\ \mu A < -I_{RS} < 200\ \mu A$	$0.4\ V_{DD} < V_{RS} < 0.6\ V_{DD}$
High-speed	$-I_{RS} < 610\ \mu A$	$0 < V_{RS} < 0.3 V_{DD}$

Table 1: Describes the mode of CAN

2.6 TXD Permanent Dominant Detection

If the MCP2551 detects an extended low state on the TXD input, it will disable the CANH and CANL output drivers in order to prevent the corruption of data on the CAN bus. The drivers are disabled if TXD is low for more than 1.25 ms (minimum). This implies a maximum bit time of 62.5 µs (16 kb/s bus rate), allowing up to 20 consecutive transmitted dominant bits during a multiple bit error and error frame scenario. The drivers remain disabled as long as TXD remains low. A rising edge on TXD will reset the timer logic and enable the CANH and CANL output drivers.

2.7 Powers-on Reset

When the device is powered on, CANH and CANL remain in a high-impedance state until VDD reaches the voltage-level VPORH. In addition, CANH and CANL will remain in a high-impedance state if TXD is low when VDD reaches VPORH. CANH and CANL will become active only after TXD is asserted high. Once powered on, CANH and CANL will enter a high-impedance state if the voltage level at VDD falls below VPORL, providing voltage brown-out protection during normal operation

2.8 Pin Descriptions

The 8-pin pinout

Pin Number	Pin Name	Pin Function
1	TXD	Transmit Data Input
2	Vss	Ground
3	Vdd	Supply Voltage
4	RXD	Receive Data Output
5	Vref	Reference Output Voltage
6	CANL	CAN Low-Level Voltage I/O
7	CANH	CAN High-Level Voltage I/O
8	Rs	Slope-Control Input

Table 2: MCP2551 PINOUT

2.8.1 Transmitter Data Input (TXD)

TXD is a TTL-compatible input pin. The data on this pin is driven out on the CANH and CANL differential output pins. It is usually connected to the transmitter data output of the CAN controller device. When TXD is low, CANH and CANL are in the dominant state. When TXD is high, CANH and CANL are in the recessive state, provided that another CAN node is not driving the CAN bus with a dominant state. TXD has an internal pull-up resistor (nominal 25 KΩ to VDD).

2.8.2 Ground supply (VSS)

Ground supply pin.

2.8.3 Supply voltage(VDD)

Positive supply voltage pin.

2.8.4 Receiver Data Output (RXD)

RXD is a CMOS-compatible output that drives high or low depending on the differential signals on the CANH and CANL pins and is usually connected to the receiver data input of the CAN controller device. RXD is high when the CAN bus is recessive and low in the dominant state.

2.8.5 Reference voltage (VREF)

Reference Voltage Output (Defined as VDD/2).

2.8.6 CAN Low (CANL)

The CANL output drives the low side of the CAN differential bus. This pin is also tied internally to the receive input comparator.

2.8.7 CAN High (CANH)

The CANH output drives the high-side of the CAN differential bus. This pin is also tied internally to the receive input comparator.

2.8.8 Slope Resistor Input (RS)

The RS pin is used to select High-speed, Slope-control

3. Contribution of CAN

The controller area network (CAN) is another type of serial communications protocol that was developed with in automotive industry to allow a number of electronic units on a single vehicle to share essential control data. A vehicle nowadays uses many microcontrollers for autonomous control systems. Each microcontroller system is referred to as an electronic control unit (ECU) and these include the engine management ECU, an anti-braking system(ABS) ECU, a dashboard ECU, active suspension and the radio/CD player, for example. Each of these ECUs manages its own control strategies, but they all need to access information relevant to their own operation, for example drawn from engine speed, throttle position, brake pedal position and engine temperature. However, an automotive vehicle generates a high level of electromagnetic interference and a wide temperature and humidity range; this is a hostile environment for any signal and indeed for any electronic device. Moreover, very high reliability is essential as vehicles are safety-critical systems. The serial standards developed for the benign environment of home or office, such as UART, SPI and IC, are completely inappropriate for the hostile environment, and a new standard was therefore needed. Initially, CAN was developed by the German company Bosch. They published Version 2.0 of the standard in 1991, and in 1993 it was adopted by the International Organization for Standardization (ISO), as ISO 11898. At the time of writing, CAN specification Version 2.0B can be downloaded from the Bosch website. The CAN standard has a high level of data security, it is

inevitable complex and just the briefest overview is given here.

3.1 CAN on the mbed

The mbed has two CAN controller interfaces on pins 9-10 and 30-29;note that only the latter two pins appear as a CAN interface . The CAN interface can be used to write data words out of a CAN port and will return with data received from another CAN device. The clock frequency can also be configured. A selection of the CAN application programming interface (API) functions .

When using the mbed CAN API, we first have to define a CAN object and then define a message (CAN Message) to be written to or read from the CAN bus. The mbed CAN controllers alone, however, cannot be used to communicate directly with a CAN network. To do this we need to use the mbed CAN interface to control a specific CAN transceiver IC such as the Microchip MCP2551.

The CAN bus implementation used is differential, and connects to the CANH and CANL pins. An external resistor connected at Rs controls the slew rate of the data signal, with a slower rate minimizing electromagnetic interference; although for simple testing this can be connected directly to ground. The mbed connects from its CAN controllers to the transceiver's RXD and TXD pins.

To demonstrate CAN with the mbed we will communicate some data between two mbeds over a CAN network. One mbed will send data to the CAN bus while the other will receive data from the CAN bus. The hardware connections required to create a simple CAN network test are shown in above figure and table. Notice that the 'CAN bus' is defined as the wires CANH and CANL, which sometimes require a termination resistance of 100-200Ω between them over long wires. In the simple, noise-free example the short connections mean that a termination resistor is not necessary. Note also that we use the mbed CAN controller on pins 30 and 29 for the send system and the CAN controller uses pin 9 and 10- these are arbitrarily chosen, predominantly to make the hardware diagram neat.

3.2 Block Diagram of CAN on the mbed

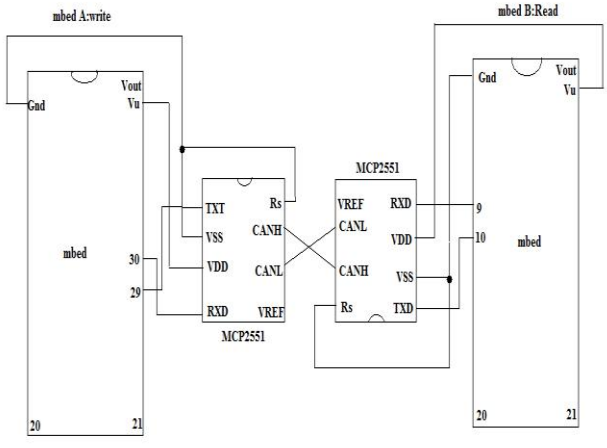

Figure 6: Hardware build for mbed-to-mbed CAN communication

Notice that the 'CAN bus' is defined as the wires CANH and CANL, which sometimes require a termination resistance of 100-200Ω between them over long wires. In the simple, noise-free example the short connections mean that a termination resistor is not necessary. Note also that we use the mbed CAN controller on pins 30 and 29 for the send system and the CAN controller uses pin 9 and 10- these are arbitrarily chosen, predominantly to make the hardware diagram neat

The mbed CAN controllers alone, however, cannot be used to communicate directly with a CAN network. To do this we need to use the mbed CAN interface to control a specific CAN transceiver IC such as the Microchip MCP2551.

. In the simple, noise-free example the short connections mean that a termination resistor is not necessary. Note also that we use the mbed CAN controller on pins 30 and 29 for the send system and the CAN controller uses pin 9 and 10- these are arbitrarily chosen, predominantly to make the hardware diagram neat.

MCP2551 pin	Description	Pin connection
Write system		
1 TXD	Transmit Data Input	mbed p29 CAN TD
2 VSS	Ground	mbed p1 GND
3 VDD	Supply Voltage	mbed p39 Vu(5V)
4 RXD	Receive Data Output	mbed p30 CAN RD
5 Rs	Reference Output Voltage	mbed p1 GND
6 CANL	CAN Low-Level Voltage I/O	CAN bus low
7 CANH	CAN High-Level Voltage I/O	CAN bus high
8 VREF	Slope-Control Input	---
Read system		
1TXD	Transmit Data Input	mbed p10 CAN TD
2 VSS	Ground	mbed p1 GND
3 VDD	Supply Voltage	mbed p39 Vu (5 V)
4 RXD	Receive Data Output	mbed p9 CAN RD
5 Rs	Reference Output Voltage	mbed p1 GND
6 CANL	CAN Low-Level Voltage I/O	CAN bus low
7 CANH	CAN High-Level Voltage I/O	CAN bus high
8 VREF	Slope-Control Input	---

Table 6: pin connections between mbed to- CAN IC

3.3 Example program to show CAN data write

```
#include"mbed.h"

Serial pc(USBIX, USBRX); // tx , rx for Tera Term

                                    Output

DigitalOut led1(LED1);        //status LED

CAN can1(p30, p29);

Char counter=0;

int main()

 {

    Printf("send….");

   While(1)

     {

    //send value to CAN bus AND monitor return value to check if CAN message was sent successfully. If so display, increment and toggle.

  If(can1.write(CANMessage(1 ,&counter,1)))

  {

 pc.printf("Message sent: %d\n",counter);   //display

      couter++;                    //increment

led1=!led1;                        //toggle status LED

else

can1.reset();

}

Wait(1);

}

}
```

3.4 Example program to show CAN Read data

```
 #include"mbed.h"

Serial  pc(USBTX, USBRX);
//tx,rx for Tera Term output

DigitalOut led2(LED2);        // status LED

CAN  can1(p9, p10);           // CAN interface

Int main()

{

CANMessage;     //create empty CAN message

Printf("read….\n");

While(1)

{

If(can1.read(msg))    // if message is available, read

                          Into msg

{

      Printf("Message received: %d\n" , msg.data[0]);
                                //display message  data


led2 =  ! led2  ;        //toggle status LED

}

}

}
```

4. Security aspects:

The data is very secured as CAN Provides peer-to-peer configuration and transferred in frames.CAN bus provides message filtering to provide whether data on the bus is relevant to them. The high level of data security with exhaustive error checking.

5. Results

The result seen on the Tera Term provides that the data is secured one. As the mbed is used for rapid proto-typing, this can be implemented in Automobile industry and wherever the communication between two controllers is needed.

6. Conclusion

From the security aspects and the results derived, the hardware implementation is easy between mbed's through CAN. The results might also conclude that this hardware implementation can also be used wherever, fast prototyping and data security needed. The dumping of a program into mbed is easy as explained through online compiling. The CAN bus "message filtering" techniques shows how data is secured.

7. References

[1] Rob Toulson and Tim Wilmshurst, Fast and Effective Embedded System Design, Communicating control data over the CAN, PP.288-292

[2] The CAN section of the Bosch website. www.can.bosch.com

[3] FlexRayTM. "About Flex Ray™" [Website] http://www.flexray.com, visited on 28th March, 2012.

[4] Microchip Technology (2003). MCP2551 High speed CAN Transceiver Data sheet. Document DS21730E.

[5] [Website],http://zone.ni.com/devzone/cda/tut/p/id/3352 , visited on 28th March, 2012.

[6] National Instruments. "Flex Ray Automotive Communication Bus Overview"

[7] Passemard, Michel. "Atmel Microcontrollers for Controller area Network (CAN)", Atmel Corporation, 4069A-CAN-02/04, http://www.atmel.com [Website].

[8] CAN in Automation. "CAN made easy Basic information on the CAN physical and data link layer", CiA, retrieved December 13th, 2011, from www.cancia. org/pg/can/additional/can_basics_print.pdf.

[9] www.mbed.org

[10] mbed CAN-bus Demo Board. http://mbed.org /forum/ news-announce