v2.0 October 22, 2019

**NOTE:** Version 1.0 of the executable files had a bug which could cause them to flash the contents of a file marked in the directory as deleted. Those executables, dated April 19, 2019, should no longer be used. That bug has been fixed, and functional changes have also been made to accomodate "always-on" devices, as described below.

# SD Card Bootloader for MSP430

This is a system for updating the firmware of MSP430 microcontrollers using an SD or SDHC memory card. If an SD or microSD port is built into the circuit, the user can copy a downloaded update file to the SD card on his computer, then insert the card into the port, and power up the device. The bootloader will detect the card, then perform the update as it reads the data from the file via SPI. This eliminates the need for interface devices, cables or jumpers, and the need for any installed driver or software on the user's computer, or even the use of any particular operating system. However, the user will need an SD or microSD card slot on his computer, or a USB SD card reader so he can copy the downloaded file to the card.

The bootloader is installed by flashing it to the MSP430 using standard JTAG or BSL flashing. It occupies 1K at the beginning of MAIN memory. The reset vector at 0xFFFE must always point to the beginning of the bootloader. Thereafter, on power up, if no card is detected, the bootloader jumps to the application, which must begin at MAIN + 0x400 (the "NEWMAIN" address).

Included here in addition to the bootloader is the Windows console program SDBSL.exe, which takes in a standard Intel-HEX or TI-TXT file and creates the update file in the binary form needed by the bootloader. The C source code is included.

Bootloader examples included here are specifically for the MSP430G2452's USI module and the MSP430G2553's USCI-B module. It may be possible to adapt the code to other processors with those modules. The assembler source code is provided.

## Requirements and Limitations

To achieve the smallest footprint for the bootloader, and thereby make it usable in processors with little memory, practical limitations were adopted. This is what the bootloader supports:

1. **SD and SDHC Cards:** The bootloader works only with standard SD/SDHC and microSD/SDHC cards, not MMC or SDXC. Standard SD cards are available in various sizes from 128MB to 2GB, any of which is more than big enough for this process, but a larger SDHC card can also be used.

2. **Partition Format:** The SD card must be formatted with a Master Boot Record ("MBR"). The partition containing the update file must be the first entry in the partition table beginning at 0x01BE in the MBR, and the partition must be formatted as FAT16 or FAT32.

3. **File Storage:** Each file containing a firmware update must be stored in the Root Directory of the partition, and the file contents must be stored in consecutive sectors. Consecutive sectors will automatically result if the formatted cluster size is greater than the file size. For example, the G2553 update file is about 15KB. If the cluster size is 16KB, all update files will automatically be saved in the consecutive sectors of a single cluster. If a smaller cluster size is used, file sectors may still be consecutive if files are copied to a newly formatted card in sequence and nothing is ever deleted. With small (128MB and 256MB) SD cards being available for US$1.00 or less, it makes sense to include such a card with each copy of the project device. The developer can then make sure the card is formatted properly.

4. **File Format:** The update file must contain in binary form an image of the entire contents of NEWMAIN memory from the beginning of the flash segment following the end of the bootloader, which is the NEWMAIN address, up to, but NOT including, the reset vector at 0xFFFE, plus a one-byte XOR checksum. So for the G2553, the file must always be exactly 15,359 bytes long, and a G2452 file will always be 7167 bytes long.

5. **Firmware Format:** The firmware application must always begin with executable code at NEWMAIN. The bootloader will always jump to that address on power up if no card is detected.

6. **File Naming:** Files can be named anything, but the bootloader only looks at the short 8.3 DOS version of the filename, which always has uppercase letters. It searches the entire Root Directory, and selects the file, if any, which (1) is the right size, and (2) has the highest "value" in the filename's first four characters. The characters are compared using their ASCII values, and interpreted as a four-digit value, with the first character being the most significant. So best practice would be to put the version number at the beginning of the filename. This system allows the old versions to remain on the card in case it becomes necessary to reinstall an earlier version (the later version can be marked Hidden and the bootloader will not see it).

## SD Card Formatting

The bootloader does not implement any FAT traversal, and cannot follow cluster chains. Cards should therefore be formatted with the largest practical cluster size. The "Formatter" provided by the SD Association formats 2GB and smaller SD cards as FAT 16 with 16K clusters, and larger SDHC cards as FAT32 with 32K clusters. Use of the Formatter is strongly recommended. "Quick" format works fine. A card's current cluster size is reported by the CHKDSK function in Windows.

https://www.sdcard.org/downloads/formatter/index.html

## The Windows Application

The SDBSL.exe program converts a typical Intel-HEX or TI-TXT file output by a compiler or assembler into a binary file with the correct format for the bootloader. Your source code for a G2553 should be written with the application beginning execution at the NEWMAIN address C400, and should also set the reset vector at 0xFFFE to NEWMAIN. NEWMAIN for the G2452 is E400. SDBSL.exe will not actually include that reset vector in the binary file, but it will check to make sure it's there in the input file, and that something other than FFs are stored at NEWMAIN - it will error out if that's not the case. There should be no content specified for Information Memory or MAIN memory occupied by the bootloader itself because the bootloader can only flash NEWMAIN memory. SDBSL ignores the checksum byte found at the end of each Intel-HEX line. SDBSL is a Windows command line utility with the following format:

```
SDBSL.exe  {Infile}  {Outfile}  {NEWMAIN}
  Example for a G2553:
SDBSL.exe  v3.2Firmware.hex  v3.2Firmware.bin  C400
```

## Update Procedure

1. Compile or assemble the new firmware version to an Intel-Hex or TI-txt file.

2. Run SDBSL.exe to produce the corresponding binary output file ( usually a .bin).

3. Distribute the .bin file to users.

4. The user copies the .bin file to the SD card, inserts the card into the SD port, and powers up.

5. The bootloader detects the card, selects the file, and then, segment by segment, it erases flash memory and writes the new contents to it. It also saves its own entry point (C000 or E000) as the Reset Vector. The indicator LED lights up if flashing is completed successfully.

6. After flashing, the user should remove the card from the holder before power cycling the processor or pressing a Reset button. Otherwise the bootloader will install the new firmware again when it starts up. If it is desireable to store the card in the device, there should be a switch or jumper on the card's Vdd line which turns off power to the card except when updating is needed.

## Soft Power Switch

A developer may may wish to have his device power up by pushing a momentary button, then have a GPIO pin take over maintaining power and eventually automatically shutting itself off. The bootloader has no code to support that, and there may not be enough room left within the 1K bootloader space to add it. In such a case, it appears the best solution is to have the user simply hold down the momentary button until updating has completed - about 3 seconds for the G2553, or half that for the G2452. Then the button would be released to turn off the power, at which time the user can remove the SD card.

## Always-On Devices

It is best to power-cycle the device after flashing, or press a Reset button, so the new firmware can start up on a fully-reset device (known as a "power-on-reset" or "POR"). But there may be designs which contain no ON/OFF switch or Reset button, and the device spends idle time in sleep mode and never shuts down. Version 2 of this BSL has been modified to support these devices in case it is not convenient to disconnect the battery or otherwise turn off power. But supporting code will also be needed in the firmware to make this work.

Version 1 shut down the processor when flashing had been completed, thus requiring a power cycle or Reset. But Version 2 instead lights an LED, then waits for the card to be removed. At that point, the user can turn off power, then remove the card, or hold down the Reset button while removing the card, and the result will be a clean POR boot into the new firmware.

But for always-on devices, the user can instead simply remove the card while the power is still on. The BSL will detect that, and exit by generating a "power-up-clear" ("PUC"), which re-initializes many registers, then jumps to the Reset address at 0xFFFE, which of course points to the BSL. But this time the card is NOT present, so BSL jumps to the new firmware at NEWMAIN.

But be warned that a PUC leaves certain processor registers unchanged, such as those for TimerA, TimerB, ADC10 and ADC12, including any interrupts enabled on those modules. So if a PUC is to be relied upon, the new firmware must deal with anything that might have been set up by the old firmware and might still be in effect. BSL itself changes only registers which ARE re-initialized by a PUC (the GPIO and SPI-related USI or USCI registers).

If there is to be no power cycle or Reset, there still must be a way to get BSL to run in the first place. The solution is that code must be included in all versions of the firmware to detect the presence of an SD card, either by polling or interrupt, wait a sufficient time to allow for any bouncing, turn off anything that has been set up that will not be cleared by a PUC and that might interfere with BSL (interrupts in particular), then initiate a PUC (which can be done by illegally writing a value to WDTCTL without the password in the upper byte). When BSL runs after the PUC, the card will be present, and flashing will proceed.

In case it is needed for the first flashing after the BSL is installed, the installer also contains a small "firmware" program at NEWMAIN which just detects when a card has been inserted, then initiates a PUC. See the end of the installer source code for details of that firmware.

## Other MSP430 Parts

The code presented here should also work with minor changes for other MSP430 parts which have USI or USCI modules. It may also work with modifications for parts with USART modules. However, the examples included here fit into 1K of flash memory with only a few bytes to spare, so it is not certain that any other version would fit within the two-segment space. If a new version of the BSL extends past the 1K boundary, the definition of NEWMAIN will have to be changed to MAIN + 0x600 to allocate an additional segment to the BSL.

The bootloader is written in assembler for the Naken Assembler, which is a free, small, simple command-line assembler written by Michael Kohn. Many thanks to Mike.

https://www.mikekohn.net/micro/naken_asm.php

https://github.com/mikeakohn/naken_asm

## Circuit Design

SD cards are 3.3V devices, and are therefore compatible with MSP430 parts. Four GPIO pins are used to connect the processor to the card. Communication is via SPI. For the G2553, the USCI-B0 module was chosen, which uses P1.5, P1.6 and P1.7 for SPI . The additional Card Select ("CS") pin can be any other I/O pin - P1.4 is used in the source code here. For the G2452, the USI module is used. It uses the same pins, but the functions of the P1.6 and P1.7 data pins are reversed. P1.3 is used for an LED indicator for both processors. The CS and LED pin assignments can be anything, but changing them from P1.4 and P1.3 would require reassembling the source. The assignments are defined at the very beginning of the source code.

The processor's CS pin is connected to the SD card's Card Detect ("CD") pin, which has a nominal 50K internal pullup resistor. In the beginning, CS is configured as an input pin so the voltage on that pin can be used to detect the presence of an SD card. To prevent the CS pin from floating when a card is not present, an external 1Meg pulldown resistor must be used. After the card is detected, the CS pin is reconfigured as an output pin to be used when sending commands to the card.

To be safe, a 47K external pullup resistor may be placed on the processor's data input pin (SOMI). Some cards may configure this line as an open-drain output, which must be externally pulled high.

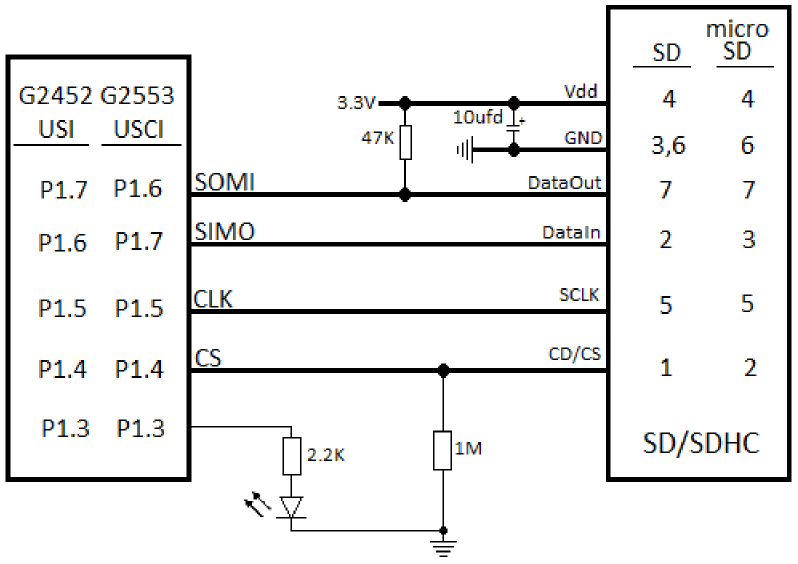
Current drawn by the cards tested here ranged from 10mA for an Ebay $1 no-name 256MB microSD card, to 20mA for a Transcend 4GB microSDHC card. It appears most cards draw 30ma or less during reading (the bootloader does not write to the card). Some card brands automatically shut down to a microamp current range after an idle period, but others continue to draw 20mA or more so long as they are powered up. Best practice is to remove the card or disconnect Vdd after flashing.

The LED is optional, but if present it will light up when flashing has completed successfully. If the LED does not light within about five seconds, something has gone wrong:

1. The wrong type of SD card was used (SDXC, for example).
2. The card is formatted with something other than FAT16 or FAT32
3. No file of the right size was found in the Root Directory
4. Flashing was completed, but the XOR checksum did not match.
5. There is an error in the wiring.
6. Insufficient power to the SD card.

Below is a diagram showing how the processor and the SD card are connected:

## SD CARD BOOTLOADER



G2452 G2553
USI USCI
P1.7 P1.6
P1.6 P1.7
P1.5 P1.5
P1.4 P1.4
P1.3 P1.3
SOMI
SIMO
CLK
CS
3.3V
47K
10ufd
2.2K
1M
Vdd
GND
DataOut
DataIn
SCLK
CD/CS
SD
micro SD
4 4
3,6 6
7 7
2 3
5 5
1 2
SD/SDHC