Encrypted SD/MMC Bootloader
PIC24/dsPIC33 Series

Andrew Smallridge
asmallri@brushelectronics.com
www.brushelectronics.com

The Brush Electronic's Encrypted SD/MMC Bootloader – PIC24/dsPIC33 Series (SDLX_C30) has been developed to enable the firmware upgrade of SD/MMC card equipped Microchip PIC24/dsPIC33 series microcontrollers. The bootloader will enable the upgrade of the user application firmware in the microcontroller and, optionally, the data in an external SPI based serial EEPROM.

The bootloader software has been developed in the C programming language using the Microchip C30 C compiler. The bootloader and the user's code execute independently. When the user application program is executing, the bootloader does not require PICs resources other than the consumed program memory. The bootloader software resides in low program memory of the PIC and transparently remaps the user application program's reset vector. The bootloader does not use interrupts and does not overlay the PIC's interrupt vector tables leaving these available for the user application.

The Bootloader solution includes the bootloader code resident in the target PIC, the XTEA encrypter, the images to be bootloaded into the PIC stored in the root directory of the SD/MMC card, and configuration file containing instructions for the bootloader stored on the SD/MMC card.

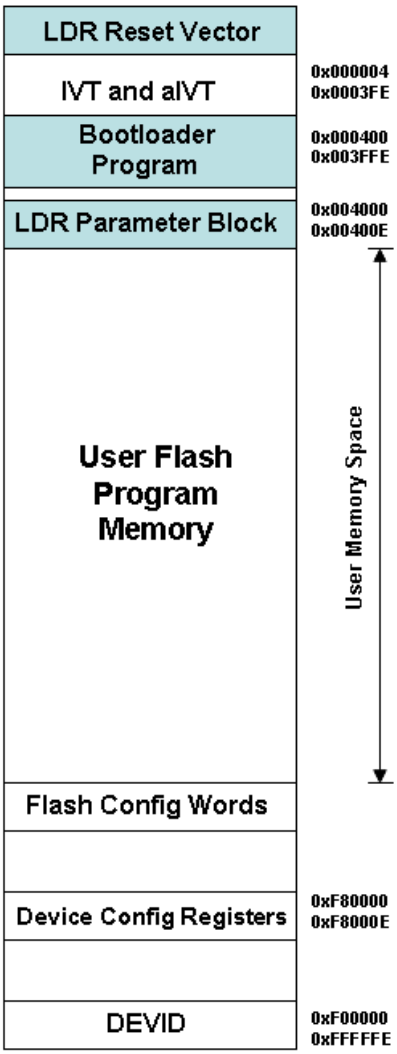
Key attributes of Brush Electronic's family of Encrypted SD/MMC Bootloaders:

- SD/MMC media uses the standard FAT file system
- User Program and EEPROM images are encrypted
- Plug and Play operation –the bootloader will automatically determine if the target platform should be bootloaded from the images on the SD/MMC card
- Incremental bootloader. As little as a single byte can be modified
- No resources are required on the target PIC other than the flash memory holding the boot code
- Does not consume the target application's RAM, EEPROM resources or the system interrupts. The bootloader is located in low program memory and automatically intercepts the application programs reset vectors.

PIC24/dsPIC33 Series Bootloader Memory MAP

The bootloader code resides in the lower 16K word of the program memory space above the Alternate Interrupt Vector Table. A 16 word bootloader parameter block (Parameter Block) is located on the first erase page boundary above the bootloader.

The bootloader transparently intercepts the PIC's reset vector, that is, the lower 4 words of program memory. The user's reset vector is transparently mapped into the LDR Parameter Block. The user's code MUST implement a GOTO instruction (a long jump) in the first 2 instructions. This requirement is met by a typical application linker script. Some development environments, such as the CCS PCD compiler does not use linker scripts, in this case a #build directive incorporate in the user application source code may be required. The LDR Parameter Block code includes a RESET instruction immediately following the remapped user's reset vector. If the user's program fails to implement a GOTO instruction in the first 2 instructions, then the RESET instruction is executed. The following diagram shows the loader's memory MAP.



Typical PIC24/dsPIC33 Series SD/MMC Bootloader Memory Map

The bootloader image resides between word address 0x00400 and 0x03FFE and may be protected if the target PIC supports boot-block-protection. In addition, the LDR Parameter Block is located between word address 0x04000 and 0x0400E. The parameter block contains the user application's reset vector, control flags and the ID

(identifier). The ID identifies the last LOADER.CFG configuration loaded by the bootloader. The LDR Parameter Block must reside in a non protected memory region to allow the bootloader to update the parameter block.

Encrypter Application

The XTEA.EXE Encrypter application is a Windows console application that accepts either one or three command line arguments. These arguments are the source file name of the standard Intel hex file to be encrypted, the 16 byte *Cipher Key* and the number of *Iterations* the cipher should be applied to the cipher text. The *Cipher key* must be exactly 16 bytes and must match the hard coded *XTEA_Key* in the Bootloader source code. The *Iterations* (typically 16) must match the hard coded *XTEA_Iterations* in the Bootloader source code. If only the source filename is specified on the command line then the XTEA's applications hard coded *XTEA_Key* and *XTEA_Iterations* constants are used.

The Encrypter generates the encrypted output file using the same filename as the source substituting *.cry* for the file extension.

Usage: XTEA sample.hex

Usage: XTEA sample.hex 123helloworld321 16

Basic Bootloader Operation

The bootloader software assumes control of the target platform on reset of the Microcontroller. If an SD/MMC card is present, the bootloader software will examine the contents of the root directory of the card to determine if the bootloader configuration file LOADER.CFG is present. If the file is present, the bootloader will parse the file to determine if a bootload process is required. If a bootload process is required, the encrypted hex files for the PIC and/or EEPROM are read from the SD/MMC card, decrypted and programmed in to the appropriate device. After the PIC and/or EEPROM has been bootloaded, and provided the user reset vector is installed in the LDR Parameter Block and the CodeState flag indicates the user application space is valid, the bootloader software will pass control to the user's application program. The CodeState flag is automatically invalidated when the configuration file contains an instruction to erase the PIC's program memory. The CodeState is set to valid when a correctly formatted EOF record has been processed from the PICs hex file and the user application's reset vector has been installed in the LDR Parameter Block.

Up to three files are used in the bootload process, a configuration file, the User Application Program's encrypted hex file for the PIC, and the external EEPROM encrypted hex file. The configuration file uses an identifier to enable the bootloader to determine if the LOADER.CFG file was the same as the configuration file used to install the current images in the target platform. If the ID in the configuration file matches the ID in the LDR Parameter Block, then the bootload process for the PIC and/or EEPROM is skipped.

The **LOADER.CFG** file, located in the root directory of the SD/MMC card, contains instructions for the bootloader and the ID (identifer) of the LOADER.CFG file. The bootloader code has been developed to support FAT12, FAT16 and FAT32 file systems however technical support is limited to the FAT16 and FAT32 file systems. The following is a sample of the file:

```
ID = 4me

[XEE]
Name = XEE.cry
ERASE

[PGM]
ERASE
Name = PGM.hcry
```

The LOADER.CFG file has three sections, the global section (not specifically named), the XEE section and the PGM section. The XEE section contains bootloader commands for the external EEPROM bootloader function. The PGM section contains bootloader commands for the PIC program memory bootloader function.

The ID parameter is a three character, case sensitive, identifier used to identify the instance of the LOADER.CFG file. If the ID matched the equivalent field in the bootloader's parameter block, then no bootload takes place. The ID field in the bootloader's parameter block is updated after a successful bootload process.

The XEE and PGM sections contain the file name for the respective image and an optional ERASE command. The sections are optional. This enables either or both PIC and external EEPROM bootloading. The ERASE command specifies if the contents of the external EEPROM or PIC's user application space should be erased. The bootloader is an incremental bootloader enabling small code sections or the entire image in external EEPROM or the PIC's program memory to be updated.

The image filenames are limited to DOS 8.3 filename convention (does not support extended filenames).

Status LEDs:
The optional status LEDs provide visibility into the operation of the bootloader

LDR_MODE	set indicates the PIC is in bootloader mode
Status	flashes periodically to indicate the bootloader is running
PIC_PGM:	PIC Program memory update is in progress
XEE_PGM:	External EEPROM memory update is in progress

Customization
The bootloader must be customized to support different hardware plaforms (targets) and microcontrollers. In general the customisation requires the following steps:

- Define target hardware platform in the platform.h file
- In the Microchip IDE, select the processor in the the menu Configure / Select Device
- Create the processor specific bootloader linker script for compiling the bootloader. Refer to the sample bootloader linker scripts supplied with the bootloader package which contains instructions for modifying a standard linker script
- Create the processor specific bootloader linker script for compiling the user application to coreside with the bootloader. Refer to the sample application linker scripts supplied with the bootloader package which contains instructions for modifying a standard linker script
- Modify the bootloader's main source file to specify the target specific fuses

Limitations:
The file names of the PIC Intel hex file and the EEPROM Intel hex files are limited to DOS 8.3 file name convention. Extended filenames are not supported.

The bootloader firmware does not program the PIC's config fuses.

The user application code MUST implement a GOTO instruction (a long jump) in the first 2 instructions (4 program words). If the user's program fails to implement a GOTO instruction in the first 2 instructions, then a RESET instruction is executed. This requirement is met by a typical application linker script. Some development environments, such as the CCS PCD compiler does not use linker scripts, in this case a #build directive incorporate in the user application source code may be required.

Need Something Special?

What if you need some unique feature added to the Bootloader or a Bootloader developed for some other product? Brush Electronics specializes in the development of Bootloaders for Microchip Microcontrollers and welcome the opportunity to work with you to develop a custom product that meets your specific needs.

Brush Electronics
2 Brush Court
Canning Vale
Western Australia 6155
Australia

Tel: +61 (0) 894676358
Email: info@brushelectronics.com
www: www.brushelectronics.com

