Knowledgebase

FCU BootLoader Process

The FCU bootloader for PIC processors is as follows:

1. Upon selecting a node to be reloaded the FCU will prompt for a hex file to be loaded.
2. The target processor of the hex file (Address 0x828) is compared with the module's processor type (RQNPN param 15) . A warning is generated if they are incompatible.
3. The FCU will send that node a BOOTM CBUS message.
4. The module will write 0xFF into the top most byte of EEPROM (the boot flag) and then cause a processor reset.
5. The bootloader code resides at address 0x0000 and will be executed at reset.
6. The bootloader checks the value of the boot flag and if it is 0x00 it will pass control to the module application code.
7. Otherwise the bootloader awaits receipt of extended CAN messages containing the address and data to be written to FLASH program memory plus EEPROM and CONFIG memory if requested. The HEX files will contain any EEPROM and CONFIG data by default but the user can opt to not program these in the FCU settings.
8. Upon receiving the final bootload message FCU will send a RESET command and the module will verify the data, clear the boot flag and perform a reset itself.

Note that FCU will automatically fill any unused space up to the highest specified address with 0xFF. Therefore uninitialised data must occupy space **above** the program space.

It is also believed that FCU will ignore any sections of the hex file below 0x800 as this is reserved for the bootloader, which shouldn't be overwritten by itself.

Important. The PIC18F2*K80 allow the PLL to be enabled using CONFIG bits. This should **NOT** be done. The standard bootloader clears CLKSEL in CIOCON which means the CAN clock is driven by the PLL output. The CAN bit rate is calculated assuming the PLL is disabled. If the CONFIG enabled the PLL then the Bootloader's CBUS timing will be disrupted and loose communications with the FCU towards the end of the bootloading process.

Bootloader CAN Message Format

Commands: Put commands received from source (Master –> Slave) The count (DLC) can vary.

XXXXXXXXXXX 0 0 8 XXXXXXXX XXXXXX00 ADDRL ADDRH ADDRU RESVD CTLBT SPCMD CPDTL CPDTH
XXXXXXXXXXX 0 0 8 XXXXXXXX XXXXXX01 DATA0 DATA1 DATA2 DATA3 DATA4 DATA5 DATA6 DATA7

Where:
ADDRL - Bits 0 to 7 of the memory pointer.
ADDRH - Bits 8 - 15 of the memory pointer.
ADDRU - Bits 16 - 23 of the memory pointer.
RESVD - Reserved for future use.
CTLBT - Control bits.
SPCMD - Special command.
CPDTL - Bits 0 - 7 of 2s complement checksum
CPDTH - Bits 8 - 15 of 2s complement checksum
DATAX - General data.

Control bits:
MODE_WRT_UNLCK-Set this to allow write and erase operations to memory.
MODE_ERASE_ONLY-Set this to only erase Program Memory on a put command. Must be on 64-byte boundary.

MODE_AUTO_ERASE-Set this to automatically erase Program Memory while writing data.
MODE_AUTO_INC-Set this to automatically increment the pointer after writing.
MODE_ACK-Set this to generate an acknowledge after a 'put' (PG Mode only)

Special Commands:

Command	Bit	Usage
CMD_NOP	0x00	Do nothing
CMD_RESET	0x01	Issue a soft reset after setting last EEPROM data to 0x00
CMD_RST_CHKSM	0x02	Reset the checksum and verify
CMD_CHK_RUN	0x03	Add checksum to special data, if verify and zero checksum
CMD_BOOT_TEST	0x04	Just sends a message frame back to verify boot mode

Bootloader use of Extended Frames

An extended frame has the same number of data bytes compared to a standard frame, but it has a larger id field of 29 bits instead of 11 bits. This is implemented in a PIC by providing 2 extra header bytes, called EIDH and EIDL. An extended frame is identified by having bit 3 of the standard frame header byte SIDL set to a 1. All standard frames have this set to zero.

The bootloader code sets up a hardware filter so that it will only accept extended frames, and then only where all bits of EIDH are 0 and bits 3-7 of EIDL are 0.

It then uses EIDL bit 0 to flag whether a packet is control or data, and bit 1 to flag if it is a get or put operation (the boot loader can also be used to read data from a module, but I don't think we have implemented that anywhere for CBUS).

I can't see it referencing EIDL bit 2 anywhere, presumably reserved for future use.

So this means that extended frames can also be used for other things without conflicting with the bootloader by using extended frame where any bits in EIDH and any of bits 3-7 of EIDL are set to 1, so the bootloader code will ignore such frames.

Normal CBUS messages are carried in standard CAN frames so the code that receives these messages can safely ignore all extended frames, which, depending on processor capabilities, can either be done with a hardware filter setting or by the code checking for the extended frame bit in the CAN header.

Bootloader Module requirements

Since the bootloader itself is not overwritten during a load from FCU there must be compatibility between the bootloader execution and the module application. The PIC18F2XkXX series of PICs supports PLL which may be enabled either within configuration bits or under software control. To permit compatibility and interoperability when operating with a 16MHz oscillator the PLL should be DISABLED in configuration bits so that the bootloader executes at a clock speed of 16MHz. The application can enable the PLL if required so that the application executes at 64MHz.
