An Adaptive and Compact UART BootLoader

ANxxxxx

Author: Rolf Nooteboom
Associated Project: Yes
Associated Part Family: PSoC-1
GET FREE SAMPLES HERE (CY Sample Request Form for all Product Lines)
Software Version: PSoC Designer™ 5.0 SP2
Associated Application Notes: None

Abstract

This application note describes how to implement an UART bootloader in your project, without loosing any of the digital- or analog block resources. Hardware examples includes RS-232 operation, RS-485 operation and Bluetooth operation.

Introduction

In today's world, many products consist of a microprocessor or microcontroller. The product's functionality is defined within the software running on the processor or controller. The software is programmed into the product before it leaves the factory. It may be possible that the software changes after products are fabricated. For instance: when functionality is added or when previous software contained bugs. In this case, the product has to be reprogrammed with the new software.

There are several possibilities to update software running on a PSoC™ Mixed Signal Array:

1. Replace the PSoC.
2. Reprogram the PSoC using the In-System Serial Programming (ISSP) protocol.
3. Reprogram the PSoC using a pre-programmed bootloader.

The first two options are are generally not predestined for the product's users. In case you want a user to be able to update the product with the new software, a bootloader implementation is a good feature.

A bootloader application should be easy to handle for the user. A good bootloader is able to deal with user errors, such as power interruption when programming or uploading wrong firmware. On the other hand, the bootloader code should be as compact as possible and use less to no system resources.

A Compact UART bootloader

The proposed PSoC™ UART bootloader is self configuring. This means no dynamic reconfiguration is required for the bootloader at the target project. Also no RAM is taken from the target project. The only resource, to be taken in account, is the usage of less then 1K of flash at the end of the memory map (fig. 1).

There are several ways to communicate with a bootloader. For instance: via USB, I2C or SPI. The main reason for selecting a UART is: all PSoC™ devices support UART. Older pc's are equipped with a COM (serial UART) port and USB-UART bridges are cheap and easy to get. Bluetooth also supports UART communication.

To ensure reliable data communication between host and target, all communication is validated with a checksum. To keep the bootloader compact, no checksum validation is done on the flash blocks. Additionally, the bootloader monitors each flash erase- and write-cycle to minimize the chance of programming error. In cases where high data security is needed, flash checksum can be implemented. Figure 1 show the reserved space for future checksum implementations.

Figure 1. Flash map of a 32 K PSoC™ with bootloader.

UART bootloader	0x7FFF 0x7C40
Configuration block: Startvector + future use (eg. checksum store)	0x7C3F 0x7C00
User Program Space	0x7BFF 0x0040
Vector Space	0x003F 0x0000

Protected Flash
Read protected Flash

Bootloader implementation

Host implementation

The bootloader comes with two Windows based applications. One for adding the bootloader to the project hex output (figure 2), another for communication with the target (figure 3). Find them in the ANxxx.zip file.

Figure 2. The bootloader HEXfile application

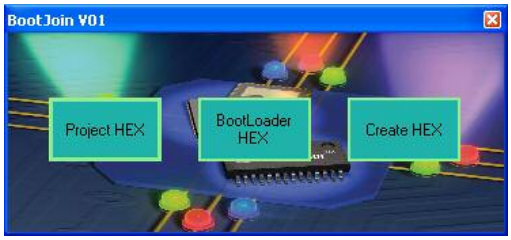
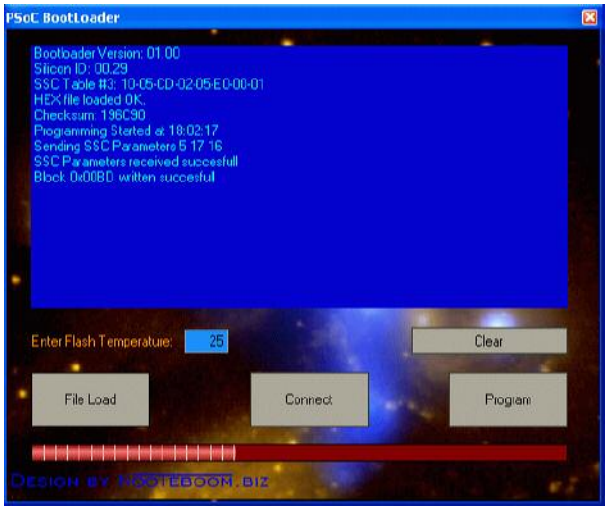


Figure 3. The bootloader PC application



Firmware implementation

Implementation of the bootloader firmware is very easy. Take the next steps to add the bootloader to your project:

1. Make sure there is no code or data at the bootloader space. To accomplish this with the HI-TECH C compiler, click Project>Settings>Compiler and type "--ROM=default,-7C00-7FFF" in the options field.
2. Do a complete project rebuild: Build>Rebuild Project.
3. Start the BootJoin PC application (figure 2) and load the Project HEX output and the BootLoader HEX output.
4. Click Create HEX to create the joined file.

Now the HEX file is created and you are ready to program the target ☺..

Hardware implementation

The UART needs two I/O pins (RXD and TXD) to communicate. No handshake is needed. The RXD and TXD signals are 'active low' and can be coupled directly to a serial driver IC (e.g. a MAX232). For computers without a serial port, you could use a Bluetooth serial module or USB-serial converter. Also RS-485 half-duplex and full-duplex modes are supported. For half-duplex, an extra output is used for data direction. See the schematics (figure 4) for details.

To change the pin configuration, the definitions in the bootloader.h file have to be changed. The following definitions will set the UART to PORT 1, the RX pin to P1[2] and the TX pin to P1[0]

```
<bootloader.h>

#define       UARTport      1
#define       RXpin         2
#define       TXpin         0
```

The bootloader in use

To update the firmware on your application, proceed the following steps:

1. Start the bootloader pc application.
2. Connect application to the host pc.
3. Click the Connect button on the pc application.
4. Power up the PSoC application. Now the version info and silicon ID should appear on the screen.
5. Load the hex file with the firmware at your choice. This hex file does not need to contain the bootloader code as it is already in the PSoC's flash.
6. Click the Program button and the firmware is updated.
7. You may disconnect the PSoC application when the process is finished.

If any of these steps may fail, the bootloader firmware should still be intact. The bootloader area is write protected en prevents itself from overwriting.

Bootloader Operation

The bootloader is invoked after a (Power On) reset or by a jump to the bootloader start address. The bootloader start address is located at the highest flash location minus 0x03FF (eg: 0x7C40 for a 32K device). First, the bootloader checks the accumulator. If the accumulator is zero (at a reset), the bootloader loop time is set to 100 milliseconds. If the accumulator is set to any non-zero value then the loop time is set to infinite. In this way it is possible for the main program to invoke the bootloader on, for example, a keypress.

If the string ”BOOT” is received, the loop time is set to infinite and the bootloader doesn't return until the 'exit' (table 1) command is received.

If the sequence *(string) ”BOOT” + (char) ”S” + (word) checksum* is received, the bootloader sends the SROM table 3 information, together with it's version number. The SROM information is needed to calculate the FlashErase and FlashWrite timings. The calculations are done by host (PC), to minimize bootloader flash size.

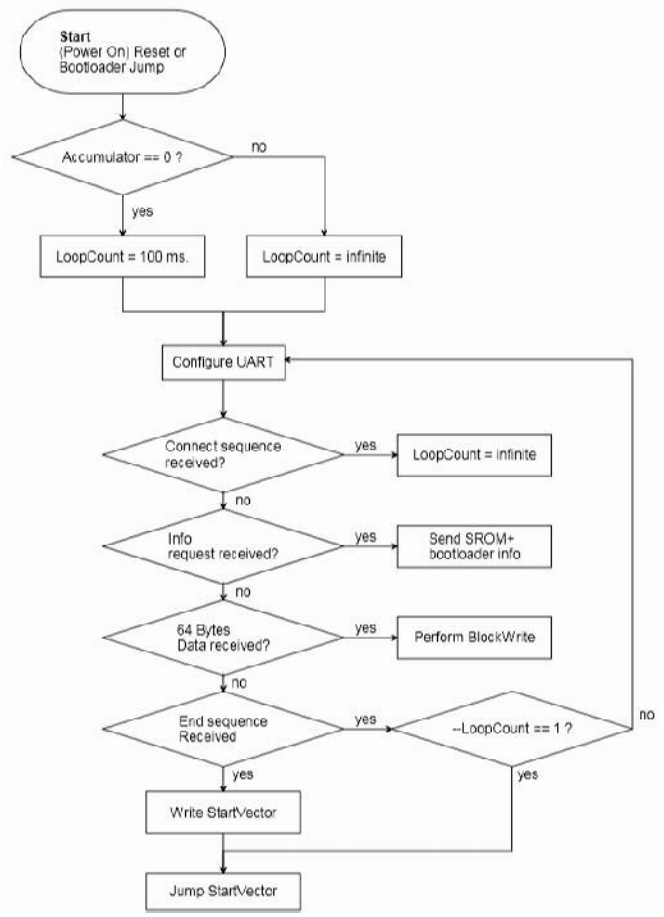
A block of 64 bytes is written by the bootloader once the following data is received: *(string) ”BOOT” + (char) ”W” + (byte) ClockE + (byte) ClockW + (word) BlockID + (byte[]) Data[64] + (word) checksum.*

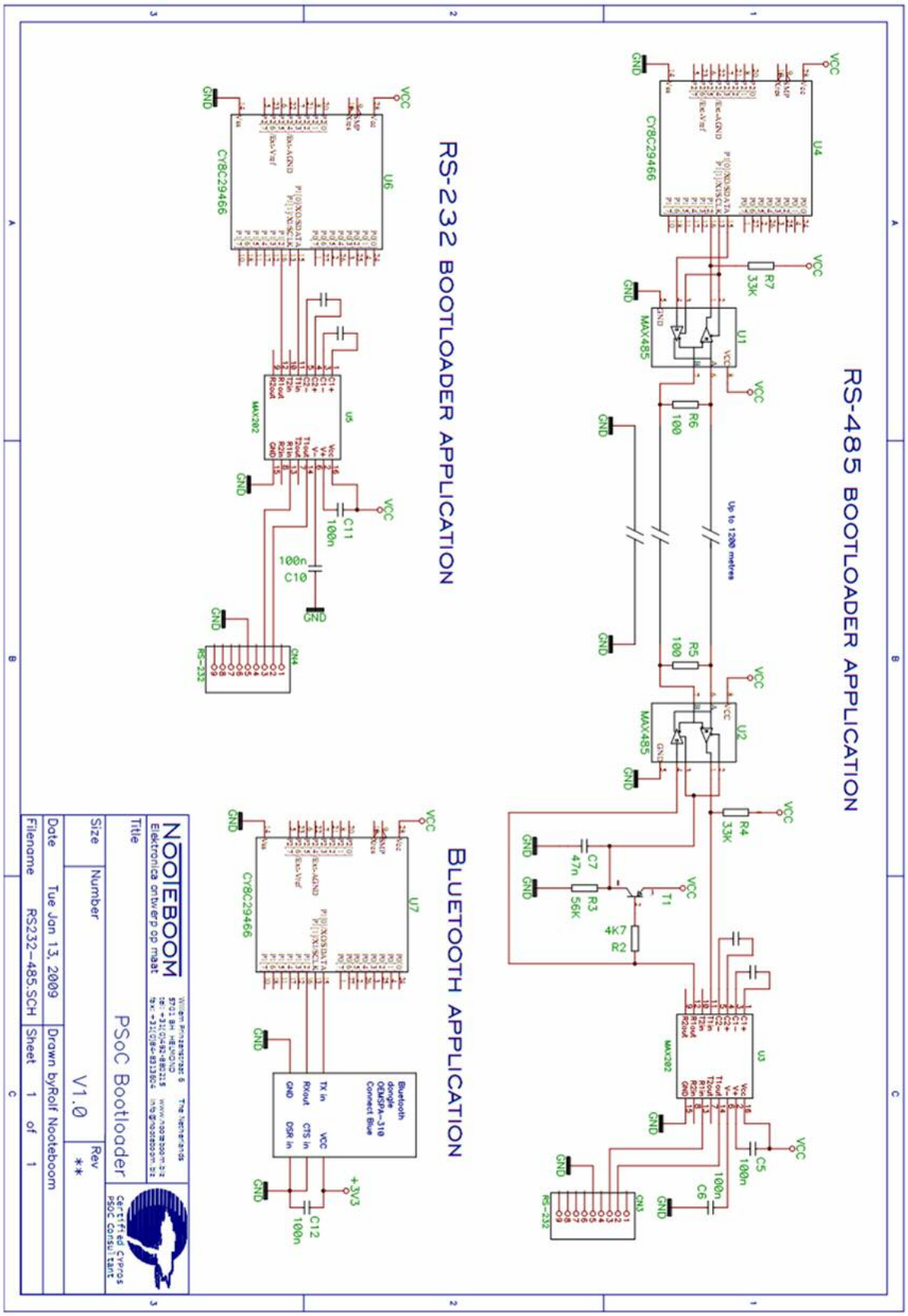
The bootloader operation is suspended at reception of the following data: *(string) ”BOOT” + (char) ”E” + (word) checksum.* At this time the program startvector is written to the configuration block (fig. 1).

Table 1: the command summary

Bootloader string	Command	Action
“BOOT”	**Identification**	**Timer = ∞**
“BOOTS”	**SROM request**	**Send SROM table 3 info**
“BOOTW”	**Write Block**	**Write 64 bytes of flash**
“BOOTE”	**Exit**	**Exit bootloader, start main**

Figure 4. The bootloader operation flowchart.




RS-485 BOOTLOADER APPLICATION
RS-232 BOOTLOADER APPLICATION
BLUETOOTH APPLICATION
CY8C29466
MAX485
MAX202
Up to 1200 metres
R7 33K
R6 100
R5 100
R4 33K
R3 56K
R2 4K7
C7 47n
C5 100n
C6 100n
C10 100n
C11 100n
C12 100n
T1
RS-232
Bluetooth dongle OEMSPA-310 Connect Blue
TX in
RXout
CTS in
DSR in
VCC
GND
+3V3
NOOTEBOOM
Elektronica ontwerp op maat
Title
PSoC Bootloader
Size
Number
V1.0
Rev
**
Date
Tue Jan 13, 2009
Filename
RS232-485.SCH
Sheet
1 of 1
Drawn by Rolf Nooteboom
Certified Cypros PSoC Consultant

About the Author (optional)

Name:	Rolf Nooteboom
Title:	Hard- and software designer and owner at Nooteboom Elektronica, The Netherlands
Background:	R. Nooteboom started an Electronics Design House in 1994. With the majority of 8 bit controller designs, Nooteboom started working with PSoC since it's introduction in 2001. Certified CYPros PSoC and Intelligent Lighting consultant.
Contact:	rolf@nooteboom-elektronica.nl +31 492 880215

The blue bar and the information below it are placed at the bottom portion of the page.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com/

Document History Page

Document Title: Application Note Template in MS Word

Document Number: 001-08990

Revision	ECN	Orig. of Change	Description of Change
**	483207	YIS	New Spec.
*A	823941	HMT	Update copyright. Add software/firmware disclaimer. Add Sample Request form URL. Add Character Set Quick Reference table. Add Sample Request From explanation. Add App. Note # and document # relation in footer. Suggest Firmware Flowchart as Figure 1. Add Equation Title style.
*B	1331006	HMT	Update gray to CY blue. Modify first-page header and AN Title for search. Fix symbol and bullets. Fix off color in text. Add AN bookmark field for AN# in header. Add secondary bullet. Update Equation style. Add Summary heading. Add subject-specific trademark information. Coordinate everything with FrameMaker template.

The Document History Page is only for approval of this template and must be deleted.