AWR1843 Secondary Bootloader Design Guide

This document is a guide to the Secondary Bootloader application for AWR1843 device.

Contents

Contents **1**

List of Figures **1**

1 Introduction **2**

1.1 Design Scheme 2

2 Application Design Flow **5**

3 Decision Analysis and Resolution (DAR) **6**

3.1 TCMA vs TCMB 6

4 Detailed Flow **7**

4.1 Pre Initialization 7
4.2 MPU Configuration 7
4.3 PinMux 7
4.4 User Input 7
4.5 Can Downloader 8
4.6 Flash Downloader 9
4.7 Soft Reset 9

5 Configurable options **10**

6 Usage **11**

6.1 Pre requisites 11
6.2 Running the application 11

7 PC based CAN application **15**

List of Figures

Figure 1: Can Downloader 3
Figure 2: Flash Downloader 4
Figure 3: SBL Design Flow 5
Figure 4: Can downloader Setup 11
Figure 5: SBL Input Prompt 12
Figure 6: Can Downloader Start 13
Figure 7: Can Downloader Finish 14
Figure 8: Flash Downloader 14

1 Introduction

In automotive applications, CAN is the only available external interface to the XWR1843 device and hence any update to the application binaries residing in the SFLASH need to be received over CAN. The secondary bootloader primarily is responsible for supporting the following functionality:

1. Provides the capability of updating application binaries in the SFLASH by receiving the binaries over CAN
2. Loads the application to run on the device. The SBL is never updated once deployed in the field. This ensures that even in case of power outages during the update of the application there is no impact to the availability of the device. On the subsequent power cycle, the SBL again is loaded by the primary (ROM) bootloader to allow for an update to the application.

Secondary bootloader application aims at providing a reference for an application which can fulfill this requirement.

The following Nomenclature is used throughout this document.

Primary Bootloader or PBL	The ROM bootloader
Secondary Bootloader or SBL	This application
Application Meta-Image or User Application	The user’s application Meta-Image that will be loaded over CAN interface

1.1 Design Scheme

The application has following two parts:

- Can Downloader: Update a new or existing application Meta-Image in SFLASH

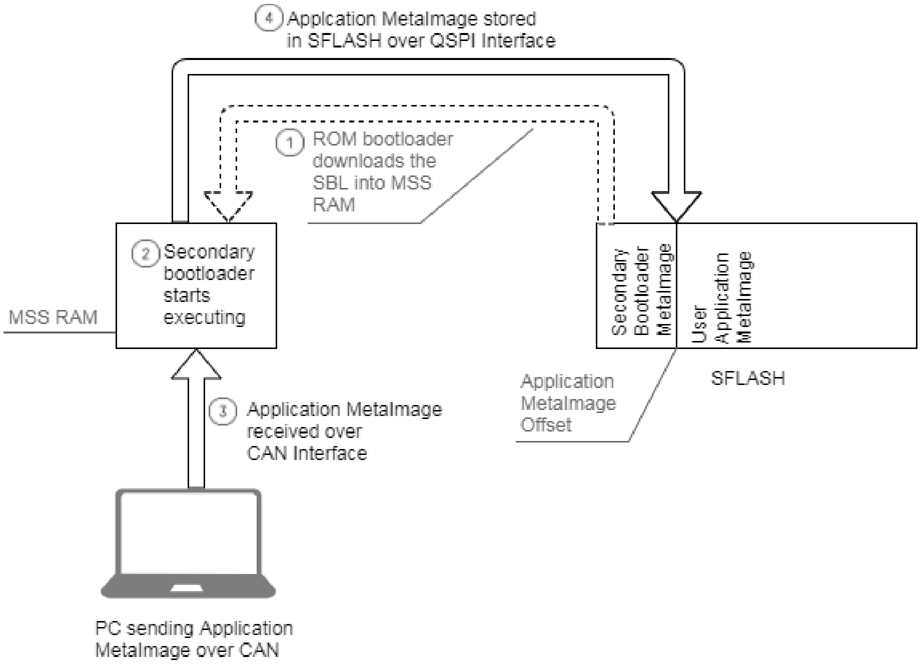

4 Applcation MetaImage stored in SFLASH over QSPI Interface
1 ROM bootloader downloads the SBL into MSS RAM
2 Secondary bootloader starts executing
MSS RAM
Secondary Bootloader MetaImage
User Application MetaImage
SFLASH
Application MetaImage Offset
3 Application MetaImage received over CAN Interface
PC sending Application MetaImage over CAN

Figure 1: Can Downloader

- Flash Downloader: Download the existing application Meta-Image from FLASH into the RAMs of all subsystems

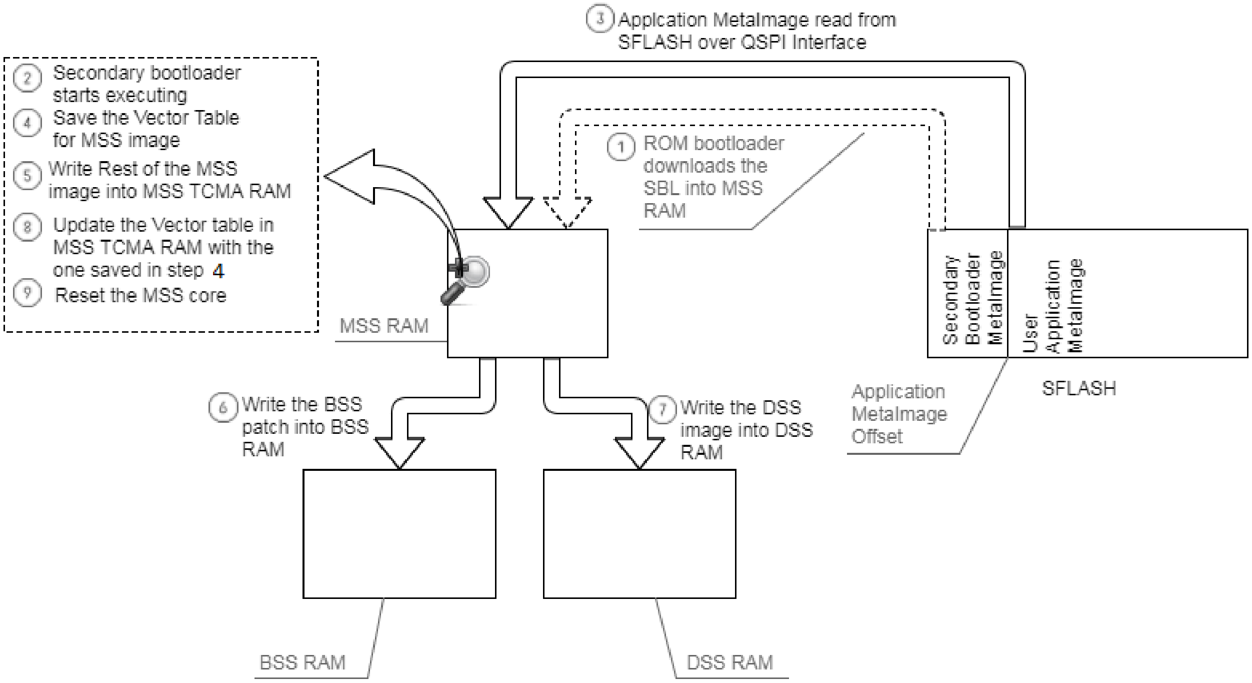

3 Applcation MetaImage read from SFLASH over QSPI Interface
2 Secondary bootloader starts executing
4 Save the Vector Table for MSS image
5 Write Rest of the MSS image into MSS TCMA RAM
8 Update the Vector table in MSS TCMA RAM with the one saved in step 4
9 Reset the MSS core
1 ROM bootloader downloads the SBL into MSS RAM
MSS RAM
Secondary Bootloader MetaImage
User Application MetaImage
SFLASH
Application MetaImage Offset
6 Write the BSS patch into BSS RAM
7 Write the DSS image into DSS RAM
BSS RAM
DSS RAM

Figure 2: Flash Downloader

2 Application Design Flow

The idea behind this approach is that the Meta-Image (without the BSS patch and DSS image) for the secondary bootloader application will be flashed once over UART. From there on, following each Cold/Warm reset, this application will be loaded from the SFLASH into the MSS RAM by the primary bootloader, which then will allow the user to do either of the following based on the user input over UART:

- Flash the Application Meta-Image at a fixed offset (from the Secondary bootloader image) in the SFLASH.
- Load the Application Meta-Image present at a fixed offset (from the Secondary bootloader image) to the following memories:
  - MSS – TCMA, TCMB
  - BSS – TCMA, TCMB
  - DSS – L2, L3 memories

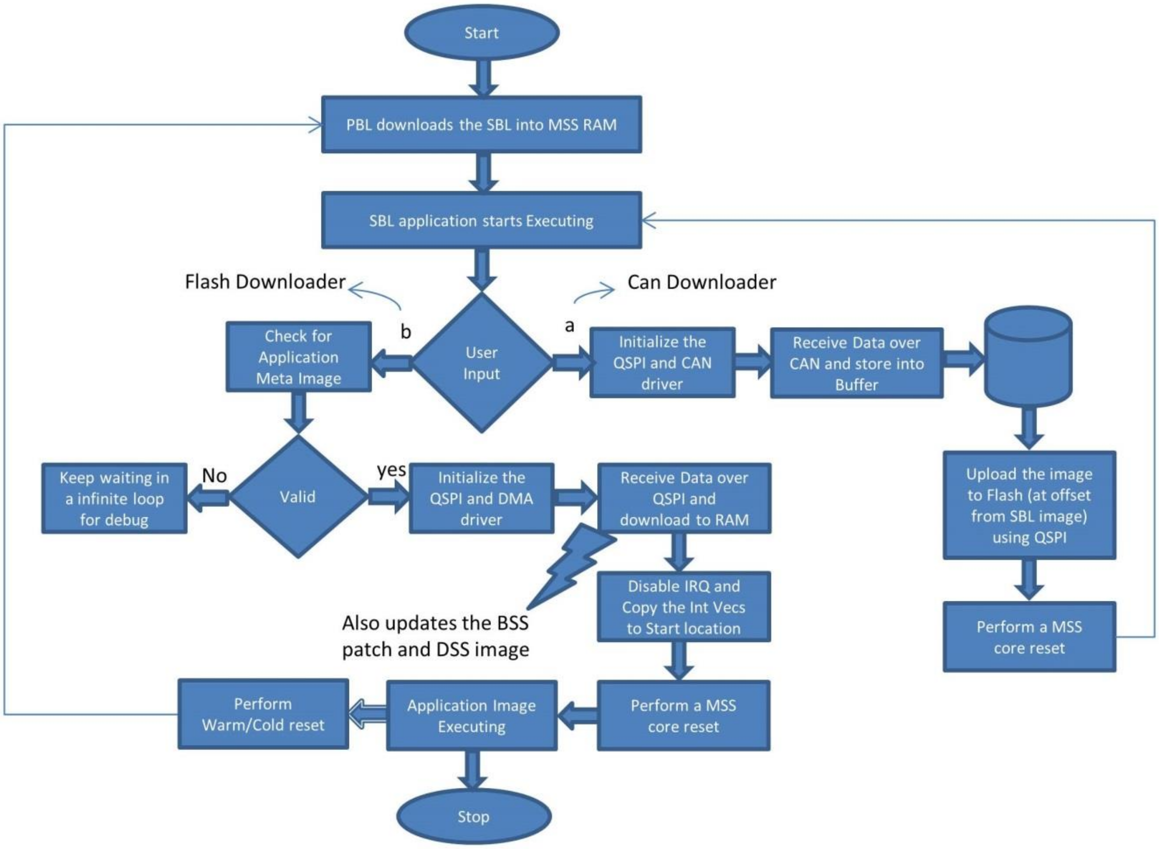

Start
PBL downloads the SBL into MSS RAM
SBL application starts Executing
Flash Downloader
Can Downloader
b
a
User Input
Check for Application Meta Image
Initialize the QSPI and CAN driver
Receive Data over CAN and store into Buffer
Upload the image to Flash (at offset from SBL image) using QSPI
Perform a MSS core reset
Keep waiting in a infinite loop for debug
No
Valid
yes
Initialize the QSPI and DMA driver
Receive Data over QSPI and download to RAM
Disable IRQ and Copy the Int Vecs to Start location
Also updates the BSS patch and DSS image
Perform Warm/Cold reset
Application Image Executing
Perform a MSS core reset
Stop

Figure 3: SBL Design Flow

3 Decision Analysis and Resolution (DAR)

3.1 TCMA vs TCMB

Whether to execute the Secondary Bootloader code from TCMA or TCMB.

- **TCMA advantages over TCMB**

As the primary bootloader already load the code into the TCMA, no additional effort is required in the SBL application code. Thus, the execution time will be relatively less and the SBL code would be simpler.

- **TCMB advantages over TCMA**

If the SBL code is executing from the TCMA, the user application code has to be loaded at an offset larger or equal to the SBL code size. The available TCMA memory available for the user application will be reduced. Whereas, executing the code from TCMB allows the user application code to use the whole of TCMA memory. Also, the TCMB memory is initialized by the application startup code ensuring there would be no reduction in the TCMB memory for user application.

- **Decision**

Apart from the startup code responsible for copying the code into the TCMB memory, the SBL code will be kept in TCMB memory.

4 Detailed Flow

The Secondary Bootloader application code is divided into various phases. The functioning and relevance of each phases is explained in the following subsections.

4.1 Pre Initialization

As most of the code would be executing from the MSS TCMB memory, it needs to be copied from TCMA to TCMB as part of the pre initialization routine. The following variables are defined in the linker command file which will be used to copy the code.

```
extern uint32_t _libRunAddr, _libLoadStart, _libLoadEnd;

extern uint32_t _constRunAddr, _constLoadStart, _constLoadEnd;

extern uint32_t _startupRunAddr, _startupLoadStart, _startupLoadEnd;
```

4.2 MPU Configuration

The following Memory sections will be relevant for this application. The MPU setting of each of these sections is also specified.

- MSS TCMA : Read, Write and Executable
- MSS TCMB : Read, Write and Executable
- BSS TCMA : Read, Write and Executable
- BSS TCMB : Read and Write
- DSS L1 :
- DSS L2 :

4.3 PinMux

The following peripherals are used for this application. Hence the pin muxing for these peripherals is required.

- CAN : CAN-Tx and CAN-Rx
- QSPI : CLK, CS, D0(IN/OUT), D1(IN), D2(IN), D3(IN)
- UART : UART-Tx, UART-Rx

4.4 User Input

The UART is initialized with following configuration:

Baudrate : 115200

Data : 8 bit

Parity : None

Stop bit : 1 bit

Flow control : None

User can choose from one of the following option:

a. Update Meta Image via CAN(FD)

b. Download existing Meta Image from Flash

4.5 Can Downloader

The following steps are performed in this phase:

- Initialize QSPI Flash interface.
- Initialize CAN interface with following configuration:
  - 1 Mbps as nominal bit-rate

Propagation segment: 8

Phase segment1: 6

Phase segment2 : 5

Sync jump: 1

BRP : 2

Nominal Bit rate = (40)/(((8+6+5)+1)*BRP) = 1Mhz

- 5 Mbps as data bit-rate

Propagation segment: 2

Phase segment1: 2

Phase Segment2 : 3

Sync jump: 1

BRP : 1

Data Bit rate = (40)/(((2+2+3)+1)*BRP) = 5Mhz

    - Create an Rx object with DATA_MSG_ID
    - Create an Rx object with TERMINATE_MSG_ID
- Erase the SFLASH for ERASE_SIZE_KB at an offset of METAIMAGE_OFFSET
- Wait to receive a CAN message in a loop
    - If the received message ID is DATA_MSG_ID, write it to the SFLASH
    - If the received message ID is TERMINATE_MSG_ID, break out of the loop
- Close the QSPI Flash interface

4.6 Flash Downloader

The following steps are performed in this phase:

- Initialize the DMA
- Initialize QSPI Flash interface
- Parse the Application Meta-Image present in the SFLASH
- Load each Sub-System image in their respective RAMs
    - If no Error: Close the QSPI Flash interface
    - If Error: wait in an infinite loop for debug

4.7 Soft Reset

Reset the MSS core

5 Configurable options

Only the following options should be configured as per the requirement (in main.c):

```
#define SBL_CANFD_DATA_MSG_ID                      0x29E
```

The CAN message ID for the receiving the Meta-Image.

```
#define SBL_CANFD_TERMINATE_MSG_ID                 0x19E
```

The CAN message ID for the notifying the end of CAN transmission.

```
#define METAIMAGE_OFFSET            (300 * 1024)
```

The user application Meta-Image offset from the start of flash memory. This value shall be larger than the size of SBL Meta-Image.

```
#define ERASE_SIZE_KB               (1024)
```

This is the size of flash memory that needs to be cleared before flashing the application Meta-Image. This value shall be larger than the size of the application Meta-Image.

6 Usage

6.1 Pre requisites

- **ECO’s:** As this application receives the image over CAN interface, the ECOs for enabling the CAN-FD interface on the EVM are required. Below are the ECOs
  - Mount 0 ohms on R11 and R12
  - Remove R6 and R4
- **Setup:** The setup required for this application is shown below.

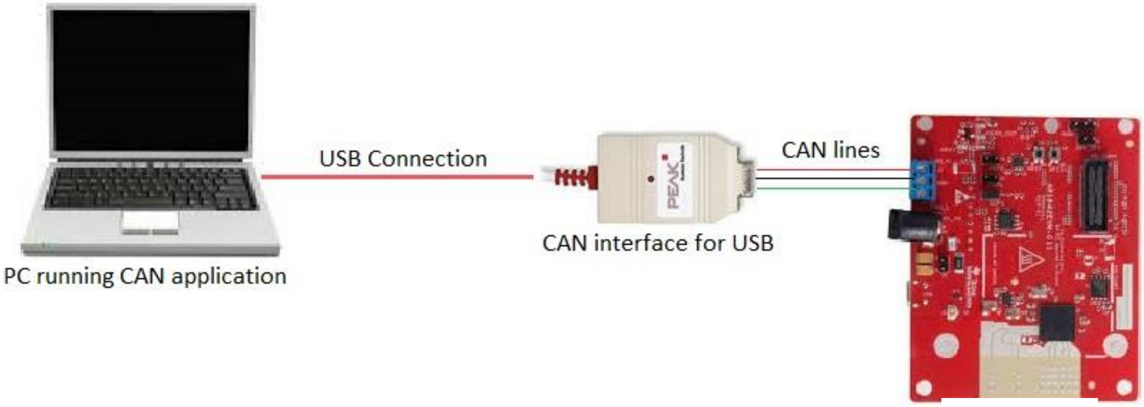


Figure 4: Can downloader Setup

- **PC based CAN application:** For the CAN downloader part of the application, where the Application Meta-Image will be updated at an offset in the SFlash, a PC based application is required. This PC based application will communicate with the device over USB-to-CAN dongle. The purpose of this application is to send the Meta-Image over CAN in chunks of 64 bytes each. For more information on this application refer to PC based CAN application.

6.2 Running the application

The Meta-Image for SBL application will be created automatically along with the out file upon compiling this application. Note that this image will contain only the MSS binary (no BSS or DSS binary). This Meta-Image needs to be flashed once using the conventional method (uniflash). From this point onwards, this image will reside in the SFLASH as the primary Meta-Image (until a new image is flashed using the uniflash) and will be loaded each time after a cold or warm reset. Follow the below steps to run the application

- Find the COM port which shows as “*XDS110 Class Application/User UART”* under device manager.

- Connect to the COM port found in previous step (using any serial terminal application like Teraterm, Hyperterminal etc.) and configure this port with following configuration.

Baudrate : 115200

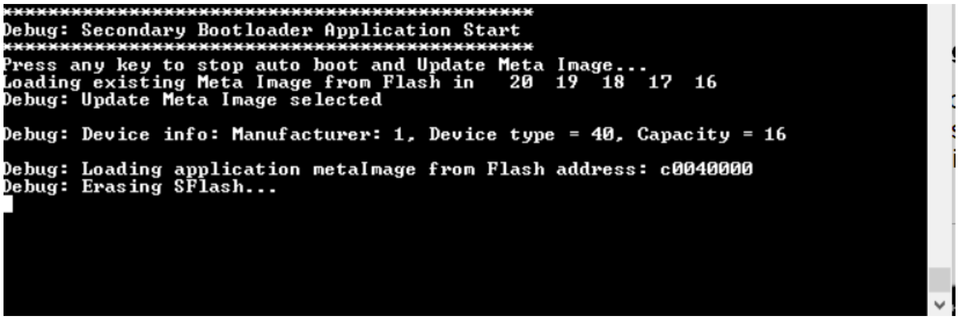
Data : 8 bit

Parity : None

Stop bit : 1 bit

Flow control : None

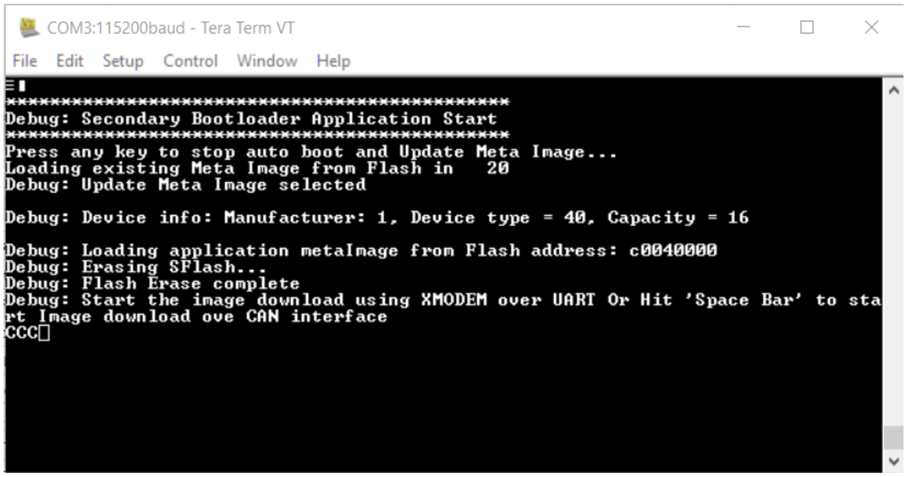
- Restart the device by pressing the *"SW2"* on the EVM. The following prompt will appear on the serial terminal console. Refer to the below figure.
- Hit "Enter" on the console. The application will start erasing the portion of SLFASH, where the user application Meta-Image will reside. After the Erase has completed the application will ask the user to start the PC based CAN application. Refer to below figure.



```
**********************************************
Debug: Secondary Bootloader Application Start
**********************************************
Press any key to stop auto boot and Update Meta Image...
Loading existing Meta Image from Flash in   20  19  18  17  16
Debug: Update Meta Image selected

Debug: Device info: Manufacturer: 1, Device type = 40, Capacity = 16

Debug: Loading application metaImage from Flash address: c0040000
Debug: Erasing SFlash...
```

Figure 5: SBL Input Prompt

- Hit “space bar” on the console. The SBL will wait for the CAN messages to update the flash.

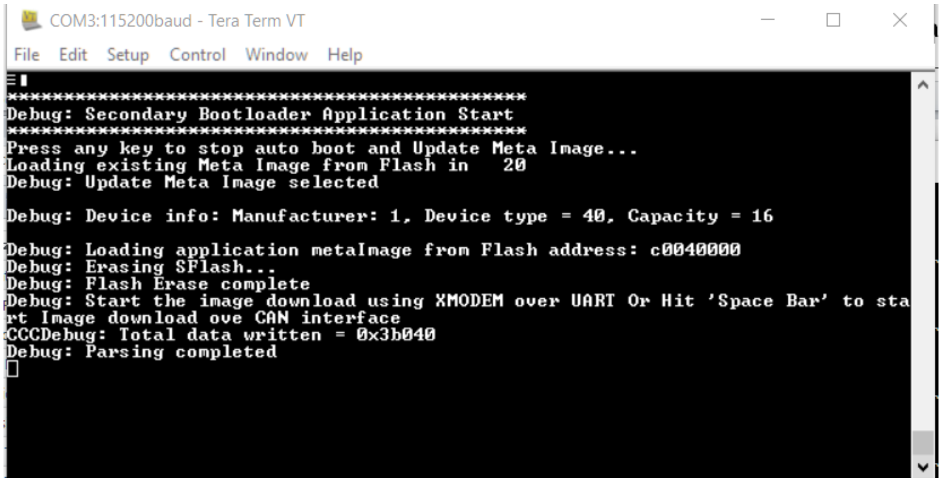

COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
**********************************************
Debug: Secondary Bootloader Application Start
**********************************************
Press any key to stop auto boot and Update Meta Image...
Loading existing Meta Image from Flash in   20
Debug: Update Meta Image selected

Debug: Device info: Manufacturer: 1, Device type = 40, Capacity = 16

Debug: Loading application metaImage from Flash address: c0040000
Debug: Erasing SFlash...
Debug: Flash Erase complete
Debug: Start the image download using XMODEM over UART Or Hit 'Space Bar' to sta
rt Image download ove CAN interface
CCC
```

Figure 6: Can Downloader Start

- This application uses PCAN (Peak) USB-CAN adaptor to download firmware to device. To use this application with Peak PCAN adaptor follow these steps:
  - Copy PCANBasic.dll to 'C:\windows\SysWOW64' directory(Download from PCAN-Basic API)
  - Open the command prompt in the directory where the "CAN_Metaimage_Flasher.exe" is located
  - Run the PC based CAN CLI based application. Open the CLI in Windows PC and run below command
    CAN_Metaimage_Flasher.exe <MetaImage_path_name>
    <MetaIMage_path_name> => path and name of the MetaImage

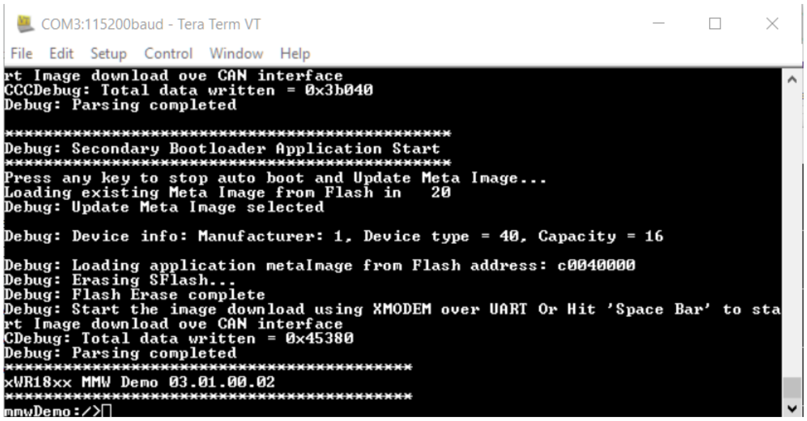
- Once the PC application has sent the complete Meta-Image over the CAN, the console will show the total number of bytes transferred. This can be used to ensure that the complete image has been transferred. The application will restart and the input prompt will show up again. Refer to the below figure.


COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
**********************************************
Debug: Secondary Bootloader Application Start
**********************************************
Press any key to stop auto boot and Update Meta Image...
Loading existing Meta Image from Flash in   20
Debug: Update Meta Image selected

Debug: Device info: Manufacturer: 1, Device type = 40, Capacity = 16

Debug: Loading application metaImage from Flash address: c0040000
Debug: Erasing SFlash...
Debug: Flash Erase complete
Debug: Start the image download using XMODEM over UART Or Hit 'Space Bar' to sta
rt Image download ove CAN interface
CCCDebug: Total data written = 0x3b040
Debug: Parsing completed
```

Figure 7: Can Downloader Finish

- If there is no error is the application Meta-Image, the application will start executing automatically. Refer to the below figure.

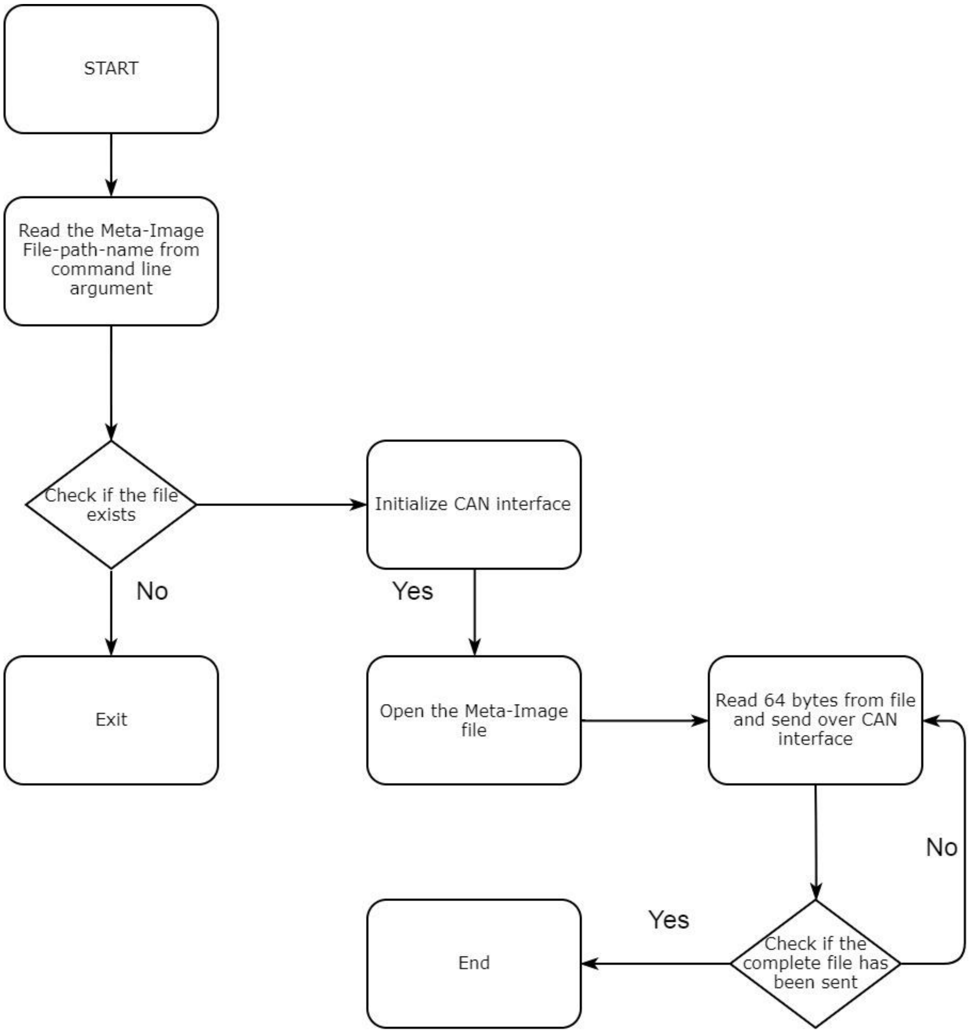

COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
rt Image download ove CAN interface
CCCDebug: Total data written = 0x3b040
Debug: Parsing completed

**********************************************
Debug: Secondary Bootloader Application Start
**********************************************
Press any key to stop auto boot and Update Meta Image...
Loading existing Meta Image from Flash in   20
Debug: Update Meta Image selected

Debug: Device info: Manufacturer: 1, Device type = 40, Capacity = 16

Debug: Loading application metaImage from Flash address: c0040000
Debug: Erasing SFlash...
Debug: Flash Erase complete
Debug: Start the image download using XMODEM over UART Or Hit 'Space Bar' to sta
rt Image download ove CAN interface
CDebug: Total data written = 0x45380
Debug: Parsing completed
******************************************
xWR18xx MMW Demo 03.01.00.02
******************************************
mmwDemo:/>
```

Figure 8: Flash Downloader

7 PC based CAN application

A PC based CAN application based on the Peak System USB to CAN Dongle is already provided. For other similar dongles, the application has to be ported. The General flow of the application is explained in the below flow diagram


START
Read the Meta-Image File-path-name from command line argument
Check if the file exists
No
Exit
Yes
Initialize CAN interface
Open the Meta-Image file
Read 64 bytes from file and send over CAN interface
Check if the complete file has been sent
No
Yes
End