

PICPROJECTS.NET

PIC Microcontroller Projects

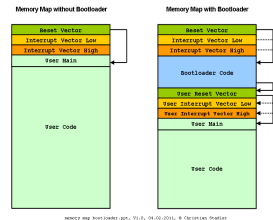
HOME PROJECTS DEVELOPMENT TOOLS BOOTLOADER FORUM DISCLAIMER PIC PROGRAMMER PI-PROJECTS.NET

Ads by Google ▶ Bootloader ▶ USB Pic ▶ IDE Pic ▶ Pic Low

SEARCH

Serial PIC Bootloader

- [Introduction](#)
- [Bootloader Basics](#)
- [Hardware](#)
- [Bootloader Firmware](#)
- [Bootloader Application](#)
- [Download](#)
- [User Code Relocation](#)



memory_map_bootloader.ppt, V1.0, 04.02.2011, © Christian Stadler

USB 2.0/3.0 White Paper

Key parameters for USB 2.0/3.0 for Switches and Applications

○ ○

Introduction

What is a bootloader? A bootloader is a small program running in the microcontroller to be programmed. This program allows downloading new firmware to the microcontroller via e.g. a serial interface like RS232 or via USB.

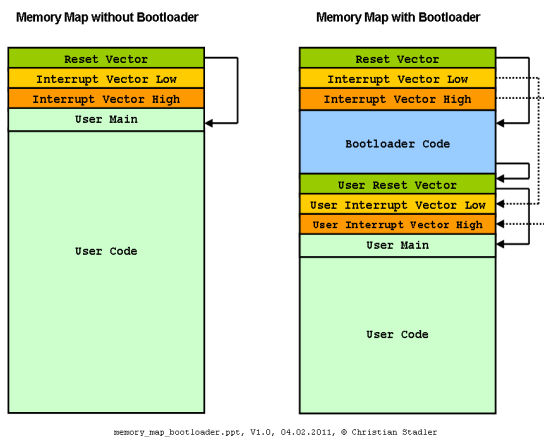
This project shows how to implement a serial bootloader for PIC microcontrollers. The download of the new firmware is done via a RS232 connection. Currently the bootloader only supports PIC18F devices and is written with CCS C compiler (C18 compiler support will follow soon).

Bootloader Basics

The bootloader needs to be the first thing which is running after device reset. Hence the bootloader needs to use the reset vector, i.e. the reset vector will point to the start of the bootloader code.

After device reset, the bootloader checks if a new firmware shall be programmed or if the application firmware shall be started. In this example this is done by checking the state of an I/O pin. If the I/O pin is high the bootloader waits for firmware download, if the I/O pin is low, it starts the application firmware.

Further the interrupt vector needs to be remapped. This is necessary because the interrupt service routines will be located at the user code.



memory_map_bootloader.ppt, V1.0, 04.02.2011, © Christian Stadler

Search

Google™ Custom Search

Create Your Website

Just ₹ 99 for Complete Website. Free Domain & Hosting Included.

○ ○

PIC PROJECTS

RS232 Communication

DS1820 Temperature Sensor

DS2482 1-Wire bridge

RFM12 Module

DCF77 Time Signal Decoding

SD-Card and FAT16 Demo

RC5 Decoder

HARDWARE PROJECTS

PIC16F/18F Experiment Board

ENC28J60 Modul

RS232 Module

LCD Adapter

DEVELOPMENT TOOLS

PIC C Compiler Overview

PIC Programmer

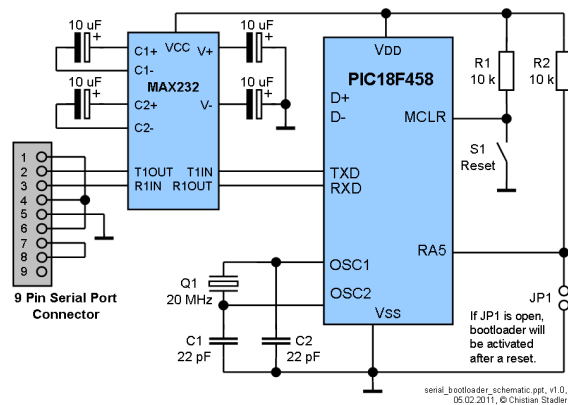
BOOTLOADER

Serial PIC Bootloader

USB PIC Bootloader

Hardware

The bootloader uses the PIC's UART to receive the HEX file to be programmed. The HEX file is sent by an application program running on the PC.



Before the bootloader can be used, the bootloader firmware must be programmed into the PIC. For that a PIC programmer hardware and corresponding Software is required. This is only required once to get the bootloader code into the PIC. More info about PIC programmer hardware and software can be found at the [PICPgm page](#).

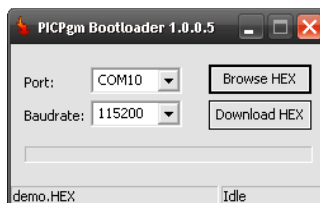
The PICPgm Bootloader application can be downloaded below. Additionally, the source code can be downloaded below.

Bootloader Firmware

The bootloader firmware is written in C language. Currently only CCS C compiler is supported, but Microchip C18 compiler support will follow soon.

Bootloader Application

The task of the bootloader application is to send the HEX file to be programmed to the bootloader firmware running in the PIC to be programmed.



Download

- Serial bootloader firmware (CCS Compiler): [bootloader_firmware_v1001.zip](#)
- Bootloader PC Application: [picpgmboot_v1005.zip](#)
- Bootloader PC Application Source: [picpgmboot_src_v1005.zip](#)

How to Relocate User Code

Programs which shall be flashed with the bootloader need to be adapted since the bootloader uses the memory area 0x000 to 0x3FF. This area is usually used by the application software itself. So the following changes are required:

- reserve boot block area (0x000-0x3FF)
- map reset vector from 0x000 to 0x400
- map interrupt vector from 0x008/0x018 to 0x408/0x418

How this can be done is depending on the compiler which is used for generation of the HEX file.

RASPBERRY PI & CO

[Projects with Raspberry Pi](#) and other devices running Linux.



MPLAB C18 compiler

For the MPLAB C18 compiler the following changes need to be done:

```
...
extern void startup(void);           // See c018i.c in your C18 compiler dir
#pragma code _RESET_INTERRUPT_VECTOR = 0x000400
void _reset(void)
{
    asm goto _startup_endasm
}
#pragma code
#pragma code HIGH_INTERRUPT_VECTOR = 0x000408
void _high_ISR(void)
{
}
#pragma code LOW_INTERRUPT_VECTOR = 0x000418
void _low_ISR(void)
{
}
/* This pragma forces the code below this line to be put into the code */
/* section (memory address >= 0x42A). See linker script for details. */
#pragma code
...
```

Further, a linker script is required to make the application software start at 0x400.

TODO: ADD LINKER SCRIPT EXAMPLE HERE

CCS C Compiler

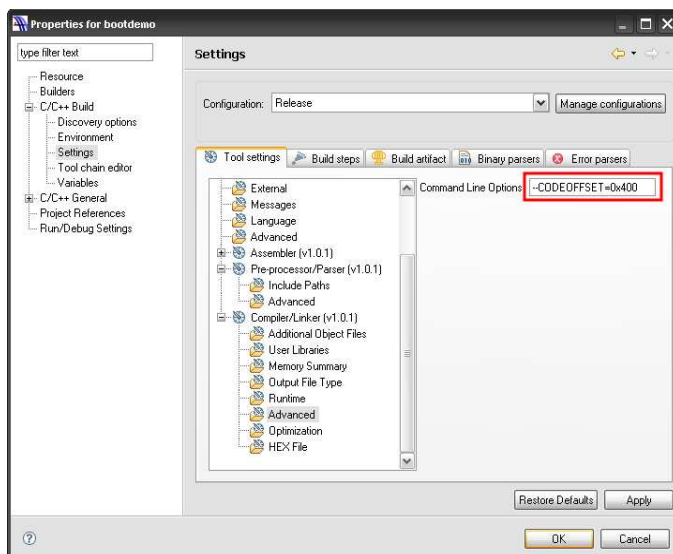
For the CCS C compiler it is a little bit simpler:

```
...
/* ----- */
/* map reset vector and interrupt vector */
/* 0x000-0x3FF is used by the bootloader. The bootloader maps the original */
/* reset vector (0x000) to 0x400 and the interrupt vector (0x008) to 0x408. */
/* ----- */
#pragma build (reset=0x400, interrupt=0x408)
/* ----- */
/* reserve boot block area */
/* This memory range is used by the bootloader, so the application must not */
/* use this area. */
/* ----- */
#pragma org 0, 0x3FF {}
...
```

HI-TECH C Compiler

For the HI-TECH C compiler the reset vector can be changed via the compiler option **--CODEOFFSET**. For the MCHPUSB bootloader we have to set the option to **--CODEOFFSET=0x0400**

If you use the HI-TIDE IDE, the option can be specified in the project properties (Project => Properties => C/C++ Build => Settings => Advanced => Command Line Options):



picprojects.net

PIC microcontroller hardware and software projects with description, schematics and source code!

Development Tools

[PICPgm PIC Programmer](#)

[PIC Compiler Overview](#)

[USB Bootloader](#)

[Serial Bootloader](#)

Latest Articles

[LCD Adapter](#)

[RS232 Communication](#)

[RC5 Decoder](#)

picprojects.net

