Basic operation of CBUS Bootloader by Mike Bolton

Introduction

This is a small piece of assembly code that resides in a 18F2480 PIC and allows applications code (hex files) to be loaded into a CBUS module over the CAN bus. It is an adaptation of the bootloader code in the Microchip AN247. You should refer to this document for more details of the command and data structures.

The Boot code (Boot2.asm)

This occupies 512 bytes of the PIC from address 0000 to 0x01FF. The boot block can be protected from accidental overwrites but the 18F2480 boot block protection has a minimum size of 2K bytes (0x07FF) So this can be utilised if wanted, the user code must start at 0x0800. The HPINT vector is relocated to 0x0808 and the LPINT vector to 0x0818.

The Boot2 code assumes a PIC 18F2480 with an oscillator frequency of 4MHz and the usual CAN interface. This is standard on all the CBUS modules (so far).

The Boot code uses extended CAN frames. This has the advantage of avoiding conflict with CBUS frames which are standard frames only. However, the CAN_USB module which feeds the bootloader needs the latest code (rev g) so it can send and receive both types of CAN frame. Also a CAN_RS may be used with rev d code.

Bootloader commands and data.

The loader accepts two types of frame. A command frame and a data frame. It also sends message frames back to the PC. Within a command frame there are 5 different commands specified.

Code	command	Description
00h	No operation	No op. but used to set the loading address.
01h	Reset	Issues a software reset and clears boot mode. the module now runs normally. *
02h	Initialise	Clears all boot registers ready for loading.
03h	Check	Sent at end of programming and has the checksum from the PC. If no errors the boot code sends a reply. Also sends an error reply if needed. *
04h	Test	Boot code sends a message if the module is in a boot state.*

* These actions are different from the AN247 ones.

Data frames.

These are 8 bytes of data. They are loaded into successive bytes in the PIC. They must contain all 8 bytes. If the data do not occupy all 8 bytes, they must be filled with FFh bytes. The starting address is set by a command frame. The loader automatically increments the addresses till another command frame is received.

The bootloader also erases any pre-existing user program bytes.

PC program requirements.

Output strings required from the PC to the CAN_USB

Commands

NOP (00h)

:X00080004N, “addrl”, “addrh”, “addru”, 000D**00**0000;

This sets the start address of any data block. The address is 3 bytes long. e.g. for the usual code start of 00 08 00, the string would be:

:X00080004N000800000D**00**0000;

For the start of the EEPROM (F0 00 00) section it would be

:X00080004N0000F0000D**00**0000;

Reset (01h)

:X00080004N000000000D**01**0000;

The address bytes are not relevant here.

Initialise (02h)

:X00080004N, “addrl”, “addrh”, “addru”, 000D**02**0000;

This initialises the bootloader and sets the start address. e.g. for a start of 00 08 00 it would be

:X00080004N000800000D**02**0000;

Check (03h)

:X00080004N000000000D**03** “chkh” chkl” ;

Chkh and chkl are the two lower hex bytes of the 2s complement of the addition of all the data bytes. If the checksum was 4A 3B then the string would be

:X00080004N000000000D**03**4A3B;

Test (04h)

:X00080004N000000000D**04**0000;

The address bytes are not relevant. This is sent to check if a module is in boot mode. As this frame is extended, it will only be received if the module is in boot mode. If so, the response is

:X80080004N**02**;

All response messages are one byte. See below for response messages.

If the module is not in boot mode, there is no reply so the PC program must have a timeout. One second is more than enough.

Data frames

All data frames have the following format.

:X00080005N <data0><data1><data2><data3><data4><data5><data6><data7> ;

The data is sent with the lowest address location first. This is the same as in the HEX file. There must be 8 bytes in every data frame. Pad with FFh if needed. Again, each data byte in HEX is represented by two ASCII chars.

Messages from the bootloader.

Error message. Sent after a 'Check' frame if there is a checksum or verify error.

:X80080004N**00**;

OK message. Sent after a 'Check' frame if there is no checksum or verify error.

:X80080004N**01**;

Boot mode confirm. Sent in response to a 'Test' message.

:X80080004N**02**;

Boot or run mode identification

The bootloader uses the last byte in EEPROM (F000FF) to identify whether a PIC is in boot mode or run mode. If this byte is FF, then it is in boot mode. An erased or virgin PIC will have this byte as FF by default. When powered on or otherwise reset, the bootloader first checks this byte. If it is FF it activates the bootloader code. If it is not FF, it goes straight to the run mode. The user code must not modify this byte.

Once in run mode, the PIC cannot be returned to boot mode unless the EEPROM byte is reset to FF. This is a function of the user code. We have decided to use the CBUS OPC of 5C to do this. A CBUS command of <5C><NNhi><NNlo> should set the last EEPROM byte to FF and execute a 'reset' PIC command. This is a standard CBUS message

:S0000N5C <NNhi><NNlo>;

For SLiM consumers, the NN is irrelevant. For nodes with a NN, then only that node should go into boot mode.

If this is done by mistake, you can exit the boot mode by sending a reset frame to the bootloader.

:X00080004N000000000D**01**0000;

I suggest that the PC program used for bootloading has the ability to send a standard CBUS 5C message to a specified node number so it can put a PIC into boot mode if it already has an application code running.

Notes:

The bootloader requires the CAN_USB (or CAN_RS) interface to send / receive both standard and extended frames. Also the header bytes (SIDH, SIDL, EIDH and EIDL) are set by the PC and not by the interface module. The CAN_USB code Rev g must be used. The CAN_RS equivalent is rev d.

The bootloader requires time to program the PIC memory bytes. There is no handshake. The PC program should have delays of 10millisecs between program data frames and 50millisecs between config and EEPROM data frames. The program (Flash) data is written 8 bytes at a time but the config and EEPROM data is written one byte at a time.

Except for the DCC CABs, existing CBUS modules have a green LED (on RB7) and a yellow LED (on RB6). When in boot mode, the bootloader program puts both of these LEDs on. This should be the only situation in which both LEDs are on and steady. The CABs put both the forward and reverse LEDs on. This is a way of indicating a module is in boot mode.

The bootloader program now allows for a configuration which includes a watchdog timer (WDT).

I have a program for loading etc. in VB5 if anyone is interested. This takes the Intel HEX file produced by the Microchip MPLAB IDE.

Mike B 30/12/09