David C Brown (Jodrell Bank Observatory, University of Manchester) Version 0.1 25th March 2002

0. Introduction

The latest family of PIC microcontrollers are able to write to their own program memory space. This allows a small bootloader program to receive new program code over the SPI bus (which is normally used to communicate with the M&C bus).

1. Implementation

A small bootloader routine resides in a protected area in the top 256 words of program memory and the reset vector contains a jump to the start of this routine. The first action of the routine is to examine the SPI chip select line. If this line is inactive the bootloader jumps to the start of normal code. If the line is active the bootloader reads new code from the SPI bus and reprograms the memory. This process is explained in detail in the next section.

Whilst new code is being loaded the PIC must be regarded as unusable and a flag at the very top of memory is set to indicate this state. The flag is only reset when the full program has been satisfactorily loaded and will be left set if the programming sequence fails for any reason. If the bootloader finds that the flag is set it will not allow normal program execution to start.

2. Limitations

Three limitations must be pointed out. If the unprotected code at the reset vector (three words because of the need to page switch) becomes corrupt the scheme bcomes unworkable and the device must be reprogrammed in the normal way. Also the main program code must always start at a fixed location. This should not usually be a problem as the fixed location can be regarded as an alternative restart vector and be programmed with a jump. Finally it is not possible to reprogram the EEPROM data memory.

3. Bootloader process

The routine waits until the chip select line is deasserted then enables the SPI system. It then sends NULL bytes until it receives an RDY byte in response. If the RDY byte is not received within a few seconds it is assumed that the PIC is disconnected or broken and the programming attempt is abandoned.

The programming device then sends a single record, in the format described below, without any handshaking. Having received the entire record satisfactorily, without any data overruns, and stored it in RAM, the bootloader evaluates the CRC. If this is correct it programs the block, a process that can take ten milliseconds per word.

The programming device polls the routine by sending NULL bytes which will be returned unchanged until the programming is complete or has been aborted. The routine will return a code either indicating that the programming has completed successfully or indicating the nature of the fault.

This continues until a special last transfer record is received, after which the bootloader will stop execution. The last transfer record is characterised by having a special header.

4. Error Checking

The bootloader checks on each byte that there has been no data overrun. It checks that the CRC is correct. It checks that the number of words in each record is equal to that given in the header. This latter check requires two implementation details. First: the chip select line must be active during the entire block, not activated on a byte-by-byte basis. Second: the chip select line must be connected to another input pin on the PIC so that it can be read by the program. If these conditions cannot be met the length checks cannot be made. Even without those checks a correct CRC gives a very high level of confidence that the transfer was successful.

The programming device must make the following checks before sending the record. The block length is no more than 32 words. The address for each word must be greater than 0x03 and less than 0x1F00 (16F876/7) or 0x0F00 (16F873/4) or 0x700 (16F870/1). Each data word is not greater than 0x3FFF.

5. Record Format

This is loosely modelled on the Intel Hex Format but is word oriented. Each record has the following format:

XB AA DD DD ... DD DD CC

Where

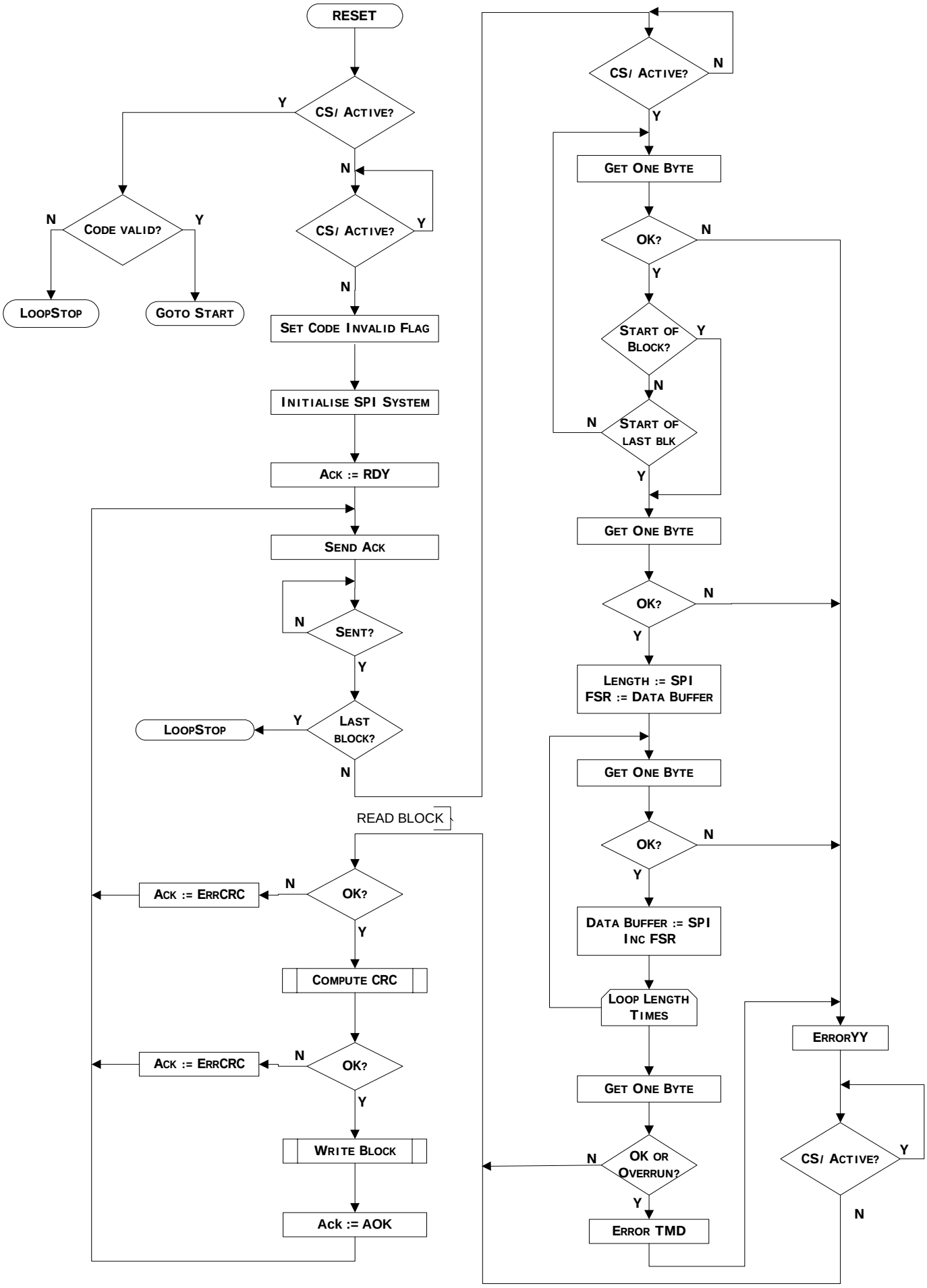
- X is the start of record byte. $D1 for normal, $E2 for the last record
- B is the number of data words in the record. Although the format allows for 256 the loader has a maximum record size of 32 words
- AA is the word address to which the record is to be loaded. The most significant byte is transmitted first
- DD are the data bytes transmitted, most significant byte first
- CC is a cyclic redundancy check of the address and data words computed to the CCITT polynomial $x^{16}+x^{12}+x^2+1$ with an initial value of 0xFFFF

6. Return Codes

$A4	AOK	success
$89	CRC	cyclic redundancy check fail
$92	OVR	data overrun
$C7	NED	block is shorter than indicated by header
$DC	TMD	block is longer than indicated by header

7. Reference

Microchip Technology Application Note AN732
(http://www.microchip.com/1000/suppdoc/appnote/category/listing/index9.htm#432)


RESET
CS/ ACTIVE?
Y
N
CODE VALID?
N
Y
LOOPSTOP
GOTO START
CS/ ACTIVE?
Y
N
SET CODE INVALID FLAG
INITIALISE SPI SYSTEM
ACK := RDY
SEND ACK
SENT?
N
Y
LAST BLOCK?
Y
LOOPSTOP
N
READ BLOCK
OK?
N
ACK := ERRCRC
Y
COMPUTE CRC
OK?
N
ACK := ERRCRC
Y
WRITE BLOCK
Ack := AOK
CS/ ACTIVE?
N
Y
GET ONE BYTE
OK?
N
Y
START OF BLOCK?
Y
N
START OF LAST BLK
N
Y
GET ONE BYTE
OK?
N
Y
LENGTH := SPI
FSR := DATA BUFFER
GET ONE BYTE
OK?
N
Y
DATA BUFFER := SPI
INC FSR
LOOP LENGTH TIMES
GET ONE BYTE
OK OR OVERRUN?
N
Y
ERROR TMD
ERRORYY
CS/ ACTIVE?
Y
N

```
MPASM 02.70 Released            BOOT.ASM   3-25-2002  11:35:03         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001 ;==========================================================================
                     00002
                     00003         LIST    P=16F877,N=0
                     00004         TITLE   " ALMA BootLoader "
                     00005
                     00006         INCLUDE <D16F877.INC>   ;
                     00001 ; P16F877.INC  Standard Header File, Version 1.00    Microchip Technology Inc
                     00372         LIST
                     00007
2007   2F22          00008        __CONFIG _CP_UPPER_256 & _DEBUG_OFF & _WRT_ENABLE_ON & _LVP_OFF & _BODEN_OFF & _PWRTE_ON & _WDT_O
                           FF & _HS_OSC
                     00009
2000   0000 0004 0000 00010        __IDLOCS    0X0401
       0001
                     00011
                     00012         INCLUDE         <DCBMACRO.INC>
                     00001 ; DCBMACRO.INC   Simple macros.  Versin 1.10    dcb
                     00337         LIST
                     00013
                     00014 ;==========================================================================;
                     00015 ; Define the area of memory occupied by the program.   It always resides in   ;
                     00016 ; the top 256 words of a page so bootstart is 0x70,0xf0,0x17 or 0x1F.  The    ;
                     00017 ; valid word is always at the very top of memory                              ;
                     00018
                     00019
                     00020         #DEFINE         PAG     3               ; Code resides at top of page
  00001F00           00021         BOOTSTART       EQU     (8*PAG+7)*100
  0000001F           00022         VALH            EQU     (8*PAG+7)
  000000FF           00023         VALL            EQU     0XFF
  00001FFF           00024         VALID           EQU     VALH*100 + VALL
                     00025         ;
                     00026
```

```
00027 ;*******************************************************************************
00028 ; These are registers and bits used  by the program - mapped across all banks ;
00029 ; The area from 0x20 to 0x6F is used to buffer the received data allowing a   ;
00030 ; block size of 38 data words plus address and CRC words                      ;
00031
00032         #DEFINE ADDRH   0X20
00033         #DEFINE ADDR    0X21
00034         #DEFINE DATAH   0X22
00035         #DEFINE DATA    0X23
00036
00037         #DEFINE RESVRD  0X70    ; Used by debugger when in ICD mode
00038
00039         #DEFINE CRCD_H  0X78    ; Used in CRC routine only
00040         #DEFINE CRCD_L  0X79    ; A temporary buffer for the data word
00041         #DEFINE CRCA_H  0X7A    ; and an accumulator for the CRC
00042         #DEFINE CRCA_L  0X7B    ;
00043
00044         #DEFINE ABUFH   0X78    ; Buffers for flash read and write
00045         #DEFINE ABUF    0X79    ; overlaid on CRC so be carefull
00046         #DEFINE ACNT    0X7A    ;
00047
00048         #DEFINE ICNT    0X7C    ; A general loop counter
00049         #DEFINE BCNT    0X7D    ; A bit counter
00050         #DEFINE LEN     0X7E    ; Length in WORD
00051         #DEFINE FLAGS   0X7F
00052
00053         #DEFINE LSTBLK  FLAGS,7 ; Set to whenlast block is received
00054         #DEFINE SPIF    FLAGS,0 ; Set if an error receiving byte
00055
00056         #DEFINE CSINN   PORTB,0 ; The readable chip select pin
00057
00058 ;*******************************************************************************
00059 ; CONSTANTS                                                                    ;
00060 ;                                                                             ;
00061         #DEFINE SOB     0XD0    ; Start of block code
00062         #DEFINE SLB     0X0D    ; Start of last block code
00063
00064         #DEFINE RDY     0XAA    ; Ready Code
00065         #DEFINE AOK     0XA0    ; AcknowledgeCode - good block
00066
00067         #DEFINE ERR_CRC 0XA1    ; Acknowledge Code - CRC error
00068         #DEFINE ERR_NED 0XA2    ; Acknowledge Code - block shorter than header
00069         #DEFINE ERR_OVR 0XA3    ; Acknowledge Code - data overrun
00070         #DEFINE ERR_TMD 0XA4    ; Acknowledge Code - block longer than header
00071
00072
```

```
      ;=========================================================================;
00073 ; Macro to check the CRC of the received data block.   It is only used once  ;
00074 ; but included as a macro for clarity of main loop                           ;
00075 ;                                                                           ;
00076 ; AT START The data buffer must contain the block to be checked              ;
00077 ;          LEN must contain the number of data words in the block           ;
00078 ;                                                                           ;
00079 ; AT END   IF CRC is correct: Z set,W=0.                                     ;
00080 ;           If CRC is faulty: Z clear,W=ERR_CRC                             ;
00081 ;          ICNT,BCNT,FSR,CRCD_H,CRCD_L ,CRCA_H,CRCA_L are undefined         ;
00082 ;-------------------------------------------------------------------------:
00083
00084 CRC     MACRO
00085
00086         #DEFINE HASH_H  0X10
00087         #DEFINE HASH_L  0X12
00088
00089         MOVFF   LEN,ICNT        ; Number of words
00090         INCF    ICNT,F          ; Add one for address
00091
00092         MOVLW   0XFF            ; Initialise the accumulator
00093         MOVWF   CRCA_H          ;
00094         MOVWF   CRCA_L          ;
00095
00096         MOVLF   ADDRH,FSR
00097
00098 CRC2    MOVFF   INDF,CRCD_H     ; Read a word into the buffer
00099         INCF    FSR,F           ;
00100         MOVFF   INDF,CRCD_L     ;
00101         INCF    FSR,F           ;
00102
00103
00104         MOVLF   0X10,BCNT       ; Set bit counter to 16
00105 CRC3    RRF     CRCD_H,F        ; Shift the data one place right
00106         RRF     CRCD_L,F        ;
00107         RRF     CRCA_H,F        ; into the accumulator
00108         RRF     CRCA_L,F        ;
00109
00110         BTFSS   STATUS,C        ; If a carry out occurs
00111         GOTO    CRC4            ; then hash the accumulator
00112         XORLF   HASH_H,CRCA_H,F ;
00113         XORLF   HASH_L,CRCA_L,F ;
00114
00115 CRC4    DECFSZ  BCNT,F          ; Loop on all bits in word
00116         GOTO    CRC3            ;
00117
00118         DECFSZ  ICNT,F          ; Loop on all words in block
00119         GOTO    CRC2            ;
00120
```

```
00121         MOVFW   INDF            ; CRCA now contains recomputed CRC
00122         INCF    FSR             ; Get MS part of transmitted CRC
00123         XORWF   CRCA_H          ; and compare them
00124         SKPZ                    ; OK so check LS part
00125         GOTO    CRC5            ; Faulty so dont bother with LS part
00126
00127         MOVFW   INDF            ; Get LS part of transmitted CRC
00128         XORWF   CRCA_L          ; and compare them
00129
00130 CRC5    MOVLW   0
00131         SKPZ
00132         MOVLW   ERR_CRC
00133
00134         ENDM
00135
00136
00137 ;=============================================================================;
00138 ; Macro to get a block of data from the SPI.  It is only used once but is    ;
00139 ; included as a macro for clarity of main loop                                ;
00140 ;                                                                            ;
00141 ; AT START no conditions                                                     ;
00142 ;                                                                            ;
00143 ; AT END LEN contains the number of words in the transfer                    ;
00144 ;        If the block is received without error: W=0                        ;
00145 ;        If an overrun is detected: W=ERR_OVR                               ;
00146 ;        If the transfer terminates before LEN words are received: W=ERR_NED ;
00147 ;        If the transfer continues after LEN words are received: W=ERR_TMD   ;
00148 ;        ICNT is undefined                                                  ;
00149 ;                                                                           ;
00150 ; Even if an error occurs the macro continues until the transfer is ended    ;
00151 ;-----------------------------------------------------------------------------;
00152
00153 GETBLK  MACRO
00154
00155 GB6     BTFSC   CSINN           ; wait for start of transfer
00156         GOTO    GB6             ;
00157
00158 GB0     MOVLW   0
00159         CALL    GETONE          ; Look for the start byte
00160         BTFSC   SPIF
00161         GOTO    GBZ1            ; error can be OVR or NED
00162
00163         SUBLW   SOB             ; Test the byte for start of block code
00164         SKPNZ
00165         GOTO    GB2             ; If it is then carry on
00166
00167         ADDLW   SLB             ; Test the byte for start of last block
00168         SKPZ
00169         GOTO    GB0             ; If it isn't ignore the byte and try again
```

```
00170
00171          BSF     LSTBLK
00172
00173 GB2      CALL    GETONE          ; Get the length (in words)
00174          BTFSC   SPIF
00175          GOTO    GBZ1            ; error can be OVR or NED
00176
00177          MOVWF   LEN             ; Save length for CRC and WRITE routines
00178          MOVWF   ICNT            ; Move length counter
00179          INCF    ICNT,F          ; Add one for ADDR
00180          INCF    ICNT,F          ; and one for CRC
00181          BCF     STATUS,C            ;
00182          RLF     ICNT,F          ; double length to get number of bytes
00183
00184          MOVLW   ADDRH           ; Start of data buffer to FSR
00185          MOVWF   FSR             ;
00186
00187 GB3      CALL    GETONE          ; Get rest of data in loop
00188          BTFSC   SPIF            ;
00189          GOTO    GBZ1            ; error can be OVR or NED
00190
00191          MOVWF   INDF            ; Store data in buffer
00192
00193          INCF    FSR,F           ; Loop LEN+2 times
00194          DECFSZ  ICNT,F          ; If block terminates early we will have
00195          GOTO    GB3             ; jumped to GBZ1
00196
00197          CALL    GETONE          ; There shouldn't be any more data
00198          BTFSS   SPIF            ; So if a good byte is received
00199          GOTO    GBZ0            ; there is too much data
00200
00201          XORLW   ERR_NED         ; If no data remains block length is correct
00202          SKPNZ                   ; and W=0
00203          GOTO    GBZ1
00204                                  ; any other error indicates
00205 GBZ0     MOVLW   ERR_TMD         ; too much data
00206
00207 GBZ1     BTFSS   CSINN           ; wait for end of transfer
00208          GOTO    GBZ1
00209
00210          ENDM
00211
00212
00213
```

```
                     00214 ;*******************************************************************************
0000                 00215         ORG     0
                     00216
0000   0000          00217         NOP                     ; For debugger
0001   301F          00218         MOVLW   0X1F            ; Select page 3
0002   008A          00219         MOVWF   PCLATH
0003   2F05          00220         GOTO    BOOT
                     00221
                     00222 ;*******************************************************************************
                     00223
0100                 00224         ORG     0X100
0100   2900          00225 START   GOTO    START           ; The address to which the bootloader
                     00226                                 ; returns if it is not asked to load new code.
                     00227                                 ; It must contain a jump to the real code
                     00228
                     00229 ;*******************************************************************************
                     00230
1F00                 00231         ORG     BOOTSTART      ; Trap overrun of main code
                     00232
1F00   301F          00233         MOVLW   0X1F            ; Select page 3
1F01   008A          00234         MOVWF   PCLATH
                     00235
                     00236         MOVLF   0XF0,PORTD      ; Turn on top red LEDs
1F02   30F0              M         MOVLW   0XF0
1F03   0088              M         MOVWF   PORTD
1F04   2F04          00237 BOVR:   GOTO    BOVR            ; and loop stop
                     00238
                     00239 ;*******************************************************************************
                     00240
1F05   1C06          00241 BOOT    BTFSS   CSINN
1F06   2F0D          00242         GOTO    BOOTA           ; CS is low so new code is to be loaded
1F07   0000          00243         NOP                     ; CS is high so existing code is run
                     00244
1F08   27D1          00245         CALL    VAL_READ
1F09   1D03          00246         SKPZ
1F0A   2F0A          00247 BOOTX:  GOTO    BOOTX           ; Code not valid - loopstop
                     00248
1F0B   018A          00249         CLRF    PCLATH
1F0C   2900          00250         GOTO    START           ; Valid code - execute it
                     00251
```

```
                     00252 ;-------------------------------------------------------------------------
                     00253
                     00254
1F0D   1C06          00255 BOOTA:  BTFSS   CSINN           ; Wait for CS to release
1F0E   2F0D          00256         GOTO    BOOTA           ;
1F0F   0000          00257         NOP
                     00258
                     00259         MOVLF   VALH,ABUFH      ; Set the validity byte to invalid
1F10   301F              M         MOVLW   VALH
1F11   00F8              M         MOVWF   0X78
                     00260         MOVLF   VALL,ABUF       ;
1F12   30FF              M         MOVLW   VALL
1F13   00F9              M         MOVWF   0X79
                     00261         MOVLF   0X1F,DATAH      ;
1F14   301F              M         MOVLW   0X1F
1F15   00A2              M         MOVWF   0X22
                     00262         MOVLF   DATAH,FSR       ;
1F16   3022              M         MOVLW   0X22
1F17   0084              M         MOVWF   FSR
1F18   3001          00263         MOVLW   1               ;
1F19   27B1          00264         CALL    FLASH_WRITE
                     00265 ;
                     00266         MOVLF   0XFF,PORTD      ; LEDS off
1F1A   30FF              M         MOVLW   0XFF
1F1B   0088              M         MOVWF   PORTD
                     00267         MOVLF   5,SSPCON        ; Fault bits cleared. SPI disabled
1F1C   3005              M         MOVLW   5
1F1D   0094              M         MOVWF   SSPCON
                     00268                                 ; Clock idle low.  SPI mode 5
                     00269         BANK01
1F1E   1683              M         BSF     STATUS,RP0
1F1F   1303              M         BCF     STATUS,RP1
                     00270
1F20   0188          00271         CLRF    TRISD           ; LEDs enabled
                     00272         MOVLF   0X07,ADCON1     ; PORT A is all digital inputs (for SS)
1F21   3007              M         MOVLW   0X07
1F22   009F              M         MOVWF   ADCON1
1F23   1287          00273         BCF     PORTC,5         ; SDO output on PORTC-5
1F24   0194          00274         CLRF    SSPSTAT         ; Falling edge clock. BF cleared
                     00275         BANK00
1F25   1283              M         BCF     STATUS,RP0
1F26   1303              M         BCF     STATUS,RP1
                     00276
1F27   1694          00277         BSF     SSPCON,SSPEN    ; Enable SPI
                     00278
```

```
                    00279 ;-------------------------------------------------------------------------
                    00280
1F28   30AA         00281         MOVLW   RDY             ; For initial acknowledgement
                    00282
1F29   13FF         00283 BRPLY   BCF     LSTBLK          ; clear last block flag on entry
                    00284                                 ; and after a fault to facilitate retry
                    00285                                 ; on last block
                    00286
                    00287                                 ; Send acknowledge code to master which is
1F2A   1314         00288 BRPL1:  BCF     SSPCON,SSPOV    ; polling and may have overflowed
1F2B   1394         00289         BCF     SSPCON,WCOL     ; As master is polling  write will only
1F2C   0093         00290         MOVWF   SSPBUF          ; succede when a full byte is received
1F2D   1B94         00291         BTFSC   SSPCON,WCOL
1F2E   2F2A         00292         GOTO    BRPL1
                    00293
                    00294         BANK01
1F2F   1683             M         BSF     STATUS,RP0
1F30   1303             M         BCF     STATUS,RP1
1F31   1C14         00295 BRPL2:  BTFSS   SSPSTAT,BF      ; Now wait for next byte to be received
1F32   2F31         00296         GOTO    BRPL2           ; then master will have read return byte
                    00297         BANK0
1F33   1283             M         BCF     STATUS,RP0
                    00298
1F34   1BFF         00299         BTFSC   LSTBLK          ; If it was the last block
1F35   2F96         00300         GOTO    BDONE           ; finish
                    00301
                    00302         GETBLK                  ; Reads a block from the SPI.
                        M
1F36   1806             M GB6     BTFSC   CSINN           ; wait for start of transfer
1F37   2F36             M         GOTO    GB6             ;
                        M
1F38   3000             M GB0     MOVLW   0
1F39   27A3             M         CALL    GETONE          ; Look for the start byte
1F3A   187F             M         BTFSC   SPIF
1F3B   2F5C             M         GOTO    GBZ1            ; error can be OVR or NED
                        M
1F3C   3CD0             M         SUBLW   SOB             ; Test the byte for start of block code
1F3D   1903             M         SKPNZ
1F3E   2F43             M         GOTO    GB2             ; If it is then carry on
                        M
1F3F   3E0D             M         ADDLW   SLB             ; Test the byte for start of last block
1F40   1D03             M         SKPZ
1F41   2F38             M         GOTO    GB0             ; If it isn't ignore the byte and try again
                        M
1F42   17FF             M         BSF     LSTBLK
                        M
1F43   27A3             M GB2     CALL    GETONE          ; Get the length (in words)
1F44   187F             M         BTFSC   SPIF
```

```
1F45   2F5C              M         GOTO    GBZ1            ; error can be OVR or NED
                         M
1F46   00FE              M         MOVWF   LEN             ; Save length for CRC and WRITE routines
1F47   00FC              M         MOVWF   ICNT            ; Move length counter
1F48   0AFC              M         INCF    ICNT,F          ; Add one for ADDR
1F49   0AFC              M         INCF    ICNT,F          ; and one for CRC
1F4A   1003              M         BCF     STATUS,C            ;
1F4B   0DFC              M         RLF     ICNT,F          ; double length to get number of bytes
                         M
1F4C   3020              M         MOVLW   ADDRH           ; Start of data buffer to FSR
1F4D   0084              M         MOVWF   FSR             ;
                         M
1F4E   27A3              M GB3     CALL    GETONE          ; Get rest of data in loop
1F4F   187F              M         BTFSC   SPIF            ;
1F50   2F5C              M         GOTO    GBZ1            ; error can be OVR or NED
                         M
1F51   0080              M         MOVWF   INDF            ; Store data in buffer
                         M
1F52   0A84              M         INCF    FSR,F           ; Loop LEN+2 times
1F53   0BFC              M         DECFSZ  ICNT,F          ; If block terminates early we will have
1F54   2F4E              M         GOTO    GB3             ; jumped to GBZ1
                         M
1F55   27A3              M         CALL    GETONE          ; There shouldn't be any more data
1F56   1C7F              M         BTFSS   SPIF            ; So if a good byte is received
1F57   2F5B              M         GOTO    GBZ0            ; there is too much data
                         M
1F58   3AA2              M         XORLW   ERR_NED         ; If no data remains block length is correct
1F59   1903              M         SKPNZ                   ; and W=0
1F5A   2F5C              M         GOTO    GBZ1
                         M                                 ; any other error indicates
1F5B   30A4              M GBZ0    MOVLW   ERR_TMD         ; too much data
                         M
1F5C   1C06              M GBZ1    BTFSS   CSINN           ; wait for end of transfer
1F5D   2F5C              M         GOTO    GBZ1
                         M
                     00303         TSTW                    ; if OK carry on
1F5E   3800              M         IORLW   0               ;
1F5F   1D03          00304         SKPZ
1F60   2F29          00305         GOTO    BRPLY           ; if not skip writing - error code in W
                     00306
                     00307         CRC                     ; Checks the CRC of the Block.
                         M
                         M         #DEFINE HASH_H  0X10
                         M         #DEFINE HASH_L  0X12
                         M
                         M         MOVFF   LEN,ICNT        ; Number of words
1F61   087E              M         MOVF    0X7E,W
1F62   00FC              M         MOVWF   0X7C
1F63   0AFC              M         INCF    ICNT,F          ; Add one for address
```

```
                        M
1F64   30FF             M         MOVLW   0XFF            ; Initialise the accumulator
1F65   00FA             M         MOVWF   CRCA_H          ;
1F66   00FB             M         MOVWF   CRCA_L          ;
                        M
                        M         MOVLF   ADDRH,FSR
1F67   3020             M         MOVLW   0X20
1F68   0084             M         MOVWF   FSR
                        M
                        M CRC2    MOVFF   INDF,CRCD_H     ; Read a word into the buffer
1F69   0800             M         MOVF    INDF,W
1F6A   00F8             M         MOVWF   0X78
1F6B   0A84             M         INCF    FSR,F           ;
                        M         MOVFF   INDF,CRCD_L     ;
1F6C   0800             M         MOVF    INDF,W
1F6D   00F9             M         MOVWF   0X79
1F6E   0A84             M         INCF    FSR,F           ;
                        M
                        M
                        M         MOVLF   0X10,BCNT       ; Set bit counter to 16
1F6F   3010             M         MOVLW   0X10
1F70   00FD             M         MOVWF   0X7D
1F71   0CF8             M CRC3    RRF     CRCD_H,F        ; Shift the data one place right
1F72   0CF9             M         RRF     CRCD_L,F        ;
1F73   0CFA             M         RRF     CRCA_H,F        ; into the accumulator
1F74   0CFB             M         RRF     CRCA_L,F        ;
                        M
1F75   1C03             M         BTFSS   STATUS,C        ; If a carry out occurs
1F76   2F7B             M         GOTO    CRC4            ; then hash the accumulator
                        M         XORLF   HASH_H,CRCA_H,F ;
1F77   3010             M         MOVLW   0X10
1F78   06FA             M         XORWF   0X7A,F
                        M         XORLF   HASH_L,CRCA_L,F ;
1F79   3012             M         MOVLW   0X12
1F7A   06FB             M         XORWF   0X7B,F
                        M
1F7B   0BFD             M CRC4    DECFSZ  BCNT,F          ; Loop on all bits in word
1F7C   2F71             M         GOTO    CRC3            ;
                        M
1F7D   0BFC             M         DECFSZ  ICNT,F          ; Loop on all words in block
1F7E   2F69             M         GOTO    CRC2            ;
                        M
1F7F   0800             M         MOVFW   INDF            ; CRCA now contains recomputed CRC
1F80   0A84             M         INCF    FSR             ; Get MS part of transmitted CRC
1F81   06FA             M         XORWF   CRCA_H          ; and compare them
1F82   1D03             M         SKPZ                    ; OK so check LS part
1F83   2F86             M         GOTO    CRC5            ; Faulty so dont bother with LS part
                        M
1F84   0800             M         MOVFW   INDF            ; Get LS part of transmitted CRC
```

```
1F85   06FB              M         XORWF   CRCA_L          ; and compare them
                         M
1F86   3000              M CRC5    MOVLW   0
1F87   1D03              M         SKPZ
1F88   30A1              M         MOVLW   ERR_CRC
                         M
1F89   1D03          00308         SKPZ                    ; If OK carry on
1F8A   2F29          00309         GOTO    BRPLY           ; If not skip writing - error code in W
                     00310
                     00311         MOVLF   DATAH,FSR       ; write the data to memory
1F8B   3022              M         MOVLW   0X22
1F8C   0084              M         MOVWF   FSR
                     00312         MOVFF   ADDRH,ABUFH
1F8D   0820              M         MOVF    0X20,W
1F8E   00F8              M         MOVWF   0X78
                     00313         MOVFF   ADDR,ABUF
1F8F   0821              M         MOVF    0X21,W
1F90   00F9              M         MOVWF   0X79
1F91   087E          00314         MOVFW   LEN
1F92   1D03          00315         SKPZ                    ; unless length is zero
1F93   27B1          00316         CALL    FLASH_WRITE     ;
                     00317
1F94   30A0          00318         MOVLW   AOK             ; Block written
1F95   2F2A          00319         GOTO    BRPL1           ; acknowledge
                     00320
1F96   3000          00321 BDONE:  MOVLW   0
                     00322
                     00323         MOVLF   VALH,ABUFH      ; Set the validity byte to valid
1F97   301F              M         MOVLW   VALH
1F98   00F8              M         MOVWF   0X78
                     00324         MOVLF   VALL,ABUF       ;
1F99   30FF              M         MOVLW   VALL
1F9A   00F9              M         MOVWF   0X79
                     00325         MOVLF   0X00,DATAH      ;
1F9B   3000              M         MOVLW   0X00
1F9C   00A2              M         MOVWF   0X22
                     00326         MOVLF   DATAH,FSR       ;
1F9D   3022              M         MOVLW   0X22
1F9E   0084              M         MOVWF   FSR
1F9F   3001          00327         MOVLW   1               ;
1FA0   27B1          00328         CALL    FLASH_WRITE     ;
                     00329
1FA1   1388          00330         BCF     PORTD,7         ; Finished - green LED on
1FA2   2FA2          00331 BHERE:  GOTO    BHERE           ; and loopstop
                     00332
                     00333
```

```
                     00334 ;=========================================================================;
                     00335 ; Subroutine to get a single byte from the SPI                           ;
                     00336 ;                                                                       ;
                     00337 ; ON ENTRY  W contains a byte to send                                    ;
                     00338 ;                                                                       ;
                     00339 ; ON EXIT If byte received OK: W contains the byte. SPIF clear          ;
                     00340 ;         If a data overrun occurs: W=ERR_OVR,SPIF set                  ;
                     00341 ;         If the transfer ends before a byte is received: W=ERR_NED, SPIF set ;
                     00342 ;-------------------------------------------------------------------------;

                     00343
1FA3                 00344 GETONE
1FA3   147F          00345         BSF     SPIF            ; anticipate a fault
                     00346
1FA4   1B14          00347         BTFSC   SSPCON,SSPOV
1FA5   34A3          00348         RETLW   ERR_OVR         ; data overrun
                     00349
1FA6   0093          00350         MOVWF   SSPBUF          ; Byte to send
                     00351
                     00352 GET1    BANK0                   ;
1FA7   1283              M         BCF     STATUS,RP0
1FA8   1806          00353         BTFSC   CSINN           ;
1FA9   34A2          00354         RETLW   ERR_NED         ; transfer ended without byte - error
                     00355
                     00356         BANK1
1FAA   1683              M         BSF     STATUS,RP0
1FAB   1C14          00357         BTFSS   SSPSTAT,BF      ;
1FAC   2FA7          00358         GOTO    GET1            ; Byte not received yet - try again
                     00359
                     00360         BANK0                   ; Byte received OK
1FAD   1283              M         BCF     STATUS,RP0
1FAE   0813          00361         MOVFW   SSPBUF          ; Save it
                     00362
1FAF   107F          00363         BCF     SPIF            ; Clear fault flag
1FB0   0008          00364         RETURN
                     00365
                     00366
                     00367
```

```
                     00368 ;***********************************************************************;
                     00369 ; Subroutine to write a block to flash program memory                    ;
                     00370 ;                                                                       ;
                     00371 ; ON ENTRY ABUFH contains MS Byte of memory address of start of block   ;
                     00372 ;          ABUF  contains LS Byte of memory address of start of block   ;
                     00373 ;          FSR    points to MS Byte of first data word                  ;
                     00374 ;          FSR+1  points to LS byte of first data word                  ;
                     00375 ;          W     contains the number of words to write                  ;
                     00376 ;                                                                       ;
                     00377 ; ON EXIT  ABUFH,ABUFL,FSR,ACNT are undefined                           ;
                     00378 ;-----------------------------------------------------------------------;
                     00379
1FB1                 00380 FLASH_WRITE
                     00381
1FB1   00FA          00382         MOVWF   ACNT            ; Loop counter
                     00383
1FB2   1703          00384         BSF     STATUS,RP1
1FB3   1283          00385 FW1     BCF     STATUS,RP0      ; Bank2
                     00386
1FB4   0878          00387         MOVFW   ABUFH           ; Address to Flash address buffer
1FB5   008F          00388         MOVWF   EEADRH          ;
1FB6   0879          00389         MOVFW   ABUF            ;
1FB7   008D          00390         MOVWF   EEADR           ;
                     00391
1FB8   0800          00392         MOVFW   INDF            ; Data to flash data buffer
1FB9   008E          00393         MOVWF   EEDATH          ;
1FBA   0A84          00394         INCF    FSR             ;
1FBB   0800          00395         MOVFW   INDF            ;
1FBC   008C          00396         MOVWF   EEDATA          ;
1FBD   0A84          00397         INCF    FSR
                     00398
1FBE   1683          00399         BSF     STATUS,RP0      ; Bank3
1FBF   178C          00400         BSF     EECON1,EEPGD    ; Set flash status for write to
1FC0   150C          00401         BSF     EECON1,WREN     ; program memory
                     00402
1FC1   3055          00403         MOVLW   0X55            ; The special write sequence
1FC2   008D          00404         MOVWF   EECON2          ;
1FC3   30AA          00405         MOVLW   0XAA            ;
1FC4   008D          00406         MOVWF   EECON2          ;
1FC5   148C          00407         BSF     EECON1,WR       ;
1FC6   0000          00408         NOP                     ;
1FC7   0000          00409         NOP                     ;
1FC8   110C          00410         BCF     EECON1,WREN     ;
                     00411
1FC9   0AF9          00412         INCF    ABUF,F          ; Next address
1FCA   1903          00413         SKPNZ                   ;
1FCB   0AF8          00414         INCF    ABUFH,F         ;
                     00415
```

```
1FCC   0BFA           00416         DECFSZ  ACNT            ; Loop
1FCD   2FB3           00417         GOTO    FW1             ;
                      00418
1FCE   1283           00419         BCF     STATUS,RP0      ; Bank0
1FCF   1303           00420         BCF     STATUS,RP1
1FD0   0008           00421         RETURN
                      00422
                      00423 ;*************************************************************************;
                      00424 ; Subroutine to read the validity word                                    ;
                      00425 ;                                                                        ;
                      00426 ; ON ENTRY VAL_H,VAL_L contain the validity byte                          ;
                                                          ;
                      00427 ; ON EXIT If MS byte is zero: Z set                                       ;
                      00428 ;         If MS byte is non zero: Z clear                                ;
                      00429 ;-------------------------------------------------------------------------;
                      00430
1FD1                  00431 VAL_READ
                      00432
1FD1   1703           00433         BSF     STATUS,RP1
1FD2   1283           00434 FR1     BCF     STATUS,RP0      ; Bank2
                      00435
1FD3   301F           00436         MOVLW   VALH            ; Address to flash address buffer
1FD4   008F           00437         MOVWF   EEADRH          ;
1FD5   30FF           00438         MOVLW   VALL            ;
1FD6   008D           00439         MOVWF   EEADR           ;
                      00440
1FD7   1683           00441         BSF     STATUS,RP0      ; Bank3
                      00442
1FD8   178C           00443         BSF     EECON1,EEPGD    ; The read sequence
1FD9   140C           00444         BSF     EECON1,RD       ;
1FDA   0000           00445         NOP                     ;
1FDB   0000           00446         NOP                     ;
                      00447
1FDC   1283           00448         BCF     STATUS,RP0      ; Bank2
1FDD   080E           00449         MOVFW   EEDATH          ; Read flash data
1FDE   1303           00450         BCF     STATUS,RP1      ; Bank 0
                      00451
1FDF   0008           00452         RETURN
                      00453
                      00454
                      00455 ;*************************************************************************;
                      00456 ; The validity word is initialised to invalid                             ;
                      00457
1FFF                  00458         ORG     VALID
1FFF   3FFF           00459         DW      0X3FFF
                      00460
                      00461         END
```

```
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXX------------ ---------------- ---------------- ----------------
0100 : X--------------- ---------------- ---------------- ----------------
1F00 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
1F40 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
1F80 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
1FC0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX ---------------- ---------------X
2000 : XXXX---X-------- ---------------- ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:   230
Program Memory Words Free:  7962


Errors   :     0
Warnings :     0 reported,     4 suppressed
Messages :     0 reported,    28 suppressed
```