Afridev V2 Firmware
In-Field Firmware Upgrade Overview

Prepared by David Laone
MindInspire Inc.

Last Update
May 15, 2018

Table of Contents

Glossary of Terms

Firmware Upgrade Message	An OTA message sent by the cloud server to the Afridev unit. It contains a complete Application firmware code set.
Bootloader	An independent and self-contained code image that runs when the MSP430 first starts.
Application	An independent and self-contained code image that runs on the MSP430. The Bootloader starts the Application. It performs all water flow algorithm and data storage processing.
SOS Recovery Mode	A mode the Bootloader will drop into if it detects that there is not a valid Application to jump to.
SOS Recovery Message	A message the Bootloader will send to the Cloud every 12 hours when in SOS Recovery Mode.

1 In-filed Firmware Upgrade In-A-Nutshell

When an in-filed firmware upgrade is performed, the new firmware image is downloaded by the Application and stored in the Secondary Application Code Block. Once downloaded successfully, the Application will reboot to start the Bootloader. The Bootloader copies the new firmware image from the Secondary Application Code Block to the Primary Application Code Block and jumps to it.

Of course, there are a lot of details involved. But in-a-nutshell, that is what happens.

2 The Afridev V2 Bootloader

- The Bootloader is a standalone firmware module that runs on the Afridev V2 MSP430 microcontroller
- The Bootloader coexists in the MSP430 flash memory with the Afridev V2 Application code
- The Bootloader is a small program that runs every time the MSP430 starts
- The purpose of the Bootloader is to provide in-field software upgrade capability
- The Bootloader logic supports three operational scenarios:
    - Standard Boot
    - Firmware Upgrade
    - SOS Recovery Mode

3 In-Field Firmware Upgrade: Data Details

An in-field firmware upgrade sequence involves the (1) Application, the (2) Bootloader and (3) data. This section presents the different data pieces that come into play when performing an in-field firmware upgrade.

3.1 Afridev V2 MSP430 Memory Organization

The MSP430 FLASH and RAM organization is shown in Figure 1. Some key blocks to note are:

- The Bootloader
- The Primary Application Code Block
- The Secondary Application Code Block
- The Application Record
- The Bootloader Record

<u>MSP430G2995 Memory Summary</u>

- Total Available RAM: 4K (0x1100 – 0x20FF)
- Total Available FLASH 55K (0x2100 – 0xFFFF)

0xE000–0xFFFF	FLASH, 8K (0x2000)	**Boot Loader** *Non-Upgradable after ship*

0x9000–0xDFFF	FLASH, 20K (0x5000)	**Primary Application Code Block** This is where the Application image lives. *In-Field Upgradable*.
0x7100–0x8FFF	FLASH, 7936 (0x1F00)	**Storage Data (Data Statistics, etc.)** One week's worth of water data is stored in a 1K block. This space allows up to 7 weeks of data to be stored on the Afridev V2 unit.
0x2100–0x70FF	FLASH, 20K (0x5000)	**Secondary Application Code Block** This is used as a temporary holding area for saving a complete application code image during an in-field firmware upgrade
0x10C0–0x10FF	64 bytes (0x40)	INFO A FLASH (MSP430 Calibration)
0x1080–0x10BF	64 bytes (0x40)	INFO B FLASH (Bootloader Record)
0x1040–0x107F	64 bytes (0x40)	INFO C FLASH (Application Record)
0x1000–0x103F	64 bytes (0x40)	INFO D FLASH
0x2000–0x20FF	RAM, 256 bytes (0x100)	STACK
0x1100–0x2000	RAM, 3840 bytes (0xF00)	RAM
0x0100–0x01FF	256 (0x100)	16-bit Peripherals
0x1010-0x0FFF	240 (0xF0)	8-bit Peripherals
0x0000–0x000F	16 bytes (0x10)	SFR

Figure 1 MSP430 Memory Allocation

3.2 The Firmware Upgrade Message (message ID 0x10)

The Firmware Upgrade Message is sent by the Cloud to the Afridev unit and contains a complete Application Firmware Image. When the Sensor receives the message, it will initiate a firmware upgrade sequence. The format of the Firmware Upgrade Message is shown in Figure 2. The upgrade message contains a Code Section Header and a new Application Image. The Code Section Header contains information about the Application Image that is used during the upgrade sequence.

Message Details:

Byte Location	Name	Description
0	Message Number	0x10
1-2	Message ID	2 byte ID

3-6	Message Keys (magic numbers)	key0 = 0x31
		key1 = 0x41
		key2 = 0x59
		key3 = 0x26

7	Number of Code Sections	For Afridev, this will always be 1
8	Code Section Start Byte	0xA5
9	Code Section Number	For Afridev, this will

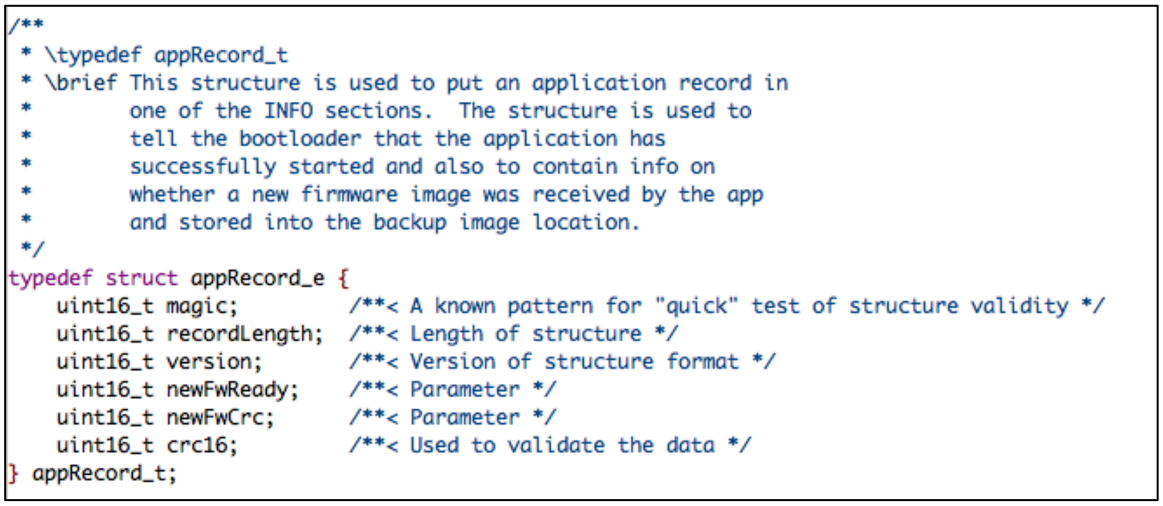
		always be 0
10-11	Code Section Start Flash Address	For Afridev, this will always be 0x9000
12-13	Code Section Length	Length in Bytes. For Afridev, this will always be 0x5000 (20K)
14-15	Code Section CRC16	Polynomial: 0x8005
16+	Application Image	The application ROM image.

Figure 2 Firmware Upgrade Message Format

3.3 The Application Record

The Application Record is a data structure stored in INFO section C on the MSP430. It is used to inform the Bootloader of two items:

1. That the Application successfully started
2. Whether a new firmware image was received by the Application and stored in the Secondary Image location



```
/**
 * \typedef appRecord_t
 * \brief This structure is used to put an application record in
 *        one of the INFO sections.  The structure is used to
 *        tell the bootloader that the application has
 *        successfully started and also to contain info on
 *        whether a new firmware image was received by the app
 *        and stored into the backup image location.
 */
typedef struct appRecord_e {
    uint16_t magic;         /**< A known pattern for "quick" test of structure validity */
    uint16_t recordLength;  /**< Length of structure */
    uint16_t version;       /**< Version of structure format */
    uint16_t newFwReady;    /**< Parameter */
    uint16_t newFwCrc;      /**< Parameter */
    uint16_t crc16;         /**< Used to validate the data */
} appRecord_t;
```

Figure 3 The Application Record Data Structure

3.3.1 Application Run-Time Verification

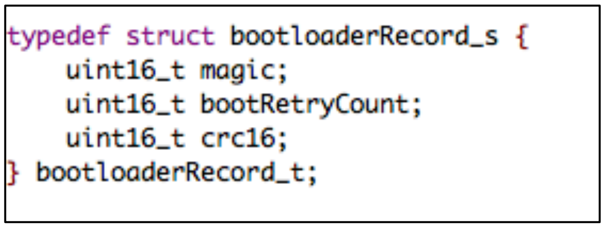
The Bootloader erases the Application Record every time it runs before jumping to the Application Image. The Application writes a new Application Record after it sends the Final Assembly message. If the Bootloader does not detect a valid Application Record when it runs, it assumes the Application did not run successfully. The Bootloader will try to re-run the Application up to four times before dropping into SOS mode.

3.3.2 New Application Image Information

When a new Application image is successfully downloaded via the Firmware Upgrade Message into the Secondary Application Code Block, the Application will add information to the Application Record regarding the new image. Every time the Bootloader runs, it checks the Application Record to identify if a new image is available and if so, the corresponding CRC of the stored image. If a new Application Image is ready, the Bootloader will proceed to update the Primary Application Code Block using the image located in the Secondary Image Code Block.

3.4 The Bootloader Record

The Bootloader maintains a data structure located in INFO section B of the MSP430. The structure contains a counter of how many times the Bootloader has detected that there is no Application Record stored before starting the Application. If the counter reaches four, then the Bootloader will fall into its SOS mode instead of jumping to the Application Image start.



```
typedef struct bootloaderRecord_s {
    uint16_t magic;
    uint16_t bootRetryCount;
    uint16_t crc16;
} bootloaderRecord_t;
```

Figure 4 Bootloader Record Data Structure

4 Firmware Upgrade Sequence

The Firmware Upgrade sequence involves both the Application and the Bootloader. The Application is responsible for receiving the Firmware Upgrade Message from the modem, and storing the new image to the Secondary Application Code Block. Once complete, the Bootloader is responsible for moving the image from the Secondary Application Code Block to the Primary Application Code Block. The following sections breakdown the steps performed by the Application and the Bootloader in more detail.

4.1 Application Firmware Upgrade Sequence

When the Application detects that a Firmware Upgrade Message is available from the modem, it will sequence into its Firmware Upgrade mode. In this mode, the Application performs the following:

- Downloads the Firmware Upgrade Message Code Section Header
- The Code Section Header identifies the Starting Address, Length and CRC16 of the Code Section
- The Application erases the Secondary Application Code Block section in flash
- The Application starts downloading the code section from the Modem. It retrieves 512 bytes at time from the modem and writes that buffer to flash.

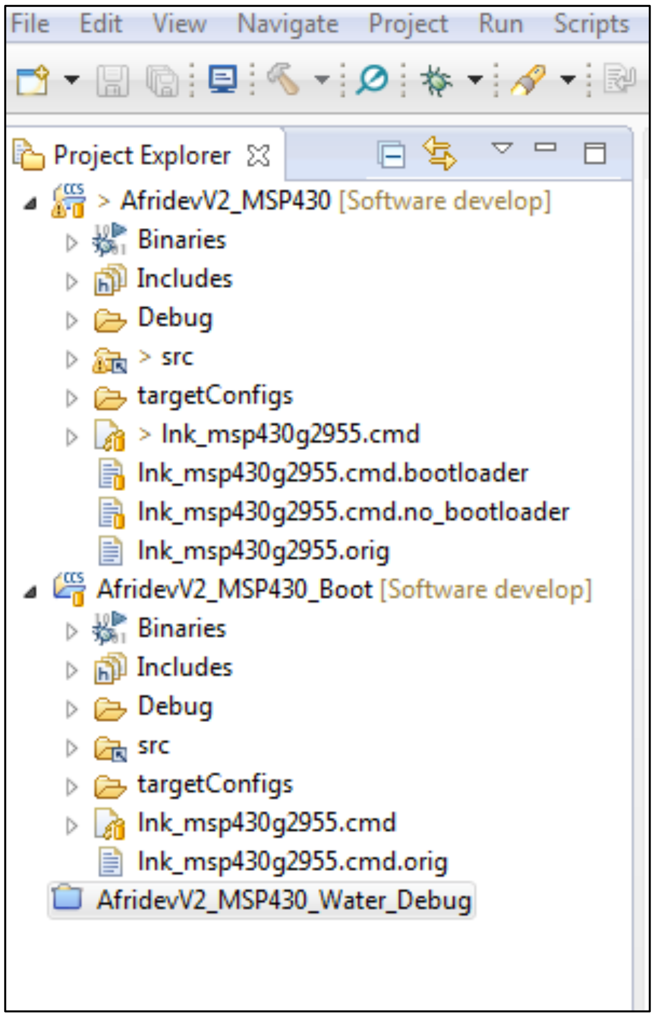
- Once all bytes of the Code Section have been read from the Modem and written to flash, the Application verifies the code by calculating a CRC16 sum and comparing it to the CRC16 provided in the Section Header
- If the CRC16 check matches, The Application updates the Application Record with the new firmware information and starts a 20 second reboot countdown sequence
- The Application sends an OTA reply in response to the Firmware Upgrade Message (pass or fail)
- The Application re-boots to start the Bootloader

4.2 Bootloader Firmware Upgrade Sequence

- Each time the Bootloader starts it checks the Application Record area to identify if there is a new firmware image available
- If there is a new firmware image available, it verifies it against the CRC16 stored in the Application Record
- If verified, it zeros the Application reset vector. Should the Bootloader reboot prematurely before the upgrade sequence is complete, this is one method the Bootloader uses to identify that there is not a valid Application image flashed.
- The Bootloader erases the Primary Application Code Block section in flash
- The Bootloader copies the Secondary Application Code Block to the Primary Application Code Block
- The Bootloader verifies that the Primary Application Code Bock exactly matches the Secondary Application Code Block
- The Bootloader verifies the Primary Code Block against the CRC16 stored in the Application Record
- The Bootloader updates the Bootloader Record
- The Bootloader Erases the Application Record
- The Bootloader Jumps to the entry point of the Application

5 Building the Firmware

The Code Composer Studio IDE from Texas Instruments is used to build the Afridev V2 firmware. At a minimum, there are two required projects: The Application and the Bootloader. For this example, these are named AfridevV2_MSP430 and AfridevV2_MSP430_Boot as shown in Figure 5.



```
▲ > AfridevV2_MSP430 [Software develop]
    ▷ Binaries
    ▷ Includes
    ▷ Debug
    ▷ > src
    ▷ targetConfigs
    ▷ > lnk_msp430g2955.cmd
      lnk_msp430g2955.cmd.bootloader
      lnk_msp430g2955.cmd.no_bootloader
      lnk_msp430g2955.orig
▲ AfridevV2_MSP430_Boot [Software develop]
    ▷ Binaries
    ▷ Includes
    ▷ Debug
    ▷ src
    ▷ targetConfigs
    ▷ lnk_msp430g2955.cmd
      lnk_msp430g2955.cmd.orig
  AfridevV2_MSP430_Water_Debug
```

Figure 5 Code Composer Application and Bootloader Projects

5.1 Building the Application Project for use with the Bootloader

When the Application is used with the Bootloader, it does not get built in the standard fashion. When the Application is built for use with the Bootloader, the project is modified as follows:

1. The Application is linked to a different address range
2. The Application is built to use a "proxy" interrupt vector table

When making the Application build, there are two steps that must be made to the project to address items 1 and 2 above. These include:

1. Using the correct linker command file
2. Defining the correct MACRO symbol in the project settings, which modifies the Application to use the "proxy" interrupt vector table.

5.2 Application Linker Command File

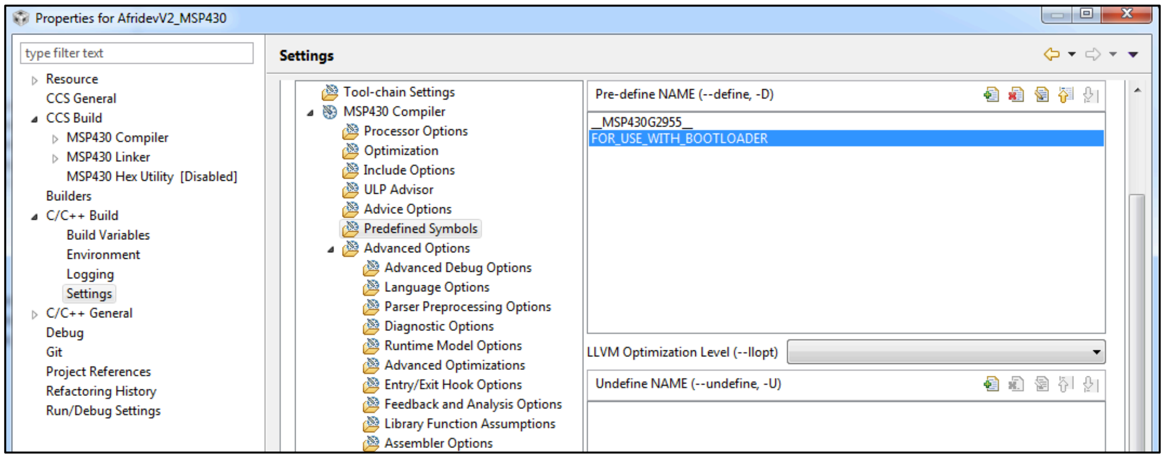
The linker command file that is used when building the Application for use with the Bootloader is different from the standard linker file that is used for a “standalone” Application. In Figure 5, under the AfridevV2_MSP430 project, you will see four linker command files:

- lnk_msp430g2955.cmd
- lnk_msp430g2955.cmd.bootloader
- lnk_msp430g2955.cmd.no_bootloader
- lnk_msp430g2955.orig

The lnk_msp430g2955.cmd file is used by Code Composer to perform the linking step during the AfridevV2_MSP430 project build. The two linker files with names ending in bootloader and no_bootloader are, as their names suggest, the linker files to “copy” for the different build types. The correct linker file must be manually copied to the “lnk_msp430g2955.cmd“ name when moving between built types. To build the Application for use with the Bootloader, the lnk_msp430g2955.cmd file must be a copy of the lnk_msp430g2955.cmd.bootloader file. Likewise, when building the Application to run standalone (i.e. no Bootloader), the lnk_msp40g29555.cmd file must be a copy of the lnk_msp430g2955.cmd.no_bootloader file.

5.3 Setting the “FOR_USE_WITH_BOOTLOAER” Macro

In order to have the Application use the proxy interrupt vector table when running with the Bootloader, the MACRO “FOR_USE_WITH_BOOTLOADER” must be defined during the build process. Defining the MACRO can be performed by adding the symbol to the project->properties->settings->predefined symbols menu. This is shown in Figure 6.



Pre-define NAME (--define, -D)
__MSP430G2955__
FOR_USE_WITH_BOOTLOADER

LLVM Optimization Level (--llopt)

Undefine NAME (--undefine, -U)

Figure 6 Setting the "FOR_USE_WITH_BOOTLOADER" MACRO

5.3.1 Proxy Interrupt Table Overview (For information only)

A "proxy" interrupt table is used to "redirect" the standard MSP430 interrupt vectors to the appropriate interrupt routines within the Application code. This technique is based on a TI Bootloader design. Please see document SLAA600A for a description of this scheme.

The MSP430 uses dedicated hardware interrupt vectors. The CPU grabs the address at the vector location and jumps to it. Because these locations are fixed according to the CPU design, a proxy method must be used so that the Application can use interrupts based on its own needs. Each vector in the standard MSP430 interrupt vector table is setup so that it points to the locations specified in the Application's proxy table.

Each entry in the proxy table contains a BRA instruction (0x4030) followed by the address of the appropriate Application ISR processing routine. When a hardware interrupt occurs, the MSP430 interrupt processing is invoked which grabs the address in the standard MSP430 interrupt vector table (setup by the Bootloader). These vectors point to locations within the Application proxy table. When the MSP430 starts executing the code located at the proxy table entry, it will be redirected to the appropriate Application ISR. If an interrupt vector is not used by the Application, the proxy table entry for that entry points to a "Dummy" interrupt stub function.

5.4 Building the Bootloader

The Bootloader can be built using the Code Composer IDE as a "standalone" project. There are no special requirements for building the Bootloader as there are with the Application.

6 Creating a Combined Bootloader plus Application Image

The Bootloader and the Application images can be combined to form a single file that can be flashed to the MSP430 all at once. A script file has been created to perform the combining of the two files (Bootloader and Application). The following section (Creating the Firmware Upgrade Message) describes how to perform this step.

7 Creating The Firmware Upgrade Message

To support creating the Firmware Upgrade Message, a python script was developed. The python script takes the Application image file, and prepends the Firmware Upgrade Message header.

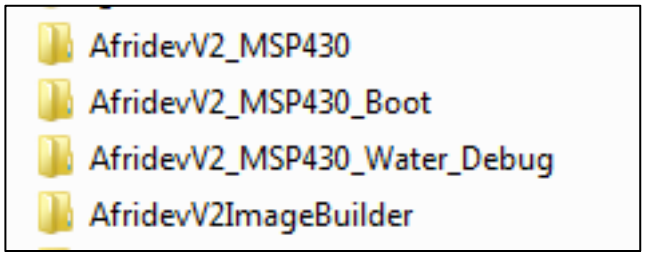
The script also creates a single file containing both the Bootloader and the Application that can be used to program the MSP430 using a JTAG tool such as Code Composer and the Elprotronic FET-Lite.

Once the Application and the Bootloader have been built, a ROM file containing their image will exist in their perspective Code Composer output directory. For the purpose of this example, they are located as follows:

- Software\AfridevV2_MSP430\CCS_AfridevV2_MSP430\Debug\AfridevV2_MSP430.txt
- Software\AfridevV2_MSP430_Boot\CCS_AfridevV2_MSP430_Boot\Debug\AfridevV2_MSP430_Boot.txt

These output files are created using the "hex30" utility as part of the last step of the Code Composer build for each project.

The script files used to create (1) the single image and (2) the firmware upgrade message are located in a directory adjacent to the project directories. The directory is named "AfridevV2ImageBuilder".



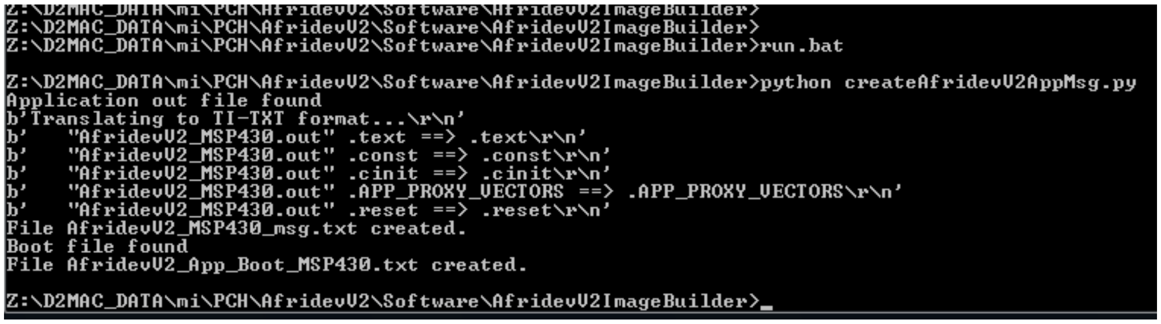
AfridevV2_MSP430
AfridevV2_MSP430_Boot
AfridevV2_MSP430_Water_Debug
AfridevV2ImageBuilder

Figure 7 Software Directories

The AfridevV2ImageBuilder directory contains the following scripts and support files.

File Name	Description
run.bat	Convenience batch file
createAfridevV2AppMsg.py	Top level Python script
afridevV2RomToMsg.py	Support Python script, called by createAfridevV2AppMsg.py
afridevV2_app_to_rom.cmd	Support file, used by the createAfridevV2AppMsg.py script

To create the support images, simply execute the “run.bat” from a command window. The batch file must be executed within the AfridevV2ImageBuilder directory:



```
Z:\D2MAC_DATA\mi\PCH\AfridevV2\Software\AfridevV2ImageBuilder>
Z:\D2MAC_DATA\mi\PCH\AfridevV2\Software\AfridevV2ImageBuilder>
Z:\D2MAC_DATA\mi\PCH\AfridevV2\Software\AfridevV2ImageBuilder>run.bat

Z:\D2MAC_DATA\mi\PCH\AfridevV2\Software\AfridevV2ImageBuilder>python createAfridevV2AppMsg.py
Application out file found
b'Translating to TI-TXT format...\r\n'
b'    "AfridevV2_MSP430.out" .text ==> .text\r\n'
b'    "AfridevV2_MSP430.out" .const ==> .const\r\n'
b'    "AfridevV2_MSP430.out" .cinit ==> .cinit\r\n'
b'    "AfridevV2_MSP430.out" .APP_PROXY_VECTORS ==> .APP_PROXY_VECTORS\r\n'
b'    "AfridevV2_MSP430.out" .reset ==> .reset\r\n'
File AfridevV2_MSP430_msg.txt created.
Boot file found
File AfridevV2_App_Boot_MSP430.txt created.

Z:\D2MAC_DATA\mi\PCH\AfridevV2\Software\AfridevV2ImageBuilder>_
```

The scripts expect the Bootloader and Application ROM files to be in their perspective directories and will fail if it cannot locate them. After running the run.bat batch file, two resulting files are created:

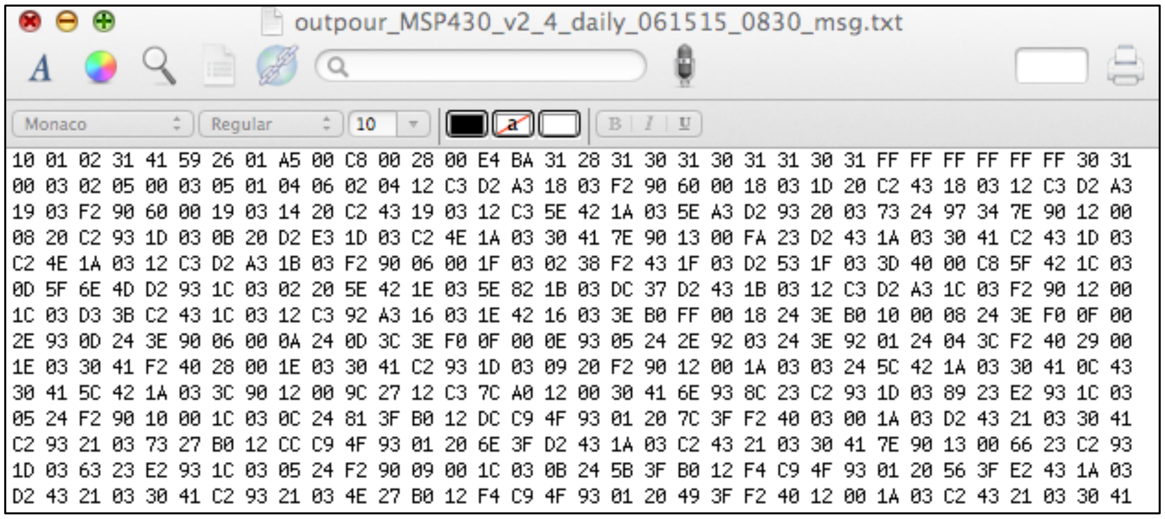
1. AfridevV2_App_Boot_MSP430.txt
2. AfridevV2_MSP430_msg.txt

As the names suggest, one file is the combined Bootloader and Application image, and one file is the Firmware Upgrade Message.

8 How to Perform an In-Field Firmware Upgrade

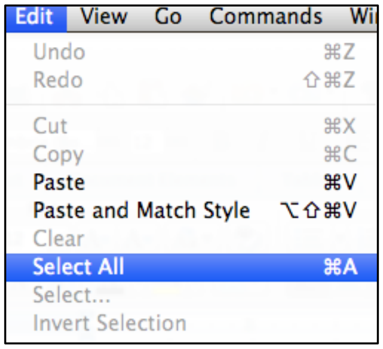
1. Obtain the Firmware Upgrade Message file. This is a text file that contains a complete Application image wrapped in a Firmware Upgrade Message. For example, with the Outpour version 1.4 release, this file is called: "outpour_MSP430_v1_4_daily_061515_0830_msg.txt"

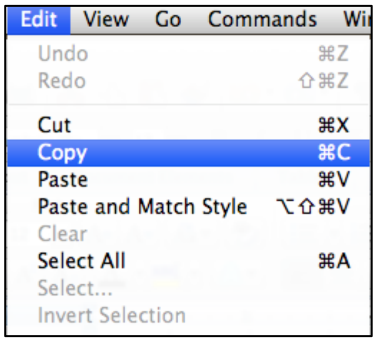
2. Using a Text Editor, open the Firmware Upgrade Message:



```
10 01 02 31 41 59 26 01 A5 00 C8 00 28 00 E4 BA 31 28 31 30 31 30 31 31 30 31 FF FF FF FF FF FF 30 31
00 03 02 05 00 03 05 01 04 06 02 04 12 C3 D2 A3 18 03 F2 90 60 00 18 03 1D 20 C2 43 18 03 12 C3 D2 A3
19 03 F2 90 60 00 19 03 14 20 C2 43 19 03 12 C3 5E 42 1A 03 5E A3 D2 93 20 03 73 24 97 34 7E 90 12 00
08 20 C2 93 1D 03 0B 20 D2 E3 1D 03 C2 4E 1A 03 30 41 7E 90 13 00 FA 23 D2 43 1A 03 30 41 C2 43 1D 03
C2 4E 1A 03 12 C3 D2 A3 1B 03 F2 90 06 00 1F 03 02 38 F2 43 1F 03 D2 53 1F 03 3D 40 00 C8 5F 42 1C 03
0D 5F 6E 4D D2 93 1C 03 02 20 5E 42 1E 03 5E 82 1B 03 DC 37 D2 43 1B 03 12 C3 D2 A3 1C 03 F2 90 12 00
1C 03 D3 3B C2 43 1C 03 12 C3 92 A3 16 03 1E 42 16 03 3E B0 FF 00 18 24 3E B0 10 00 08 24 3E F0 0F 00
2E 93 0D 24 3E 90 06 00 0A 24 0D 3C 3E F0 0F 00 0E 93 05 24 2E 92 03 24 3E 92 01 24 04 3C F2 40 29 00
1E 03 30 41 F2 40 28 00 1E 03 30 41 C2 93 1D 03 09 20 F2 90 12 00 1A 03 03 24 5C 42 1A 03 30 41 0C 43
30 41 5C 42 1A 03 3C 90 12 00 9C 27 12 C3 7C A0 12 00 30 41 6E 93 8C 23 C2 93 1D 03 89 23 E2 93 1C 03
05 24 F2 90 10 00 1C 03 0C 24 81 3F B0 12 DC C9 4F 93 01 20 7C 3F F2 40 03 00 1A 03 D2 43 21 03 30 41
C2 93 21 03 73 27 B0 12 CC C9 4F 93 01 20 6E 3F D2 43 1A 03 C2 43 21 03 30 41 7E 90 13 00 66 23 C2 93
1D 03 63 23 E2 93 1C 03 05 24 F2 90 09 00 1C 03 0B 24 5B 3F B0 12 F4 C9 4F 93 01 20 56 3F E2 43 1A 03
D2 43 21 03 30 41 C2 93 21 03 4E 27 B0 12 F4 C9 4F 93 01 20 49 3F F2 40 12 00 1A 03 C2 43 21 03 30 41
```

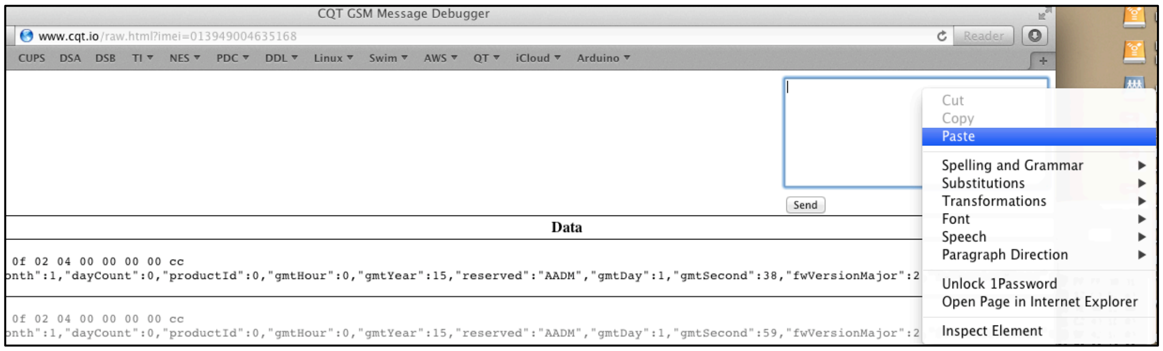
3. Perform a "Select ALL" and "Copy" operation, copying the complete message to the clipboard or equivalent.





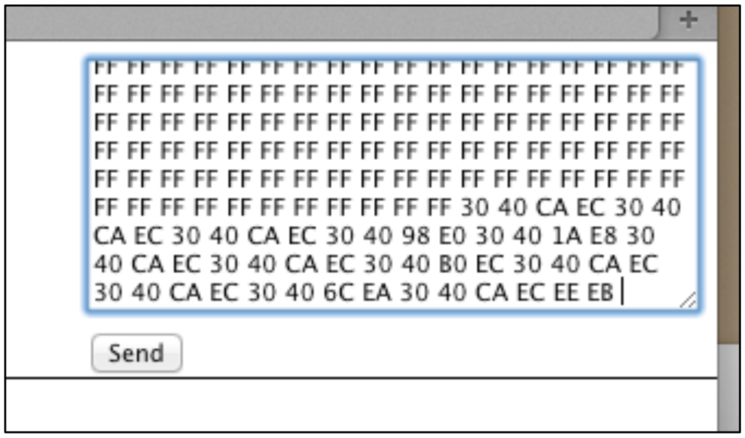
4. Go to the CQT web site for the Afridev unit

5. In the Message window, perform a paste and send of the Firmware Upgrade Message



```
0f 02 04 00 00 00 00 cc
onth":1,"dayCount":0,"productId":0,"gmtHour":0,"gmtYear":15,"reserved":"AADM","gmtDay":1,"gmtSecond":38,"fwVersionMajor":2

0f 02 04 00 00 00 00 cc
onth":1,"dayCount":0,"productId":0,"gmtHour":0,"gmtYear":15,"reserved":"AADM","gmtDay":1,"gmtSecond":59,"fwVersionMajor":2
```



```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF 30 40 CA EC 30 40
CA EC 30 40 CA EC 30 40 98 E0 30 40 1A E8 30
40 CA EC 30 40 CA EC 30 40 B0 EC 30 40 CA EC
30 40 CA EC 30 40 6C EA 30 40 CA EC EE EB
```

Then next time the Afridev unit performs a Final Assembly, Water Data or Monthly Check-In message, a firmware upgrade will be performed.

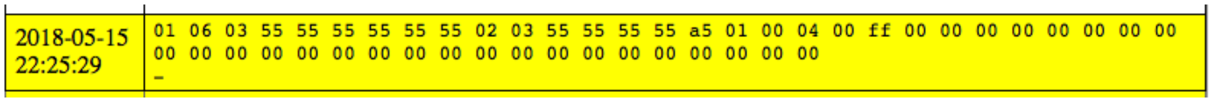
9 Bootloader SOS Recovery Mode

The Bootloader is designed with a recovery scheme. The Bootloader contains a series of checks to provide a “best effort” to detect a bad upgrade file or failed upgrade attempt. The two scenarios where a failure can occur are:

1. The Bootloader upgrade sequence fails
2. The Application contains a bug, causing the Application to reboot after the upgrade

Should the Bootloader detect that either of these cases has occurred, it will fall into the SOS Recovery Mode. In this mode, the Bootloader sends an “SOS” message to the Cloud every 12 hours. This provides the opportunity for the Cloud to identify that a failure has occurred, and allows the Outpour unit to download and program a new Application Firmware image.

The Bootloader SOS message is as follows (as shown on the CQT site):



2018-05-15 22:25:29	01 06 03 55 55 55 55 55 55 02 03 55 55 55 55 a5 01 00 04 00 ff 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 _