



**AMICUS18®**

[www.protonbasic.co.uk](http://www.protonbasic.co.uk)



---

Powered by Proton Development Suite® Compiler of Crownhill Associates Limited©

## **A UNIVERSAL PDS BOOTLOADER**

---

### **CONFIGURING USER & TEST CODES: PART2**

---

PIC®, MPLAB®, PICkit3® and ICD3® are registered trademarks of Microchip Technology Inc®.  
Proton Development Suite® or PDS® are a registered trademark of Crownhill Associates Limited®.

The project has been developed and written by Alberto Freixanet.  
The document has been edited by John Drew.

#### **INTRODUCTION:**

The "AGV Bootloader" allows loading firmware via serial port (UART1, UART2 or more) without the need for dedicated host software. The bootloader is written using the "Proton Development Suite®". Modern microcontrollers have an interesting ability to write to their own memory and this feature opens up an entirely different programming method. A special program running on the microcontroller may communicate with an external device and receive data containing the desired program, which it then writes to itself.

To configure the Bootloader Templates, read the Manual PDS Bootloader Part1.

#### **General Features:**

1. The bootloader has been written with the Proton Basic Compiler Version 3.6.0.3. All routines are included in one BASIC file only.
2. It does not require a special download management PC application.
3. The Host computer program need only be simple RS232 communication software.
4. XON/XOFF handshaking is required.
5. The bootloader is as automatic as possible requiring little or no user coding.
6. Customized firmware for each user project.
7. Self-protects so that large user applications do not over-write the bootloader.
8. The bootloader code does not use the interrupt and so avoids some issues.
9. Successful downloads start user application automatically.
10. Only uses three wires for communication: TX, RX, ground and the Reset pin for starting.
11. All resources are released after download and available to the user application.

12. Highly adaptable/maintainable using the PDS® compiler.
13. Can be compiled for many combinations of crystal speeds and PIC® family members. (PIC18® ROM >= 128KB)
14. Many controls have been introduced in the code making for a very secure bootloader.
15. Several additional routines are available to debug the downloading file.
16. Several additional OPTIONS are available to the PDS® user.
17. Before downloading a Password access system is available to the user.

### Specific Features:

- To Write programs to devices with up to 128KB ROM, minus the size of the bootloader.
- To Write discontinued (with jumps in the ROM memory) or plain user code.
- To Write to Program FLASH, ID Locations and Data EEPROM.
- Download user code by using the UART1 or UART2, at 115200, 57600, 38400 Baud adjusted automatically to maximum speed according to the FOSC used.
- Wide range of usable FOSC: 64, 48, 40, 32, 25, 24, 20, 16, 14 (14.32MHz), 12, 10.
- FOSC = 80 MHz for the PIC18FxxK20 series.
- The internal oscillator of the PIC® is available for downloading to a maximum speed of 57600 Baud)
- The firmware uses a double UART buffer to increase speed and reliability.
- Self write the boot "Goto PIC#Loader#Boot" in position 0 of ROM. (HSM only)
- Can be compiled with all options of the Watch-dog Timer.
- To control the PIC® device ID before write.
- Writing the Config Fuses is not supported for security reasons.

### How to configure the User Code:

#### A - Low Side of Memory Bootloader: (LSM)

The firmware is always positioned at the **BOTTOM** of the ROM in every compilation, like a user program. Its code is automatically written in the boot section of the PIC® starting at the address 0.

#### The header:

Copy the header code of the bootloader from [**CONFIG01**] section.

You have to choose the same options as the bootloader written in your firmware.

```
' ' The USER must copy this "Declare" in the user main project.
Device = 18F25K22
Declare Xtal = 64

' ' DEFINE THE PLL: On / OFF
$define PLL_ConfigFuses On
```

```

'' Enable the line to use the internal Oscillator.
'' Uncomment the next line to enable the configuration.
'$define _InternalOSC_

'' Enable the Line to use the CLOCK OUT FUNCTION:
'' If CLKOUT function is enabled, CLKOUT on RA6 & Port function On RA7
   otherwise Port function On RA6 & RA7.
'$Define _CLKOUT_Function_

```

The UART define line is not necessary. Only write your code for the UART that has been chosen. (**HRSOut** or **HRSOut2**)

### The Declares:

```

$define _EnableProtectBootBlock_
Declare PROTON_START_ADDRESS = 2048
Declare Optimiser_Level = 2 (or 3)
Declare Dead_Code_Remove = On
Declare Watchdog = On
Declare Bootloader = OFF

```

### Calculate the Baud Rate of the terminal:

The baud rate of the terminal is automatic and based on the [**Declare Xtal**]. Copy this code. Of course the user Baud Rate could be different to the Bootloader Baud rate. In which case the terminal will receive all characters as trash unless the terminal baud rate is changed.

The baud rate of the terminal could be chosen manually. Uncomment one line only, otherwise the configuration is automatic.

```

' Declare the UART Baud Rate manually: enable ONE line ONLY.
'   $define _BaudRate 9600
'   $define _BaudRate 19200
'   $define _BaudRate 38400
'   $define _BaudRate 57600
'   $define _BaudRate 115200

$ifndef _BaudRate
    $if _xtal >= 40
        $ifdef _InternalOSC_
            $define _BaudRate 57600
        $else
            $define _BaudRate 115200
        $endif
    $endif
    $if (_xtal >= 16) And (_xtal < 40)
        $define _BaudRate 57600
    $endif
    $if _xtal < 16
        $define _BaudRate 38400
    $endif
$endif

```

## Declare the Baud Rate of the terminal:

You can choose the [**Hserial\_Baud** or **Hserial2\_Baud**] according to your option.

```
$if _BaudRate = 9600
    Declare Hserial_Baud = 9600
    Declare Hserial_TXSTA = 36
    Declare Hserial_RCSTA = 144
    Declare Hserial_Clear = On
$endif
'-----
$if _BaudRate = 19200
    Declare Hserial_Baud = 19200
    Declare Hserial_TXSTA = 36
    Declare Hserial_RCSTA = 144
    Declare Hserial_Clear = On
$endif
'-----
$if _BaudRate = 38400
    Declare Hserial_Baud = 38400
    Declare Hserial_TXSTA = 36
    Declare Hserial_RCSTA = 144
    Declare Hserial_Clear = On
$endif
'-----
$if _BaudRate = 57600
    Declare Hserial_Baud = 57600
    Declare Hserial_TXSTA = 36
    Declare Hserial_RCSTA = 144
    Declare Hserial_Clear = On
$endif
'-----
$if _BaudRate = 115200
    Declare Hserial_Baud = 115200
    Declare Hserial_TXSTA = 36
    Declare Hserial_RCSTA = 144
    Declare Hserial_Clear = On
$endif
```

## Always Copy the Config Fuses from the Bootloader Code:

```
Config_Start
$if PLL_ConfigFuses = On
    $ifdef _InternalOSC_
        $if (_xtal = 16) Or (_xtal = 32) Or (_xtal = 64)
            $ifdef _CLKOUT_Function_
                FOSC = INTIO7 ;Internal oscillator block, CLKOUT in RA6,
                             port function on RA7.
            $else
                FOSC = INTIO67 ;Internal oscillator block,
                               port function on RA6 and RA7.
            $endif
        $else
            $error "Error in (Declare Xtal) with internal oscillator
                    & PLL_ConfigFuses = On"
        $endif
    PRICLKEN = OFF ;Primary clock can be disabled by software.
```

```

$else
    $if (_xtal >= 10) And (_xtal <= 64)
        FOSC = HSMP           ;HS oscillator, PLL enabled
                               (Clock Frequency = 4 x FOSC1) (Xtal piece 2,5-16 MHz)
    $else
        $error "Error in (Declare Xtal) with external Xtal
                & PLL_ConfigFuses = On"
    $endif
    PRICKEN = On             ;Primary clock enabled.
$endif
PLLCFG = On                 ;Oscillator multiplied by 4
$endif
$if PLL_ConfigFuses = OFF
    $ifdef _InternalOSC_
        $if (_xtal = 16)
            $ifdef _CLKOUT_Function_
                FOSC = INTIO7      ;Internal oscillator block, CLKOUT in RA6,
                                   port function on RA7.
            $else
                FOSC = INTIO67     ;Internal oscillator block, port function on RA6
                                   and RA7.
            $endif
        $else
            $error "Error in (Declare Xtal) with internal oscillator
                    & PLL_ConfigFuses = OFF"
        $endif
        PRICKEN = OFF          ;Primary clock can be disabled by software.
    $else
        $if (_xtal >= 10) And (_xtal <= 16)
            FOSC = HSMP         ;HS oscillator (medium power 4-16 MHz)
        $else
            $if (_xtal > 16) And (_xtal <= 25)
                FOSC = HSHP      ; HS oscillator (high power > 16 MHz)
            $else
                $if (_xtal > 25) And (_xtal <= 64)
                    FOSC = ECHP  ; EC oscillator, CLKOUT function on OSC2
                                   (high power, >16 MHz)
                $else
                    $error "Error in (Declare Xtal) with external Xtal
                            & PLL_ConfigFuses = OFF"
                $endif
            $endif
        $endif
        PRICKEN = On          ;Primary clock enabled.
    $endif
    PLLCFG = OFF             ;Oscillator multiplied by 1
$endif
FCMEN = OFF                 ;Fail-Safe Clock Monitor disabled
IESO = OFF                  ;Oscillator Switchover mode disabled
PWRTEN = On                 ;Power up timer enabled
BOREN = SBORDIS             ;Brown-out Reset enabled in hardware only
                               and disabled in Sleep mode (SBORDIS is disabled)
BORV = 190                  ;VBOR set to 1.90 V nominal
WDTEN = SWON                ;WDT is controlled by the SWDTEN bit of the WDTCON register.
WDTPS = 1024                ;1:1024
CCP2MX = PORTC1             ;CCP2 input/output is multiplexed with RC1
PBADEN = Off                ;PORTB<5:0> pins are configured as Digital on Reset
CCP3MX = PORTC6             ;P3A/CCP3 input/output is multiplexed with RC6
HFOFST = OFF                ;HFINTOSC output and ready status are delayed
                               by the oscillator stable status
T3CMX = PORTC0              ;T3CKI is on RC0
P2BMX = PORTB5              ;P2B is on RB5

```

```

MCLRE = EXTMCCLR      ;MCLR pin enabled, RE3 input pin disabled
STVREN = OFF          ;Stack full/underflow will not cause Reset
LVP = OFF             ;Single-Supply ICSP disabled
XINST = OFF           ;Instruction set extension and Indexed Addressing mode
                        disabled (Legacy mode)
Debug = OFF           ;Disabled
Cp0 = OFF             ;Block 0 (000800-001FFFh) not code-protected
CP1 = OFF             ;Block 1 (002000-003FFFh) not code-protected
CP2 = OFF             ;Block 2 (004000-005FFFh) not code-protected
CP3 = OFF             ;Block 3 (006000-007FFFh) not code-protected
$ifdef _EnableProtectBootBlock_
CPB = On              ;Boot block (000000-0007FFh) code-protected
$else
CPB = OFF             ;Boot block (000000-0007FFh) not code-protected
$endif
CPD = On              ;Data EEPROM code-protected
WRT0 = OFF            ;Block 0 (000800-001FFFh) not write-protected
WRT1 = OFF            ;Block 1 (002000-003FFFh) not write-protected
WRT2 = OFF            ;Block 2 (004000-005FFFh) not write-protected
WRT3 = OFF            ;Block 3 (006000-007FFFh) not write-protected
WRTC = On            ;Configuration registers (300000-3000FFh) write-protected
$ifdef _EnableProtectBootBlock_
WRTB = On             ;Boot Block (000000-0007FFh) write-protected
$else
WRTB = OFF            ;Boot Block (000000-0007FFh) not write-protected
$endif
WRD = OFF            ;Data EEPROM not write-protected
EBTR0 = OFF          ;Block 0 (000800-001FFFh) not protected from table reads
                        executed in other blocks
EBTR1 = OFF          ;Block 1 (002000-003FFFh) not protected from table reads
                        executed in other blocks
EBTR2 = OFF          ;Block 2 (004000-005FFFh) not protected from table reads
                        executed in other blocks
EBTR3 = OFF          ;Block 3 (006000-007FFFh) not protected from table reads
                        executed in other blocks
$ifdef _EnableProtectBootBlock_
EBTRB = On           ;Boot Block (000000-0007FFh) protected from table reads
                        executed in other blocks
$else
EBTRB = OFF          ;Boot Block (000000-0007FFh) not protected from table reads
                        executed in other blocks
$endif
Config_End

```

### Copy the Internal Oscillator Code (if used):

Copy the Internal Oscillator Code from the [CONFIG17]. Example for the PIC18F25K22.

```

$ifdef _InternalOSC_
INT_OSC:
    OSCCON2 = 0
    $if PLL_ConfigFuses = On
        $if _xtal = 64
            OSCCON = %01111000
            OSTUNEbits_PLEN = 1
        $endif
        $if _xtal = 32
            OSCCON = %01101000
            OSTUNEbits_PLEN = 1
        $endif
        $if _xtal = 16

```

```

        OSCCON = %01011000
        OSTUNEbits_PLEN = 1
    $endif
$endif
$if PLL_ConfigFuses = OFF
    $if _xtal = 16
        OSCCON = %01111000
        OSTUNEbits_PLEN = 0
    $endif
$endif
OSTUNEbits_INTSRC = 1
$ifdef OSCCONbits_IOFS
While OSCCONbits_IOFS = 0
    #ifdef WatchDog_Req
    Clrwdt
    #endif
Wend
$else
    $ifdef OSCCONbits_HFIOFS
    While OSCCONbits_HFIOFS = 0
        #ifdef WatchDog_Req
        Clrwdt
        #endif
    Wend
    $endif
$endif
$else
    OSTUNEbits_INTSRC = 1
    OSCCON2 = 0
    OSCCON2bits_PRISD = 1
$endif

```

### End of Configuration of the user code:

Now the user can insert the library files and write the main code.

## B - High Side of Memory Bootloader: (HSM)

The firmware is always positioned dynamically at the **TOP** of the ROM in every compilation. Its size and position (*PIC#Loader#Boot = label*) is automatically calculated according of the choice of the number of user options. The firmware does not need an external program to write or/and compile the boot.

### The header:

Copy the header code of the bootloader from [**CONFIG01**] section.

You have to choose the same options as the bootloader as written in your firmware.

```

'' The USER must copy this "Declare" in the user main project.
Device = 18F25K22
Declare Xtal = 64

'' DEFINE THE PLL: On / OFF
$define PLL_ConfigFuses On

'' Enable the to use the internal Oscillator.

```

```

'' Uncomment the next line to enable the configuration.
'$define _InternalOSC_

'' Enable the Line to use the CLOCK OUT FUNCTION:
'' If CLKOUT function is enabled, CLKOUT on RA6 & Port function On RA7
   otherwise Port function On RA6 & RA7.
'$Define _CLKOUT_Function_

```

The UART define line is not necessary. Only write your code for the UART that has been chosen. (**HRSOut** or **HRSOut2**)

### The Declares:

```

Declare Optimiser_Level = 2 (or 3)
Declare Dead_Code_Remove = On
Declare Watchdog = On
Declare Bootloader = OFF

```

### Calculate the Baud Rate of the terminal:

The baud rate of the terminal is automatic and calculated based on the [Declare Xtal]. Copy this code. Of course the user Baud Rate could be different from the Bootloader Baud rate. . In which case the terminal will receive all characters as trash unless the terminal baud rate is changed.

The baud rate of the terminal could be chosen manually. Uncomment one line only, otherwise the configuration is automatic.

```

' Declare the UART Baud Rate manually: enable ONE line ONLY.
'   $define _BaudRate 9600
'   $define _BaudRate 19200
'   $define _BaudRate 38400
'   $define _BaudRate 57600
'   $define _BaudRate 115200

$ifndef _BaudRate
    $if _xtal >= 40
        $ifdef _InternalOSC_
            $define _BaudRate 57600
        $else
            $define _BaudRate 115200
        $endif
    $endif
    $if (_xtal >= 16) And (_xtal < 40)
        $define _BaudRate 57600
    $endif
    $if _xtal < 16
        $define _BaudRate 38400
    $endif
$endif

```

### Declare the Baud Rate of the terminal:

You can choose the [**Hserial\_Baud** or **Hserial2\_Baud**] according to your option.



```

$if _BaudRate = 9600
    Declare Hserial_Baud = 9600
    Declare Hserial_TXSTA = 36
    Declare Hserial_RCSTA = 144
    Declare Hserial_Clear = On
$endif
'-----
$if _BaudRate = 19200
    Declare Hserial_Baud = 19200
    Declare Hserial_TXSTA = 36
    Declare Hserial_RCSTA = 144
    Declare Hserial_Clear = On
$endif
'-----
$if _BaudRate = 38400
    Declare Hserial_Baud = 38400
    Declare Hserial_TXSTA = 36
    Declare Hserial_RCSTA = 144
    Declare Hserial_Clear = On
$endif
'-----
$if _BaudRate = 57600
    Declare Hserial_Baud = 57600
    Declare Hserial_TXSTA = 36
    Declare Hserial_RCSTA = 144
    Declare Hserial_Clear = On
$endif
'-----
$if _BaudRate = 115200
    Declare Hserial_Baud = 115200
    Declare Hserial_TXSTA = 36
    Declare Hserial_RCSTA = 144
    Declare Hserial_Clear = On
$endif

```

### Copy the Config Fuses from the Bootloader Code (always):

```

Config_Start
$if PLL_ConfigFuses = On
    $ifdef _InternalOSC_
        $if (_xtal = 16) Or (_xtal = 32) Or (_xtal = 64)
            $ifdef _CLKOUT_Function_
                FOSC = INTIO7      ;Internal oscillator block, CLKOUT in RA6,
                                   port function on RA7.
            $else
                FOSC = INTIO67     ;Internal oscillator block,
                                   port function on RA6 and RA7.
            $endif
        $else
            $error "Error in (Declare Xtal) with internal oscillator
                    & PLL_ConfigFuses = On"
        $endif
        PRICLKEN = OFF ;Primary clock can be disabled by software.
    $else
        $if (_xtal >= 10) And (_xtal <= 64)
            FOSC = HSMP           ;HS oscillator, PLL enabled
                                   (Clock Frequency = 4 x FOSC1) (Xtal piece 2,5-16 MHz)
        $else

```

```

    $error "Error in (Declare Xtal) with external Xtal & PLL_ConfigFuses =
On"

$endif
PRICLKEN = On    ;Primary clock enabled.
$endif
PLLCFG = On      ;Oscillator multiplied by 4
$endif
$if PLL_ConfigFuses = OFF
$ifdef _InternalOSC_
    $if (_xtal = 16)
        $ifdef _CLKOUT_Function_
            FOSC = INTIO7    ;Internal oscillator block,
                             CLKOUT in RA6, port function on RA7.

        $else
            FOSC = INTIO67   ;Internal oscillator block,
                             port function on RA6 and RA7.

        $endif
    $else
        $error "Error in (Declare Xtal) with internal oscillator
& PLL_ConfigFuses = OFF"

    $endif
    PRICLKEN = OFF    ;Primary clock can be disabled by software.
$else
    $if (_xtal >= 10) And (_xtal <= 16)
        FOSC = HSMF      ;HS oscillator (medium power 4-16 MHz)
    $else
        $if (_xtal > 16) And (_xtal <= 25)
            FOSC = HSHF    ; HS oscillator (high power > 16 MHz)
        $else
            $if (_xtal > 25) And (_xtal <= 64)
                FOSC = ECHF ; EC oscillator, CLKOUT function on OSC2
                             (high power, >16 MHz)
            $else
                $error "Error in (Declare Xtal)
with external Xtal & PLL_ConfigFuses = OFF"

            $endif
        $endif
    $endif
    PRICLKEN = On    ;Primary clock enabled.
$endif
PLLCFG = OFF        ;Oscillator multiplied by 1
$endif
FCMEN = Off         ;Fail-Safe Clock Monitor disabled
IESO = Off          ;Oscillator Switchover mode disabled
PWRTEN = On         ;Power up timer enabled
BOREN = On          ;Brown-out Reset enabled and controlled by software
                     (SBOREN is enabled)
BORV = 190          ;VBOR set to 1.90 V nominal
WDTEN = SWON        ' WDT is controlled by SWDTEN Bit of the WDTCN register
WDTPS = 1024        ;1:1024
CCP2MX = PORTC1     ;CCP2 input/output is multiplexed with RC1
PBADEN = On         ' PORTB<4:0> pins are configured as analog I/O on Reset
CCP3MX = PORTB5     ;P3A/CCP3 input/output is multiplexed with RB5
HFOFST = Off        ;HFINTOSC output and ready status are delayed
                     by the oscillator stable status
T3CMX = PORTC0      ;T3CKI is on RC0
P2BMX = PORTB5      ;P2B is on RB5
MCLRE = EXTMCLR     ;MCLR pin enabled, RE3 input pin disabled
STVREN = On         ;Stack full/underflow will cause Reset
LVP = Off           ;Single-Supply ICSP disabled
XINST = Off         ;Instruction set extension and Indexed Addressing
                     mode disabled (Legacy mode)

```

```

Debug = OFF           ;Disabled
Cp0 = OFF             ;Block 0 (000800-001FFFh) not code-protected
CP1 = OFF             ;Block 1 (002000-003FFFh) not code-protected
CPB = OFF             ;Boot block (000000-0007FFh) not code-protected
CPD = On              ;Data EEPROM code-protected
WRT0 = OFF            ;Block 0 (000800-001FFFh) not write-protected
WRT1 = OFF            ;Block 1 (002000-003FFFh) not write-protected
WRTC = On             ;Configuration registers (300000-3000FFh) write-protected
WRTB = OFF            ;Boot Block (000000-0007FFh) not write-protected
WRTD = OFF            ;Data EEPROM not write-protected
EBTR0 = OFF           ;Block 0 (000800-001FFFh) not protected
                        ;from table reads executed in other blocks
EBTR1 = OFF           ;Block 1 (002000-003FFFh) not protected
                        ;from table reads executed in other blocks
EBTRB = OFF           ;Boot Block (000000-0007FFh) not protected
                        ;from table reads executed in other blocks

```

**Config\_End**

## Copy the Internal Oscillator Code (if used):

Copy the Internal Oscillator Code from the [CONFIG17]. Example for the PIC18F25K22.

```

#ifdef _InternalOSC_
INT_OSC:
    OSCCON2 = 0
    #if PLL_ConfigFuses = On
        #if _xtal = 64
            OSCCON = %01111000
            OSTUNEbits_PLEN = 1
        #endif
        #if _xtal = 32
            OSCCON = %01101000
            OSTUNEbits_PLEN = 1
        #endif
        #if _xtal = 16
            OSCCON = %01011000
            OSTUNEbits_PLEN = 1
        #endif
    #endif
    #if PLL_ConfigFuses = OFF
        #if _xtal = 16
            OSCCON = %01111000
            OSTUNEbits_PLEN = 0
        #endif
    #endif
    OSTUNEbits_INTSRC = 1
    #ifdef OSCCONbits_IOFS
    While OSCCONbits_IOFS = 0
        #ifdef WatchDog_Req
        Clrwdt
        #endif
    Wend
    #else
        #ifdef OSCCONbits_HFIOFS
        While OSCCONbits_HFIOFS = 0
            #ifdef WatchDog_Req
            Clrwdt
            #endif
        Wend
    #endif

```

```

        $endif
    $endif
$else
    OSCTUNEbits_INTSRC = 1
    OSCCON2 = 0
    OSCCON2bits_PRISD = 1
$endif

```

## End of Configuration of the user code:

Now the user can insert the library files and write the main code.

## How to use the Test Code:

Some test codes are included in the Bootloader folders. These files correspond to PIC@s whose bootloader code has been tested with a real part.

I have followed exactly the construction described in the previous chapters, which allows changing the configuration of the test code according to the options of the bootloader that the user has chosen.

## What is the function of the test code?

### Light a LED:

The test code must be simple but it must also give quality information about the operation of the PIC@.

The test code blinks a LED. The user can choose any pin of his test board, according to the following code.

```

'=====
' Define the PORT & PIN of the test LED.
'
    $define _PORT PORTC
    $define _PIN 5
'=====

```

The LED flashes at a rate defined by **Timer0** and an interrupt [**On\_Low\_Interrupt**]. The frequency may vary a bit depending on the **FOSC** chosen by the user.

### Reading Edata Strings:

Some strings have been written in the eeprom memory to test the Eread command.

```

E2p_AddressString1 EData "Testing the PIC18 AGV Bootloader V5.0 written in
High Side of the ROM.",0

```

```

E2p_AddressString2 EData "New Universal PIC18 Bootloader by Alberto
Freixanet. ",0

```

```

E2p_AddressString3 EData "A 8 digits Password could be needed to download
the user code. ",0

```

The routine of reading these strings checks that the bootloader has correctly written this data.

### Reading parameters of the asm file:

There is no command in the PDS compiler to read some parameters of the asm file.

However, you can use code in ASM format, for example to read the WDT parameter in the memory map configured by the Config Fuses.

```
' Read the WDT config in memory map from asm file.
EECON1 = %10000000      ' Access Flash memory
Set_Bank TBLPTRL
Asm
Movlw Low (Config2H)
Movwf TBLPTRL
Movlw High (Config2H)
Movwf TBLPTRH
Movlw upper (Config2H)
Movwf TBLPTRU
EndAsm
Tblrd*                  ' Read one memory position.
Temp1 = TABLAT
EECON1 = 0              ' Disable Access to the Configuration registers
TBLPTRU = 0             ' The HRSOut command of the compiler does not
                        ' clear the "TBLPTRU = 0", then take care of that.
```

At the end you must delete the **TBLPTRU** register for the **HRSOut** command to work correctly.

The DeviceList.inc library:

I have developed a library allowing a user to look up the name of the PIC® depending on the **DEVICE\_ID** read from the ROM. This code could be very useful for PDS® users.

Library piece of code:

```
$if _device = _18F25K22
$define __DEVICE_ID 682
$define __DEVICE_NAME "PIC18F25K22"
$endif
```

For new PICs, the user could add a new code.

How to use:

The **DEVICE\_ID** of the ROM is read and compared with the list in the library, it is very simple. But it can only be applied to very standard PICs. Some new PICs change the way to read the **DEVID1** parameter. In this case, a different calculation of the [**PICPartNumber**] and [**PICRevision**] must be performed. See the manual of each PIC® or an example of the Bootloader for PIC18FxxK40.

```
' Read the DEVID1 (Device ID) from the asm file.
EECON1 = %10000000      ' Access Flash memory
Set_Bank TBLPTRL
Asm
Movlw Low (DEVID1)
Movwf TBLPTRL
Movlw High (DEVID1)
Movwf TBLPTRH
Movlw upper (DEVID1)
Movwf TBLPTRU
EndAsm
```

```

Tblrd**                                ' perform table read with post-increment
DataWord.LowByte = TABLAT
Tblrd**                                ' perform table read with post-increment
DataWord.HighByte = TABLAT
TBLPTRU = 0                            ' Cleared for security
EECON1 = 0                             ' Disable Access Configuration registers
'-----
' Calculate the PIC Part Number & revision
PICPartNumber = DataWord >> 5
PICRevision = DataWord.LowByte & 31
'-----
' Print the Device Name to the Terminal.
$ifndef __DEVICE_ID
    $error "The Devicelist.inc is missing!!"
$endif
Select PICPartNumber
    Case __DEVICE_ID
        HRSOut "Device = ", __DEVICE_NAME, CR, LF
    Case Else
        HRSOut CR, LF
        HRSOut "The PIC Device number read from the Board is not
                correct!", CR, LF
        HRSOut CR, LF
    EndSelect

```

### The IDLOCS:

If the IDLOCS option of the Bootloader has been activated, some parameters could be added to the end of the file.bas, for example.

```

Asm
__IDLOCS    IDLOC0, "L"
__IDLOCS    IDLOC1, "O"
__IDLOCS    IDLOC2, "A"
__IDLOCS    IDLOC3, "D"
__IDLOCS    IDLOC4, "E"
__IDLOCS    IDLOC5, "R"
__IDLOCS    IDLOC6, "5"
__IDLOCS    IDLOC7, "0"
EndAsm

```

### CONCLUSION

I hope that these examples of unusual code can help you in your projects. Any comments would be welcome to improve the PDS Bootloader.

PDS Bootloader version 5.0. 30 December 2018

Alberto Freixanet

