ds30 Loader
Firmware manual

Copyright © 2009, Mikael Gustafsson

Table of contents

Table of contents 2
Document History 3
Rev D 3
Introduction 4
ds30 Loader 4
Prerequisites and Requirements 4
Trademarks 4
The basics 5
Different firmware versions 5
The MPLAB project 5
ds30Loader.asm / ds30Loader.s 5
settings.inc 5
devices.inc 5
xxx.lkr / xxx.gld 5
Bootloader placement 5
Usage 6
1. Select device 6
2. Customize settings.inc 6
3. Add own initialization code 7
4. Build 7
5. Write bootloader to PIC 7
Integration in user application 8
MPLAB C30 8
1. 8
2. 8
3. 8
Considerations 10
Default values 10
Data stored in flash memory 10
Oscillator 10
Using different configs for bootloader and application 10
Unplanned download of different oscillator setup 11
Linker script 11
Interrupts 11
Watchdog 11

Copyright © 2009, Mikael Gustafsson

Document History

Rev F

Added PIC16F information

Rev E

Added additional details in appendix B

Rev D

Added integration section.
Fixed incorrect page size values in appendix B
Fixed incorrect data in appendix C

Copyright © 2009, Mikael Gustafsson

Introduction

ds30 Loader

ds30 Loader is an open source bootloader for PIC18F, PIC24F, PIC24H, dsPIC30F and dsPIC33F families of MCUs from Micropchip. It supports all devices in each family out of the box (those in production), only minor adjustments need to be done in firmware. The firmware is written in asm/asm30 and comes with a preconfigured MPLAB-project. The GUI is written in C#.

Prerequisites and Requirements

Depending on which firmware is used, MPLAB ASM30 or MPASM assembler is needed. MPLAB IDE was used during development and is used in this document.

Trademarks

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

The basics

Different firmware versions

The ds30 Loader firmware comes in several different versions. Each designed to work with a specific family of PIC devices. The following firmwares are available:

- PIC18F
- PIC18FJ
- PIC24F
- PIC24FJ
- PIC24H
- dsPIC30F
- dsPIC33FJ

The MPLAB project

The firmware MPLAB project typically consists of three 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.

settings.inc

This file contains all common user customizations such as uart assignment, baudrate, device and more. This file needs to be modified in order to make the bootloader work for each different hardware setup. This file is included by ds30Loader.asm/ds30Loader.s.

devices.inc

This file contains device specific constants such as size of eeprom and number of uarts available. This file is included by ds30Loader.asm/settings.inc.

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 toolsuite.

Bootloader placement

The bootloader is normally placed at the very end of flash memory. This way there is no need to alter the linker script. Some device families place configs at the of flash memory, in those cases the bootloader is placed just before the configs.

Copyright © 2009, Mikael Gustafsson

Usage

1. Select device

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

2. Customize settings.inc

All lines commented with xxx needs to verified/changed. A short description follows of the most common settings. Not all settings are available in any firmware.

.equ __30F4011, 1
Simply set to your device name.

LIST P=18F2550
Simply set to your device name.

FCY
Set to instruction cycle clock speed (nr of instructions per second).

FOsc
Set to instruction cycle clock speed (nr of instructions per second).

BAUDRATE
Set to uart baudrate, the brg value is automatically calculated. If the error of the chosen baudrate exceeds 2.5% an error message will be displayed when assembling.

BLTIME
This is the receive timeout in milliseconds and also the time between boot to start of user application if no download is started.

USE_UARTx
Uncomment the line matching the uart you are using.

USE_ALTIO
Uncomment to user alternative i/o for uart 1. An error message will displayed if alternative i/o is not available or if uart 1 is not chosen.

USE_TXENABLE
Uncomment to use a tx enable pin allowing RS485 communication.

TRISR_TXE
Set to tris register of tx enable pin.

LATR_TXE

Copyright © 2009, Mikael Gustafsson

Set to lat register of tx enable pin.

TRISB_TXE

Set to bit in tris register of tx enable pin.

LATB_TXE

Set to bit in lat register of tx enable pin.

config xxx

Set desired configuration bits. These can also be set in MPLAB IDE instead, they are found on the menu Configure->Configuration bits...

3. Add own initialization code

If needed, add init code at designed area in ds30loader.asm/ds30loader.s. In some families the space available for user code is restricted to a few instructions. Se table below for details. The exact number depends on what features are enabled.

	Words free to use for user code
PIC16F	~15
PIC18F	~30
PIC18FJ	>100
PIC24F	~10
PIC24FJ	>100
PIC24H	>100
dsPIC30F	~10
dsPIC33FJ	>100

4. Build

On the menu Project->Build All (Ctrl+F10).
Notice any warning and fix any error.

5. Write bootloader to PIC

On the menu Programmer->Program
Notice that this step requires an ordinary programmer such as the ICD2. The bootloader itself cannot be used to download the bootloader.

Copyright © 2009, Mikael Gustafsson

Integration in user application

Integration is done differently depending on which language toolset is used.

MPLAB C30

1. Add bootloader files

Add the bootloader files in MPLAB IDE. Preferably they are copied to the user application directory first.

2. Add crt object file

Add crt0.o or crt1.o if needed, they are usually found in c:\program files \Microchip\MPLAB ASM30 Suite\lib\. See C30 user's guide section "4.5 STARTUP AND INITIALIZATION" for more information.

crt0.o:
"Primary version, with data initialization support. The linker loads this version when the --data-init option is selected."

crt1.o:
"Alternate version, without data initialization support. The linker loads this version when the --no-data-init option is selected."

3. Hardcode goto

"Hardcode" the goto to the user data initialization or main.

```
; ---------------------------------
; Start of code section in program memory
; ---------------------------------
          .section *, code,address(STARTADDR-4)
usrapp:   goto __resetPRI
```

or

```
; ---------------------------------
; Start of code section in program memory
; ---------------------------------
          .section *, code,address(STARTADDR-4)
usrapp:   goto _main
```

Copyright © 2009, Mikael Gustafsson

4. Remove configs

Remove configs from bootloader or user application. They can only be defined in one place.

5. Compile

Compile and take a look at the program memory to verify that the goto at 0x00 is pointing at the bootloader in the end of the memory range.

Copyright © 2009, Mikael Gustafsson

Considerations

Default values

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

Data stored in flash memory

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

Oscillator

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

Using different configs for bootloader 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 bootloader download times, one might want to have different oscillator setups for bootloader and application.

The solution is to add clock switching/pll initialization code in the bootloader firmware. For dsPIC33 it could look like this:

```
        ; New oscillator, FRC+PLL
        MOV           #0b001, W0
        ; OSCCONH (high byte) Unlock Sequence
        mov           #OSCCONH, W1
        mov           #0x78, W2
        mov           #0x9A, W3
        mov.b         W2, [W1]
        mov.b         W3, [W1]
        ; Set New Oscillator Selection
        mov.b         W0, [W1]
        ; OSCCONL (low byte) Unlock Sequence
        mov           #OSCCONL, W1
        mov           #0x46, W2
        mov           #0x57, W3
        mov.b         W2, [W1]
        mov.b         W3, [W1]
        ; Enable Clock Switch
        bset          OSCCON, #OSWEN
        ; Wait for clock switch to finish
waitclk:btsc          OSCCONL, #OSWEN
        bra           waitclk
```

Copyright © 2009, Mikael Gustafsson

Unplanned download of different oscillator setup

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

The simplest solution is to use the uart command reset method. That way, one can add clock switching code prior to loading the bootloader. It could look something like this in pseudo code:

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

Linker script

There is 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 bootloader. One way to solve this is to reserve the bootloader addresses in the linker script. Another solution is to place the data array at a specific address that does not interfere with the bootloader memory space.

Interrupts

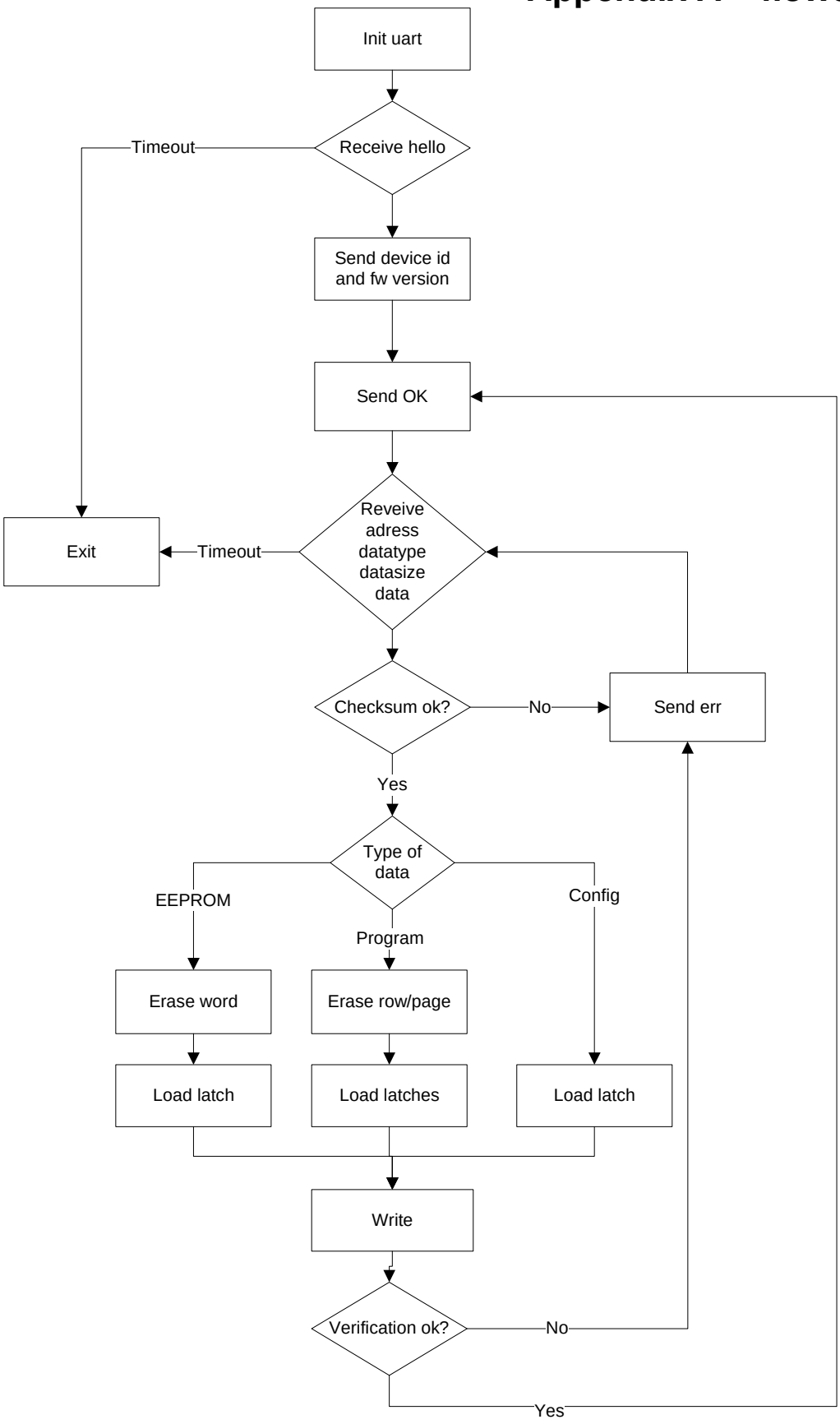
There are no considerations for interrupts.

Watchdog

A ClrWdt instruction is placed in the receive loop.

Copyright © 2009, Mikael Gustafsson

Appendix A – flowchart



Appendix B – Flash architecture details

	Pagesize [words]	Rowsize [words]	Wordsize [bits]	Flash-range	EE-range	Config-range
PIC16F	16/32	4/8	14	0x0 0x2000	0x2100 0x21FF	-
PIC18F	32	4/8/16/32	16	0x0 0x20000	0xF00000 0xF00400	0x300000 0x30000D
PIC18FJ	512	32	16	0x0 0x20000	-	-
PIC24F	n/a	32	24	0x0 0x2C00	0x7FFE00 0x800000	0xF80000 0xF8000E
PIC24FJ	512	64	24	0x0 0x2AC00	-	-
PIC24H	512	64	24	0x0 0x2AC00	-	0xF80000 0xF8000E
dsPIC30F	n/a	32	24	0x0 0x18000	0x7FF000 0x800000	0xF80000 0xF8000E
dsPIC33FJ	512	64	24	0x0 0x2AC00	-	0xF80000 0xF8000E

Copyright © 2009, Mikael Gustafsson

Appendix C – Bootloader details

	Size	Placement
PIC16F	192 words	End of memory
PIC18F	5 pages	5 last pages
PIC18FJ	1 page	2nd last page
PIC24F	4 rows	4 last rows
PIC24FJ	1 page	2nd last page
PIC24H	1 page	Last page
dsPIC30F	4 rows	4 last rows
dsPIC33FJ	1 page	Last page

Copyright © 2009, Mikael Gustafsson