

**ds30 Loader  
Firmware manual**

## Table of contents

Table of contents.....	2
Introduction.....	4
ds30 Loader.....	4
Development environment.....	4
Tool suite requirement.....	4
Flash space requirement.....	4
PIC12F & PIC16F.....	4
PIC18F.....	4
PIC18FJ.....	4
PIC24FK & dsPIC30F.....	4
PIC24H & dsPIC33F.....	5
PIC24FJ.....	5
PIC24E & dsPIC33E.....	5
PIC32MX.....	5
PIC32MZ.....	5
Trademarks.....	5
Supported communication.....	6
UART.....	6
Software UART.....	6
CAN.....	6
I 2C.....	6
SPI.....	6
SD.....	6
USB.....	6
The basics.....	7
Assembler version.....	7
The MPLAB IDE project.....	7
Boot loader placement.....	7
C version.....	8
The MPLAB IDE project.....	8
Operation.....	10
Initialization.....	10
Wait for hello command from the host application.....	10
Main loop, command handling.....	10
Erase page command.....	10
Write row command.....	10
Write eeprom word command.....	10
Communication timeout.....	10
Usage.....	11
0. Errata's.....	11
1. Select device.....	11
2. Configure boot loader settings.....	11

---

3.1 Code, asm version.....	15
3.1 Code, C version.....	15
3.2 MPLAB IDE.....	15
4.1 Analog pins.....	16
4.2 PPS.....	16
4.3 Oscillator.....	16
5. Linker script.....	16
5.1 PIC32.....	16
6. Build.....	16
7. Erase device.....	18
7.1 Programmer supported by MPLAB IDE.....	18
7.2 Programmer not supported by MPLAB IDE.....	18
8. Write boot loader to PIC.....	18
8.1 Programmer supported by MPLAB IDE.....	18
8.2 Programmer not supported by MPLAB IDE.....	18
9. Adapt the user application.....	18
9.1 PIC12 and PIC16.....	18
9.2 PIC18.....	18
9.3 PIC24F and dsPIC.....	19
9.4 PIC32.....	19
Known limitations.....	20
PIC18.....	20
Watchdog Time-out Flag bit.....	20
Considerations.....	21
Code protection.....	21
Data stored in flash memory.....	21
Linker script.....	21
Oscillator.....	21
Using different oscillator settings for boot loader and application.....	21
Unplanned download of different oscillator setup.....	21
Interrupts.....	22
PIC18 extended instruction set.....	22
PPS.....	22
Register default values.....	22
Watchdog.....	22
Appendix A – Links.....	23

## Introduction

### **ds30 Loader**

ds30 Loader is a boot loader supporting PIC12F, PIC16, PIC18, PIC24, dsPIC, and PIC32 families of MCUs from Microchip. It supports all devices that supports RTSP(run time self programming) and has enough flash and RAM memory. The firmware is written in assembler or C. The host applications run on Windows, Linux, and Mac OS X.

### **Development environment**

The firmware is delivered with both MPLAB IDE and MPLAB X IDE projects.

### **Tool suite requirement**

	Assembler	C
PIC12F	MPASM	n/a
PIC16F	MPASM	MPLAB XC8
PIC18F	MPASM	MPLAB XC8
PIC24F / H dsPIC30F, dsPIC33F	MPLAB ASM30	MPLAB XC16
PIC24E , dsPIC33E	n/a	MPLAB XC16
PIC32MX / MZ	n/a	MPLAB XC32

### **Flash space requirement**

The size and placement information may change without notification. Please contact DigSol Sweden AB to get the current values.

#### **PIC12F & PIC16F**

	Size		Placement
	Standard	Secure	
UART	0x400	0xA40	End of flash

#### **PIC18F**

	Size		Placement
	Standard	Secure	
UART	0x800	0x1380	End of flash
CAN	0x980	0x1500	
USB	0xF00	TBD	

#### **PIC18FJ**

	Size		Placement
	Standard	Secure	
UART	0x800	0x1400	Second last page
USB	TBD	TBD	

#### **PIC24FK & dsPIC30F**

	Size	Placement
--	------	-----------

	Standard	Secure	
UART CAN	0x800	0xC00	End of flash
USB	TBD	TBD	

### PIC24H & dsPIC33F

	Size		Placement
	Standard	Secure	
UART CAN	0x800	0xC00	End of flash
USB	0x1800	0x1C00	

### PIC24FJ

	Size		Placement
	Standard	Secure	
UART CAN	0x800	0xC00	Second last page and backwards
USB	0x1800	0x1C00	

### PIC24E & dsPIC33E

	Size		Placement
	Standard	Secure	
UART CAN	0x800	<=0x1000	End of main flash
USB	0x1800	<=0x2000	

### PIC32MX

	Size		Placement
	Standard	Secure	
UART CAN	<=0x1000	<=0x2000	End of main flash
USB	<=0x5000	<=0x5000	

### PIC32MZ

	Size		Placement
	Standard	Secure	
UART CAN	TBD	TBD	TBD
USB	TBD	TBD	

### Trademarks

All rights to copyrights, registered trademarks, and trademarks reside with their respective owners.

## Supported communication

ds30 Loader supports several different ways of communication:

### **UART**

The UART boot loader has the following features:

- Selectable baud rate
- Supports alternate i/o (PIC24F and dsPIC30F)
- Auto baud rate detection
- Tx enable pin for RS-485 operation
- Available for PIC12, PIC16, PIC18, PIC24, dsPIC, and PIC32
- Written in assembler or C

### **Software UART**

The software UART boot loader has the following features:

- Selectable baud rate
- Available for PIC12, PIC16, PIC18, PIC24, and dsPIC
- Written in assembler
- Not available in the free edition

### **CAN**

The CAN boot loader has the following features:

- Easy to configure bit rate
- Selectable id
- Selectable mask
- Standard/extended frames
- Available for PIC18, PIC24H/E, dsPIC30F/33F , and PIC32
- Written in assembler or C
- Not available in the free edition

### **I<sup>2</sup>C**

See ds30 HEX Loader.

### **SPI**

See ds30 HEX Loader.

### **SD**

See ds30 SD Card Loader.

### **USB**

The USB boot loader has the following features:

- Based on the Microchip USB stack

## The basics

### **Assembler version**

#### **The MPLAB IDE project**

The firmware MPLAB IDE project consists of several files:

##### **ds30Loader.asm / ds30Loader.s**

This is the main file that contains all firmware code (assembler instructions). Normally no changes need to be done in this file.

##### **can.inc**

This file contains CAN functionality. This file is only available in the commercial version.

##### **devices.inc**

This file contains device specific constants such as size of the eeprom memory and the number of UARTs available.

##### **i2c.inc**

This file contains I2C functionality. This file is only available in the commercial version.

##### **settings.inc**

This file contains all common user customizations such as communication module assignment, baud rate, device and more. This file needs to be modified in order to make the boot loader work for each different hardware setup.

##### **uart.inc**

This file contains UART functionality.

##### **uart\_soft.inc**

This file contains software UART functionality. This file is only available in the commercial version.

##### **user\_code.inc**

This file holds user code that should execute before and after boot loader operation.

##### **xxx.lkr / xxx.gld**

This is the device specific linker script need by the linker. This does not come with the ds30 Loader; it comes with the Microchip language tool suite.

### **Boot loader placement**

The boot loader is normally placed at the very end of flash memory. This way there is usually no need to reserve space for the boot loader in the linker script. In some device families the configuration words are located at the end of the flash memory. For those families the boot loader is placed in the second last page.

	<b>Size</b>	<b>Placement</b>
<b>PIC12F</b>	256 words*	end of memory
<b>PIC16F</b>	256 words*	end of memory
<b>PIC18F</b>	7 pages*	end of memory
<b>PIC18FJ</b>	1 page	2 <sup>nd</sup> last page
<b>PIC24F</b>	4 rows*	end of memory

<b>PIC24FJ</b>	1 page	2 <sup>nd</sup> last page
<b>PIC24H</b>	1 page	end of memory
<b>PIC24E</b>	1 page	end of memory
<b>dsPIC30F</b>	8 rows*	end of memory
<b>dsPIC33FJ</b>	1 page	end of memory
<b>dsPIC33E</b>	1 page	end of memory
<b>PIC32</b>	1 page	end of memory

\* May differ for different firmware versions

## **C version**

The C version is not available in the free edition.

## **The MPLAB IDE project**

The firmware MPLAB IDE project consists of several files:

### **board\_xxx.c**

This file contains the `board_init()` function that is called from `main()` on startup. The user is responsible to modify it with code to setup the board. Oscillator, I/O, PPS etc. This file is also suitable for storage of the configuration bits.

### **can\_xxx.c**

This file contains CAN functionality. It should usually not be modified.

### **main.c**

This file contains the `main()` function which is the entry point and from which the board and communication initialization routines are called. It may be modified but it is usually not needed.

The `ds30_activity()` function is called from the receive loop. The user may place code here such as kick of the watchdog. For heavier tasks it is advised to use a timer so that the heavy code is not executed for each call.

The `ds30_exit()` function is called just before the user application loaded. The user may place clean-up code here.

### **uart\_xxx.c**

This file contains UART functionality. It should usually not be modified.

### **board.h**

This file is included by various source files. It in turn includes the board specific file. This file may need to be modified.

### **board\_xxx.h**

This file contains all common user settings such as communication module assignment, baud rate, and timing. This file needs to be modified in order to make the boot loader work for each different hardware setup.

### **comm.h**

This file is included by various source files. It should usually not be modified.

### **ds30loader.h**

This file contains the `ds30_main` function prototypes. It should usually not be modified.

### **ds30\_Loader\_xxx.a**

This is the ds30 Loader engine library file.



**xxx.gld**

This is the device specific linker script, it has been modified to give the boot loader a specific placement in the flash.

## Operation

### ***Initialization***

The communication module is initialized.

### ***Wait for hello command from the host application***

The boot loader waits for a command that starts the main boot loader loop. If the hello command is not received within the configured timeout the boot loader branches to the application. If no application exists; the boot loader will perform a software reset.

### ***Main loop, command handling***

The boot loader receives read/erase/write commands from the host application. For each command the packet checksum is verified.

### **Erase page command**

This command sets a flag and the subsequent write row command performs the actual erase.

### **Write row command**

After writing the row the written data is verified.

### **Write eeprom word command**

After writing the word the written data is verified.

### ***Communication timeout***

When communication timeout eventually occurs; the boot loader branches to the application. If no application exist the boot loader resets.

## Usage

Start by opening the firmware MPLAB IDE project located in the firmware xxx directory.

### **0. Errata's**

No device specific errata workarounds are implemented. The user must read the device errata sheet carefully to make sure there are no problems that could interfere with boot loader operation.

### **1. Select device**

Select correct device on the menu *Configure->Select Device...*

### **2. Configure boot loader settings**

Most if not all settings are located in the file settings.inc (assembler firmwares) or board\_XXX.h (C firmwares). All lines commented with xxx needs to be verified/changed. Not all settings are available in any firmware. Here follows a description of all available settings.

#### **DEV\_MODE**

Used during development, delete or comment this line.

```
.equ    __30F4011, 1
```

Set to your device name. This setting is only valid for the PIC24 and dsPIC firmware.

```
LIST    P=18F2550
```

Set to your device name. This setting is only valid for the PIC12, PIC16, and PIC18 firmwares.

#### **FCY**

Set to instruction cycle clock speed (nr of instructions per second). This is only a constant it does not setup any oscillator settings such as PLL, which has to be done manually. This setting is only valid for the PIC24 and dsPIC firmware.

#### **OSCF / FOSC**

Set to oscillator frequency. This is only a constant it does not setup any oscillator settings such as PLL, which has to be done manually. This setting is only valid for the PIC12, PIC16, and PIC18 firmwares.

#### **BLINIT / HELLOTIMEOUT**

This is the receive timeout in milliseconds for the first hello command sent from the PC client. Decreasing the value means the application will start sooner on power-up. There is an upper limit which depends on the oscillator frequency.

#### **HELLOTRIES**

This is how many non-hello commands that are discarded on start-up before the boot loader is aborted and the user application is loaded. If a reset command this should be set to the

length of the reset command + 1. If no reset command is used this should be set to a low value higher than 0.

#### **BLTIME / TIMEOUT**

This is the communication receive timeout in milliseconds.

#### **USE\_UARTx**

Uncomment the line matching the uart you are using.

#### **USE\_ALTIO**

Uncomment to use alternative I/O for UART1. This setting is only valid for dsPIC30F devices. More information about the USE\_ALTIO setting is available in the device datasheet.

#### **BAUDRATE**

Set to the desired UART baud rate, the brg value is automatically calculated. If the error of the chosen baud rate exceeds 2.5% an error message will be displayed when assembling.

#### **USE\_ABAUD**

Uncomment to use auto baud rate detection. Please read errata first to make sure there are no problems when using auto baud rate detection.

#### **USE\_BRG16**

Uncomment to use 16-bit baud rate register. Please read errata first to make sure there are no problems when using BRG16=1. More information about the BRG16 settings is available in the device datasheet.

#### **USE\_BRGH**

Uncomment to use high baud rates. Please read errata first to make sure there is no problems when using BRGH=1. More information about the BRGH settings is available in the device datasheet.

#### **USE\_TXENABLE**

Uncomment to use a transmit enable pin allowing RS485 communication.

#### **TXE\_DELAY**

Time in  $\mu$ s to wait before transmitting after pulling the tx enable pin high.

#### **TRISR\_TXE**

Set to tris register of transmit enable pin.

#### **LATR\_TXE**

Set to lat register of transmit enable pin.

#### **TRISB\_TXE**

Set to bit in tris register of transmit enable pin.

**LATB\_TXE**

Set to bit in lat register of transmit enable pin.

**USE\_SWUART**

Uncomment to use the software uart. This option is only available in the commercial version.

**BITWAITCNT**

This constant gives the baud rate. Calculate it according to the formula available in the file settings.inc.

**TRISR\_TX, TRISB\_TX, LATR\_TX, LATB\_TX**

Configuration registers for the transmit pin.

**TRISR\_RX, TRISB\_RX, PORTR\_RX, PORTB\_RX**

Configuration registers for the receive pin.

**USE\_CANx**

Uncomment to select CAN controller. This option is only available in the commercial version.

**ID\_PIC**

CAN id/node number for this unit.

**ID\_GUI**

CAN id/node number of the ds30 Loader host application.

**ID\_MASK**

This setting is used to mask bits in the id of received frames. Read more about masking in the CAN section of the device datasheet.

**CAN\_EXT**

Uncomment to send and receive extended data frames. Read more about extended frames in the CAN section of the device datasheet.

**CAN\_BRP, CAN\_PROP, CAN\_SEG1, CAN\_SEG2, CAN\_SJW, CAN\_CKS**

CAN timing settings. These need to be manually calculated. The "CAN timing calculator.xls" spreadsheet may be of help. Detailed information is found in the device datasheet.

**TRISR\_CRX**

On PIC18 the CAN receive pins to explicitly be set to input. Set to tris register of CAN receive pin.

Example: TRISE

**TRISB\_CRX**

Set to bit in tris register of CAN receive pin.

Example: TRISE4

**TX\_TIMEOUT**

CAN transmit timeout in milliseconds.

**USE\_I2Cx**

Uncomment to select I2C bus controller

**ADDR\_PIC**

The 7-bit slave device id for this unit on the I<sup>2</sup>C bus.

**ADDR\_MASK**

The 7-bit id mask. Bit=1 => ignore. Usually set to 0x00. Not available for dsPIC30F.

**KICK\_WD**

If the watchdog is enablea. Uncomment this line to enable kick of the watchdog in the receive loop. If the watchdog is not enabled this line must be disabled.

**USE\_READ**

Uncomment to enable read of flash and eeprom contents. This option is only available in the commercial version.

**PROT\_GOTO**

Comment to disable protection of the goto at 0x00. It is recommended to not disable goto protection. If the goto gets corrupted the boot loader will not be called on start-up.

**PROT\_BL**

Comment to disable boot loader protection. It is not recommended to disable boot loader protection.

**BLPL**

Placement of the boot loader in the PIC flash memory, pages/rows from the end. When this is changed by the user the custom bootloader option ds30 Loader GUI must be enabled and the new values entered in the textboxes.

**BLSIZE**

The size of the boot loader, it is used by boot loader protection. When this is changed by the user the custom bootloader option ds30 Loader GUI must be enabled and the new values entered in the textboxes.

**config xxx**

See the next section.

### **3. Configuration bits**

Setting the configuration bits is a required and vital step to make the boot loader work. Also see the oscillator considerations chapter later in this manual. Information about the configuration bits is found in the device datasheet. The configuration bits can be set in either MPLAB IDE or in the code.

### 3.1 Code, asm version

There is a template to use in the last section of the file settings.inc. Make sure to check the checkbox labeled "Configuration Bits set in code" on the menu "Configure->Configuration bits...". All available can be found at the end of the devices include file. Default include file locations:

PIC12, PIC16, and PIC18: c:\Program Files\Microchip\MPASM Suite\

PIC24 and dsPIC: c:\Program files\Microchip\MPLAB ASM30 Suite\Support\family\inc

For PIC18 there is also documentation available in MPLAB IDE. Click menu Help->Topics then choose "PIC18 Config Settings" and click OK.

### 3.1 Code, C version

There is a template in the board\_XXX.c file.

### 3.2 MPLAB IDE

The configurations are found on the menu "Configure->Configuration bits...". Make sure to uncheck the checkbox labeled "Configuration Bits set in code".

## 4. Add own initialization code

If needed, add initialization and/or exit code in user\_code.inc / board\_XXX.c. In some firmwares, the space available for user code is restricted to a few instructions. See table below for details. The exact number depends on firmware version and which features are enabled.

On PIC12 and PIC16 devices, do not forget to the correct bank for each register access. Use the BANKSEL macro.

If more space is need the boot loader size and placement needs to be changed in settings.inc. In the GUI you need to check custom boot loader under the advanced tab and enter the details of the new boot loader properties.

	<b>Words free to use for user code (varies for different fw versions)</b>
<b>PIC16F</b>	~15
<b>PIC18F</b>	~30
<b>PIC18FJ</b>	>100
<b>PIC24F</b>	~10
<b>PIC24FJ</b>	>100
<b>PIC24H</b>	>100
<b>dsPIC30F</b>	~10
<b>dsPIC33FJ</b>	>100

Here are the most common things that may need initialization that is not covered automatically by ds30 Loader:

### 4.1 Analog pins

Pins that can be used by the A/D are many times configured as analog on startup. If any of those pins that are to be used by the communication module they need to be configured to be digital. Read more about this in the device datasheet, sections I/O Ports and A/D module.

### 4.2 PPS

On PICs/dsPICs with the peripheral pin select feature it must be configured manually. There is a template available in user\_code.s. More information about PPS is found in the I/O Ports section of the device datasheet.

### 4.3 Oscillator

If the internal oscillator is to be used it may need to be configured it for a higher frequency. It is often not set for to maximum frequency on startup.

## 5. Linker script

### 5.1 PIC32

The PIC32 boot loader firmware is shipped with a custom linker script which is placed in the same directory as the MPLAB IDE project file (.mcp). The MPLAB project has the build directory policy set to the project directory. If it is changed to the output directory; the linker script must be moved there. The build directory policy is found in the project build options dialog which is opened from the Project\Build options...\Project menu in MPLAB IDE.

## 6. Build

- Select "Release" in menu "Project->Build Configuration"
- Start build by clicking menu "Project->Build All"
- Notice any warnings.
- Fix errors. ds30 Loader itself may generate errors, see the table below. For other errors, consult the Microchip tool suite documentation.

Error	Description	Solution
Unknown device specified	The selected device may be not supported	Contact the author to get device support.
Do you need to configure communication pins to be digital? If not, remove this line	See 4.1	Configure A/D if needed and then remove the line that generates the error.
You need to configure PPS	See 4.2	Configure PPS then remove the line that generates the error.
Both UART and CAN is specified	ds30 Loader can only operate with one communication module	Select only one communication module in settings.inc
Neither UART nor CAN is	Exactly one	Select a communication



specified	communication module must be selected in settings.inc	module in settings.inc
Fcy specified is out of range		Change Fcy to be within the devices maximum.
Both CAN ports are specified	ds30 Loader can only operate with one communication module	Select only one communication module in settings.inc
CAN is specified for a device that don't have CAN		Select a communication module that is available for the selected device
CAN2 specified for a device that only has CAN1		Select a communication module that is available for the selected device
Both uarts are specified		Select only one communication module in settings.inc
UART2 specified for a device that only has uart1		Select a communication module that is available for the selected device
Baud rate error is more than 2.5%. Remove this check or try another baud rate and/or clock speed.		Try a different baud rate or oscillator frequency.
overflow in delay calculation	Oscillator frequency and timings may be incompatible	
BLSTART_ is out of range	Oscillator frequency and timings may be incompatible	
BLSTART_ might be out of range	Oscillator frequency and timings may be incompatible	
BLDELAY_ is out of range	Oscillator frequency and timings may be incompatible	
BLDELAY_ might be out of range	Oscillator frequency and timings may be incompatible	
You need to configure PPS	See 4.2	Configure PPS then remove the line that generates the error.
No communication is specified		Select exactly one communication module in settings.inc
CanBus specified for a device that only has uart		Select a communication module that is available for the

		selected device
UART1 and Canbus specified		Select only one communication module in settings.inc
UART2 and Canbus specified		Select only one communication module in settings.inc
TX enable is not available for CAN		Disable tx enable
UART2 specified for a device that only has uart1		Select a communication module that is available for the selected device
spbrg_value_ is out of range	Oscillator frequency and baud rate may be incompatible	Try a different baud rate or oscillator frequency
spbrg_value_ might be out of range	Oscillator frequency and baud rate may be incompatible	Try a different baud rate or oscillator frequency

## **7. Erase device**

If code protection is used the device should be erased completely. This may be essential to correct boot loader operation if code protection is used.

### **7.1 Programmer supported by MPLAB IDE**

On the menu Programmer->Erase Flash Device

### **7.2 Programmer not supported by MPLAB IDE**

Consult the programmer manual.

## **8. Write boot loader to PIC**

### **8.1 Programmer supported by MPLAB IDE**

On the menu Programmer->Program

Notice that this step requires an ordinary programmer such as the ICD2. The boot loader itself cannot be used to write the boot loader.

### **8.2 Programmer not supported by MPLAB IDE**

Consult the programmer manual.

## **9. Adapt the user application**

### **9.1 PIC12 and PIC16**

The user application does not need any adaptation.

### **9.2 PIC18**

The user application does not need any adaptation.

### **9.3 PIC24F and dsPIC**

The user application does not need any adaptation.

### **9.4 PIC32**

A custom linker script must be used. It is delivered together with firmware. The file must not be renamed and it must not be include in the MPLAB IDE/MPLAB X IDE project.

When MPLAB IDE is used the procdefs.ld linker script file must be placed in the configured build directory. The build directory is configured from the menu Project->Build Options...->Project\Directories\Build Directory Policy.

When MPLAB X IDE is used the procdefs.ld linker script file must be placed in the project directory. The project directory name ends with ".X" and contains the Makefile file.

When the application is built with custom linker script it will not run without the boot loader.

## Known limitations

### **PIC18**

#### **Watchdog Time-out Flag bit**

(commercial C version only)

The TO bit in the RCON register will not indicate correctly. There is a workaround. A copy of the RCON register with correct values is stored in RAM by ds30 Loader, read that instead of the actual RCON register.

```
UINT16* rcon_valid_ptr = (UINT16*)0x10;
UINT8 rcon_value;

if ( *rcon_valid_ptr == 0xa55a ) {
    // Application was loaded from ds30 Loader, use its RCON copy
    UINT8* rcon_ptr = (UINT8*)0x12;
    rcon_value = *rcon_ptr;
} else {
    // Application is running without ds30 Loader,
    // Use the real RCON register
    rcon_value = RCON;
}
// Use rcon_value instead of RCON
```

## Considerations

### **Code protection**

Depending on configuration, write verification and read operation may not function. Write verification must be disabled in this case.

### **Data stored in flash memory**

If the user application stores data in flash memory, this data must be placed in a separate page/row that does not contain any actual code or it will be overwritten on the next write.

### **Linker script**

There is usually no need to alter the linker script for ds30 Loader firmware. In some cases when using large data arrays, the linker or assembler may place these in the same place as the boot loader. One way to solve this is to reserve the boot loader addresses in the linker script. Another solution is to place the data array at a specific address that does not interfere with the boot loader memory space.

### **Oscillator**

It is strongly recommended to use the same oscillator setup for both the boot loader and the user application. If you have code to setup your oscillator and/or pll, it is recommended to move that code to the boot loader.

### **Using different oscillator settings for boot loader and application**

If the user application is to be run on a battery powered device, the oscillator may be running at very low speed. To still achieve low boot loader write time, one might want to have different oscillator setups for boot loader and application. The solution is to add clock switching/pll initialization code in the boot loader firmware.

### **Unplanned download of different oscillator setup**

If one needs to download a different oscillator setup and the boot loader does not already have clock switching code, great care must be taken to make sure that the boot loader will still be operable with the new oscillator setup. There are only a few ways to do this

The simplest solution is to use the command reset method. That way, one can add clock switching code prior to loading the boot loader. It is still risky because if the application gets corrupted or a write failed the boot can not get loaded with compatible oscillator configuration. It could look something like this in pseudo code:

```
if ( ReceivedBlResetCommand )
    SwitchToBootloaderOscillatorSetup()
    GotoBootloader()
end if
```

## ***Interrupts***

The boot loader does not use interrupt but some interrupt flags will be set. Always clear respective interrupts flag prior to enabling an interrupt in your application.

## ***PIC18 extended instruction set***

ds30 Loader does not rely on the extended instruction set, however the boot loader and the application should use the same setting when compiled to ensure correct operation. Extended instruction set is not supported after the trial period of the C18 evaluation version.

## ***PPS***

The PPS configuration registers are not locked by ds30 Loader.

## ***Register default values***

Some register values are not restored when download is complete. For details, examine the code.

## ***User application***

If the boot loader is activated by resetting the device, there is usually no need to adapt the user application. Performing a device reset is preferred to using call, goto or branch because the boot loader may assume reset values of some registers.

If the boot loader is called, “gotoed”, or branched to from the user application, interrupts should be disabled prior to calling the boot loader.

## ***Watchdog***

A ClrWdt instruction is placed in the receive loop. Depending on configuration this may not be enough. In this case the watchdog should be disabled during boot loader operation.

## Appendix A – Links

ds30 Loader website

<http://www.ds30loader.com>

ds30 Loader free edition website

<http://mrmackey.no-ip.org/elektronik/ds30loader/>