EC-USB Bootloader

Version 2.0.3

Table of contents

1 Introduction

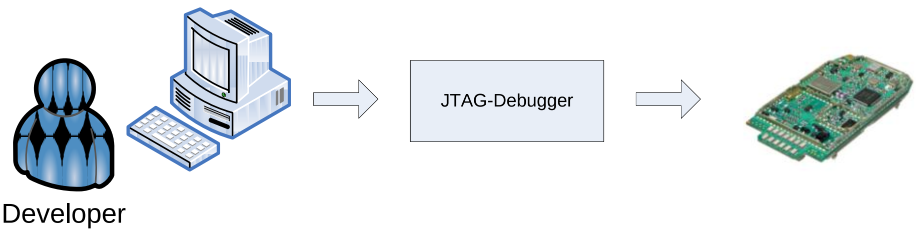
The EC-USB Bootloader is a bootloader that is part of EC-USB and EC-Boot.

This document describes how the bootloader works, both in the host and in the device.

2 Overview

2.1 General

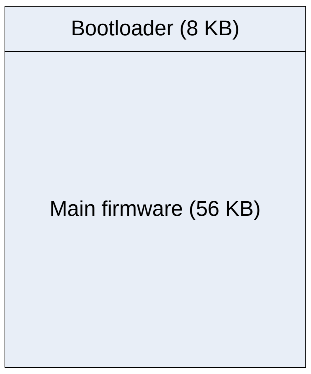
During development, a developor often use a dedicated programmer or debugger to program new firmware into a device.



This works well during development but software updates are usually also needed when the device has been sold to the end customers. Since the end customer (or the technician doing the update for the end customer) doesn't have the same equipment or skills as the developer, a different approach is needed to update the device firmware.

For this reason, a bootloader is used. A bootloader is a piece of software that is programmed into the device at production whose only purpuse is to update the firmware of the device. The memory of the device is split into a bootloader part and into a main firmware part.

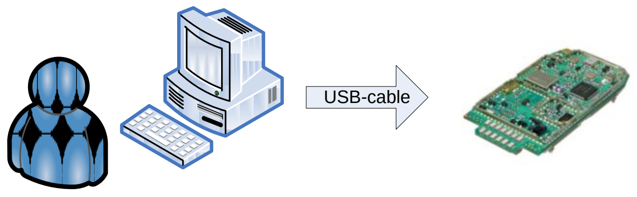
For an MCU with 64KB of flash, the memory might be split like this:



When the device powers up, the bootloader will be the first to start. If it detects that the host wants to program new firmware, the main part of the bootloader is invoked and it communicates with the host to receive the new firmware which it programs into the main firmware area of the memory.

But if the bootloader concludes that the host does not want to update the firmware it instead calls the main firmware to run the software needed for the main function of the device.

For the EC-USB bootloader, the end user or technician connects directly to the device using a USB cable to update the software.



End user/Technician

2.2 Detecting if the host wants to update the firmware

The bootloader will need to decide at startup if the host wants to update the firmware or if the device should power up normally. There are different approaches to this but for a USB device, some sort of user interaction is needed.

The included bootloader sample will always enter its bootloader mode and you will need to modifiy it for your hardware.

One way of determining if the device should enter bootloader state would be to have the user hold a button during startup if the firmware is about to be updated. The bootloader can detect this and enter its bootloader mode if the butten is pressed.

2.3 The update procedure

The basic procedure to update the firmware:

1. Make sure the WindowsUSBBootDriver is installed on the PC host that is to performing the update.

   A pre-complided, signed binary is available in:
   `EC-USB 2.0.3/Windows/Drivers/Binary/WindowsUSBBootloaderDriver`

   The source for the driver is available in:
   `EC-USB 2.0.3/Windows/Drivers/Source/WindowsUSBBootloaderDriver`

2. Hold a button or similar on the device to indicate that it should stay in bootloader mode

3. Use the UpdateFirmware Windows application to load the new software. The UpdateFirmware application is a command line tool that takes the S19 file to program as a command line argument.

   A pre-complided binary is available in:
   `EC-USB 2.0.3/Windows/Example Applications/Binary/`

   The source for the driver is available in:
   `EC-USB 2.0.3/Windows/Example Applications/Source/VS2005/CPP/UpdateFirmware/`

2.4 Static bootloader data

The included bootloader samples contains a dedicated 16 byte area in the bootloader for static data. This data can be used to store information about the device that doesn't change when the firmware is updated.

The included samples stores the following information in the bootloader area:

- **Device serial number**

  The device's serial number is usful in many situations were you need to differentiate between devices.

- **Device hardware ID**

  A hardware ID can be used to interally keep track of different types of hardware. The UpdateFirmware application could be modified to automatically identifiy which firmware is needed for the connected device and automatically send the correct one.

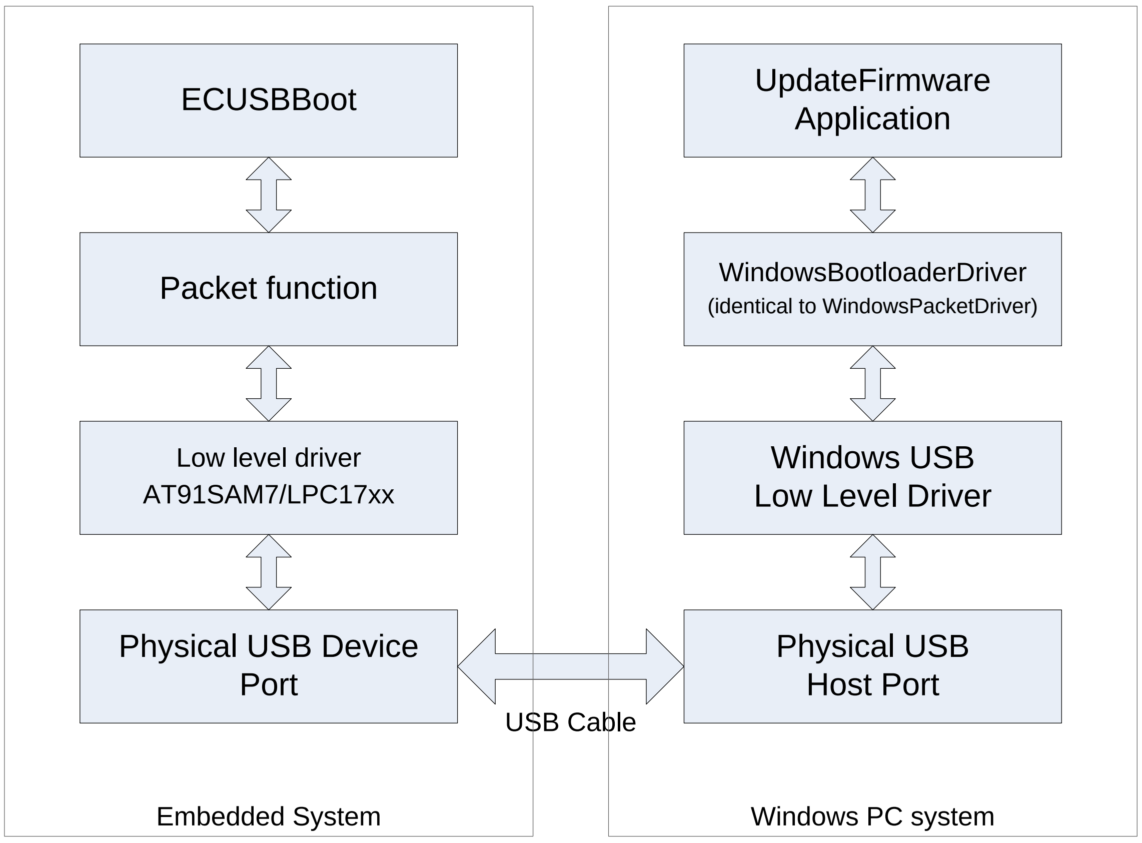
- **Bootloader version**

  Can be used to identify which bootloader version that is present.

3 Software

The EC-USB bootloader is based on the Packet sample and uses the same drivers. The only difference is that it uses another ProductID to indicate that it is a bootloader.

This structure of the software is shown below.


ECUSBBoot
Packet function
Low level driver AT91SAM7/LPC17xx
Physical USB Device Port
Embedded System
UpdateFirmware Application
WindowsBootloaderDriver (identical to WindowsPacketDriver)
Windows USB Low Level Driver
Physical USB Host Port
Windows PC system
USB Cable

4 Protocol

Once USB enumeration is completed, the host (the PC) initiates all communication with the bootloader. The bootloader receives commands and replies to the host.

The protocol is designed to be simple and not for speed so after each command the host waits for a reply before continuing.

4.1 Commands

The basic command structure consists of a byte specifying the command and the data that goes with the command shown below.

BYTE 0	1	...	...	...	...	packetSize-1
Command	Command data					

Figure 1: Basic command structure

When the device has processed the command it replies with either a success reply or a failure reply. Successful replies starts with 0x00 and optionally contains some data.

BYTE 0	1	...	...	...	...	packetSize-1
0x00	Reply data (optional)					

Figure 2: Basic successful reply structure

Failed commands reply with a single byte which is the error code.

BYTE 0
Error Code

Figure 3: Basic failed reply structure

The defined error codes are:

0x01: Generic error
0x02: Unknown command
0x03: Address out of range error

4.1.1 Get static bootloader data (0x01)

The *Get static bootloader data* command retrieves data from the static data area of the bootloader. This area is used to store static data for the bootloader. For the sample bootloader, this area is 16 bytes.

BYTE 0
0x01

Figure 4: Get static bootloader data command

If the command succeeds, the reply below is returned.

BYTE 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0x00	Hardware ID (LSB first)				Serial No (LSB first)				Bootloader version				Reserved			

Figure 5: Get static bootloader data successful reply

For a failed command the basic failure reply is returned with an error code.

4.1.2 Get address range (0x02)

The *Get static bootloader data* command retrieves information about the programmable address range of the device. For a 64 KB device where the first 8 KB is occupied by the bootloader, the command would return Start address = 0x2000 and Size = 0xE000

BYTE

0
0x02

Figure 6: Get address range command

If the command succeeds, the reply below is returned.

BYTE

0	1	2	3	4	5	6	7	8
0x00	Start address (LSB first)				Size (LSB first)			

Figure 7: Get address range successful reply

For a failed command the basic failure reply is returned with an error code.

4.1.3 Program block command (0x03)

The *Program block* command writes 512 bytes of data to the internal flash of the device.

BYTE

0	1	2	3	4	5	…	…	517
0x02	Address (LSB first)				Program data			

Figure 8: Program block command structure

The host always sends the program blocks with increasing addresses. This means that the device should erase a block when it encounters the first block in that block.

Example for device with 4096 byte pages:

- **Host:** Sends Program block, address = 0x2000 with 512 bytes of data
- **Device:** 0x2000 is on an even 4096 boundary so device erases the block located at 0x2000 – 0x2FFF
- **Device:** Writes the 512 bytes of data to 0x2000 – 0x21FF
- **Device:** Sends reply with status = 0x00 to indicate success
- **Host:** Send Program block, address = 0x2200 with 512 bytes of data
- **Device:** 0x2200 is NOT on an even page boundry so device does not erase memory
- **Device:** Writes the 512 bytes of data to 0x2200 – 0x23FF
- **Device:** Sends reply with status = 0x00 to indicate success
- Continues until the entire device memory is programmed

If the command succeeds, the basic successful reply with no reply data is returned.

For a failed command the basic failure reply is returned with an error code.