BCB Booting on an SD Card

What the ROM is looking for

For BCB boot, the ROM will go to the last sector (the final 512 bytes of the SD Card) to look for the boot config block. The boot config block contains the following information in and that information must be formatted in a specific way (which we'll go into later):

- Config Block Signature - This is always 0x00112233, just to let the ROM know it's dealing with a config block.
- Config Block Version - For our example, this was 0x1. Maybe future versions of the ROM will change this (thus requiring this example to change). For now, don't change this.
- Boot Image Tag - This is always 0x50. It tells the ROM the address of the firmware.sb bootable image is nearby.
- Number of Drives - This contains the number of drives on the medium. I set it to 0x4 for our example, though we only have 1 drive. I believe this is only useful in the ROM if you have multiple drives.
- Starting Block/Sector of the bootable image - This is adjacent to the Boot Image Tag and for our example, is the ONLY thing you need to customize for the BCB config block.

For the example used on this document, the creation of this block is handled by the bcb.c file.

Step by Step Process for Flashing SD Card

Follow this guide to use BCB boot.

Step1 - Verify your chip is configured for BCB Boot

BCB Boot. However, due to the popularity of MBR boot, many development systems have had their OTP bit blown to enable MBR boot. This is irreversible, and you cannot use BCB boot after this happens (unless you step through the ROM code in the debugger and change the program counter to jump to the BCB routine, etc).

To verify which boot mode you are using, you need a tool to view/modify OTP bits, such as BitBurner.exe. Look at the OTP bit HW_OCOTP_ROM0.SD_MBR_BOOT(3). If it is BLOWN, you are using MBR boot and can't use BCB boot ever on that board. If it is NOT blown, you are set up for BCB boot (and can never use MBR boot on that system).

Step2 - Obtain Custom SD-card information

Now it's time to figure out how many sectors your SD Card has. Plug your SD card into a Linux host, and execute this command:

```
fdisk -u /dev/sdX
```

The output for my *unformatted* 128MB Kingston SD card (the card we're using as the example), looked like this:

```
fsl@fsl-desktop:~/ernie/writefile$ sudo fdisk -ul /dev/sdf

Disk /dev/sdf: 125 MB, 125698048 bytes

4 heads, 60 sectors/track, 1022 cylinders, total 245504 sectors

Units = sectors of 1 * 512 = 512 bytes

Disk identifier: 0xddddfc4a

Device Boot      Start         End      Blocks   Id  System
```

Make a note of the number of total sectors, 245504. Each one is 512 bytes. The last one, the 245504th one, will contain our BCB config block.

Now let's execute:

```
sudo fdisk -u /dev/sdX
```

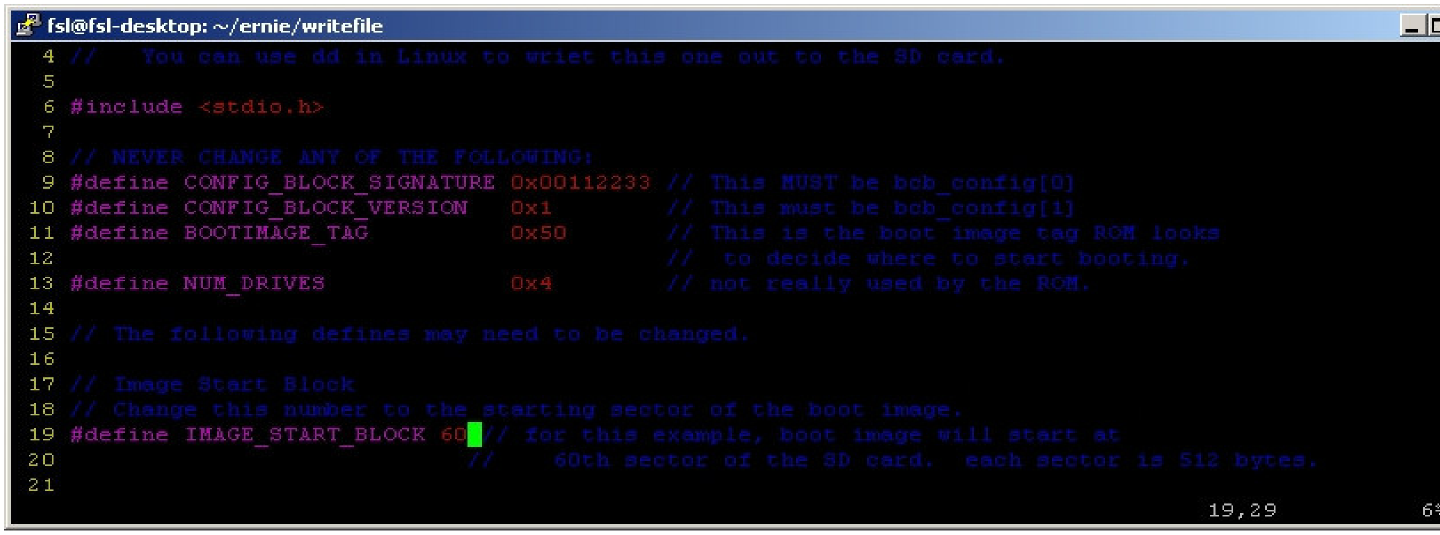
Create a new partition, at the first writable sector of the card (type n, then 1, then p). For our card, it is 60, however for other cards it may be different. Since we are writing the firmware.sb to this location, it is important to make note of it now. Also note that you don't have to put your image here, you could put it anywhere on the card, as long as it fits and doesn't overwrite the last BCB config sector. The point is to note the sector offset of where you're putting your firmware.sb (again, for our example, 60).

```
Command (m for help): n

Command action

   e   extended

   p   primary partition (1-4)

1

Invalid partition number for type `1'

Command action

   e   extended

   p   primary partition (1-4)

p

Partition number (1-4): 1

First sector (60-245503, default 60):
```

Step3 - Create a Custom BCB config block

The config block at the last 512 bytes of your card will be different for every SD card. This step shows you how to customize it using the attached bcb.c file on a Linux host. Open up bcb.c in an editor and change

IMAGE_START_BLOCK to the starting sector of your firmware.sb file. For the example on this page, that number is 60.



fsl@fsl-desktop: ~/ernie/writefile

```
//    You can use dd in Linux to wriet this one out to the SD card.

#include <stdio.h>

// NEVER CHANGE ANY OF THE FOLLOWING:
#define CONFIG_BLOCK_SIGNATURE 0x00112233 // This MUST be bcb_config[0]
#define CONFIG_BLOCK_VERSION   0x1        // This must be bcb_config[1]
#define BOOTIMAGE_TAG          0x50       // This is the boot image tag ROM looks
                                          //  to decide where to start booting.
#define NUM_DRIVES             0x4        // not really used by the ROM.

// The following defines may need to be changed.

// Image Start Block
// Change this number to the starting sector of the boot image.
#define IMAGE_START_BLOCK 60// for this example, boot image will start at
                           //    60th sector of the SD card.  each sector is 512 bytes.

                                                                                19,29
```

Now you can compile the file and obtain the new BCB config block file.

```
gcc bcb.c -o bcb
```

```
./bcb
```

The output will be called bcb_512.cfg.

Step4 - Flash your SD card

Now it's time to prepare the SD card. You can partition the SD card however you want, so long as the last 512 bytes are reserved for the config block we made in Step3. For our example, we'll place the Linux boot image at partition 1, reserve the rootfs for partition 2, and place the bcb config block at partition 3.

First, we need to format the card for the 3 partitions:

```
sudo fdisk -u /dev/sdX

n

p

1

32000

t

53

n

p

2
```

```
(Press Enter to accept the default)

For our example use 245502, leaving 1 final sector available for BCB config
block.

n

p

3

(Press Enter to accept all the defaults).

w
```

Your SD Card now looks like this for our example:

```
Disk /dev/sdf: 125 MB, 125698048 bytes

4 heads, 60 sectors/track, 1022 cylinders, total 245504 sectors

Units = sectors of 1 * 512 = 512 bytes

Disk identifier: 0xddddfc4a


   Device Boot      Start         End      Blocks   Id  System

/dev/sdf1              60       32000       15970+  53  OnTrack DM6 Aux3

Partition 1 does not end on cylinder boundary.

/dev/sdf2           32001      245502      106751   83  Linux

Partition 2 does not end on cylinder boundary.

/dev/sdf3          245503      245503           0+  83  Linux

Partition 3 does not end on cylinder boundary.
```

Now we can flash our images:

```
# Flash the Linux boot image.

sudo dd if=firmware.sb of=/dev/sdX1


# Flash the BCB config

sudo dd if=bcb_512.cfg of=/dev/sdX3


# Mounting sdX2 and copying rootfs is left as an exercise to the reader!!!
```

You are now ready to boot your BCB image on the i.MX233 platform.