RAppID Boot Loader Utility

Rev August 07, 2013

freescale™

Contents

Section number **Title** **Page**

Chapter 1
Introduction

Chapter 2
Overview

Chapter 3
General Product Description

Chapter 4
RAppID Boot Loader File Description

Chapter 1
Introduction

The RAppID Boot Loader tool developed by Freescale helps develop software for Freescale Microcontrollers. The tool allows you to update the Microcontroller software through a serial link using CCP (CAN Calibration Protocol).

1.1 Purpose

This document outlines the steps to use the RAppID Boot Loader Tool with Freescale Microcontrollers.

1.2 Scope

This document only applies to the use of the RAppID Boot Loader Tool utility.

1.3 Definitions and Acronyms

This section gives an alphabetical list of definitions and acronyms that are relevant to this software product.

Table 1-1. Definitions and Acronyms

Term / Acronym	Definition
CAN	Controller Area Network
MCU	Micro Controller Unit
RBL	RAppID Boot Loader
CCP	CAN Calibration Protocol

Table continues on the next page...

Table 1-1. Definitions and Acronyms (continued)

Term / Acronym	Definition
RBA	RAppID Boot Loader Algorithm
BAM	Boot Assist Module located on MCU silicon
RAM	Volatile Memory
RBF	RAppID Boot Loader Flash Algorithm

1.4 References

The following table lists the references.

Table 1-2. References

MPC56xx	Reference Manuals	Boot Assist Module	Auto
PXSxxxx/ PXRxxxx	Reference Manuals	Boot Assist Module	IMM
CCP	ASAP Standard	Version 2.1	18-Feb-99

Chapter 2
Overview

The RAppID Boot Loader works with the built in Boot Assist Module (BAM) included in the Freescale Qorivva & PX series family of parts. The Boot Loader supports the operational modes of the different part family BAM silicon instantiations. The Boot Loader provides a streamlined method for programming code into FLASH or RAM on either target EVBs or custom boards. Once programming is complete the application code automatically starts.

2.1 Modes of Operation

The Boot Loader has two modes of operation:

- for use as a stand-alone PC desktop GUI utility, or
- for integration with different user required tools chains through a command line interface (i.e. Eclipse Plug-in, MatLab/SimuLink etc.).

2.2 Communication Interface to Microcontroller

The Boot Loader is able to interface to the Microcontroller with the communication interfaces supported by the BAM, which are Serial (UART) port and CAN interfaces. It is also possible to change how quickly the download is over serial by changing the .rbm file. For details, refer Boot loader MCU File (*.rbm) Format

NOTE

See release notes on which targets support CAN communication.

2.3 Graphical User Interface

The Boot Loader provides a GUI allowing the user to select the target communication mode, as well as set the target address, code size, and .s19/.mot file to be programmed. It also allows for these setting to be brought in from a pre-set configuration file (.rbl) or to able to save a configuration file based on the current GUI settings.

2.4 Microcontrollers Supported

Refer to the Release Notes for the list of supported Microcontrollers and how they are supported (RBA or RBF).

2.5 RAppID Boot Loader Algorithm (RBA)

The RBA is the target Microcontroller specific algorithm, which the GUI interfaces to for boot loader operation. The Boot Loader application is structured so that it can automatically pick the correct RBA based on the Microcontroller selection. Also, based on the Microcontroller selection, the Boot Loader will automatically determine the appropriate interface protocol for the BAM, so that it can download the RBA into RAM through the BAM.

2.6 RAppID Boot Loader Summary of Operation RBA

The following figure illustrates the steps of operation for the Boot Loader with RBA:

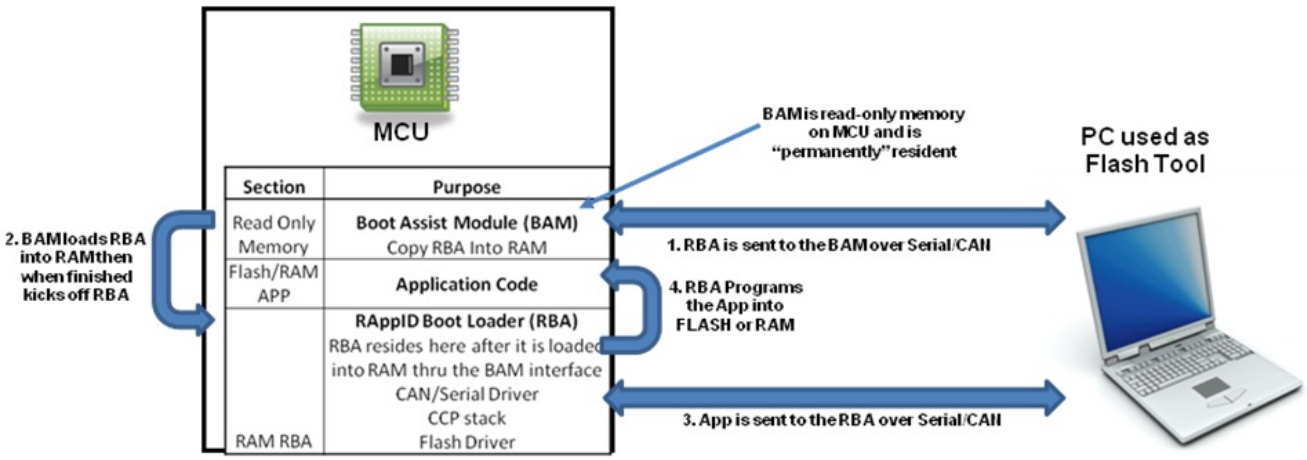

Figure 2-1. RAppID Boot Loader Summary of Operation with RBA

2.7 RAppID Boot Loader Flash Algorithm (RBF)

The RBF is the target Microcontroller specific algorithm, which the GUI interfaces to for boot loader operation. The difference between RBA and RBF is that the RBF is a Flash implementation where you must program the RBF using a third party tool before using the GUI. Ensure that you type in the file dialog box *.* or *.rbf if the file dialog box filter is filtering it out..

The Boot Loader application is structured such that when using an RBF it will skip the BTL download process and start programming the application code. When using an RBF supported Microcontroller there are some parameters that need to be considered to be included into the application:

1. Eight bytes offset from a valid RCHW (see Microcontroller Reference Manual for more information) there is a four byte field that defines the amount of time (in microseconds) the bootloader will delay before jumping to the Application. If the field is 0xFFFFFFFF then the delay will default to 20ms. Otherwise it will delay as long as the user specifies in the field.
2. Twelve bytes offset from a valid RCHW or just after the time delay field the address of where the application key is located should be inserted. The application key needs to be a value of 0x55AA55AA for the bootloader to jump to the application. The finding of a valid application key means that the programming of the application completed properly. Therefore placement of this key should be at the end of program memory. Since the address of the key will be used to find the key it does not need to be in a fixed memory location.
3. Please see the release notes for Microcontroller specific limitation.

2.8 RAppID Boot Loader Summary of Operation RBF

The following figure illustrates the steps of operation for the Boot Loader with RBF:

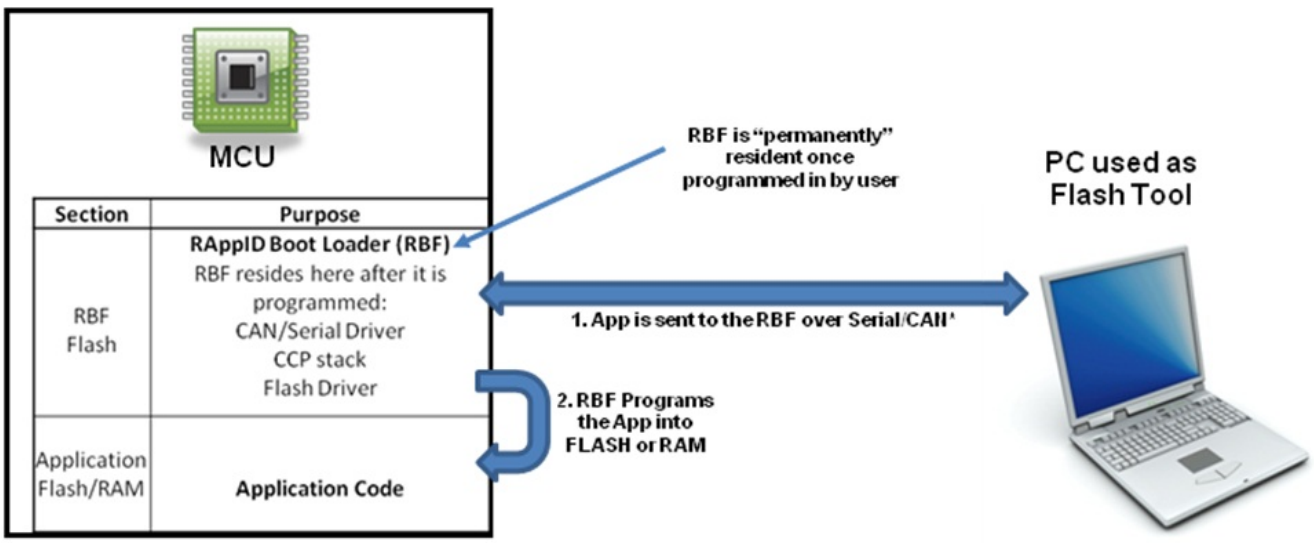

Figure 2-2. RAppID Boot Loader Summary of Operation RBF

2.9 Enabling Vector Hardware

To enable an interface between RAppID Boot Loader utility and Vector Hardware and update the Vector Hardware configuration, perform the following steps.

1. Right-click on **Application** and select **Add** to add the **RAppID_BL** application to the **Application** list.
2. Right-click on the **Vector Device** channel you wish to assign the application to and select **RAppID_BL**.

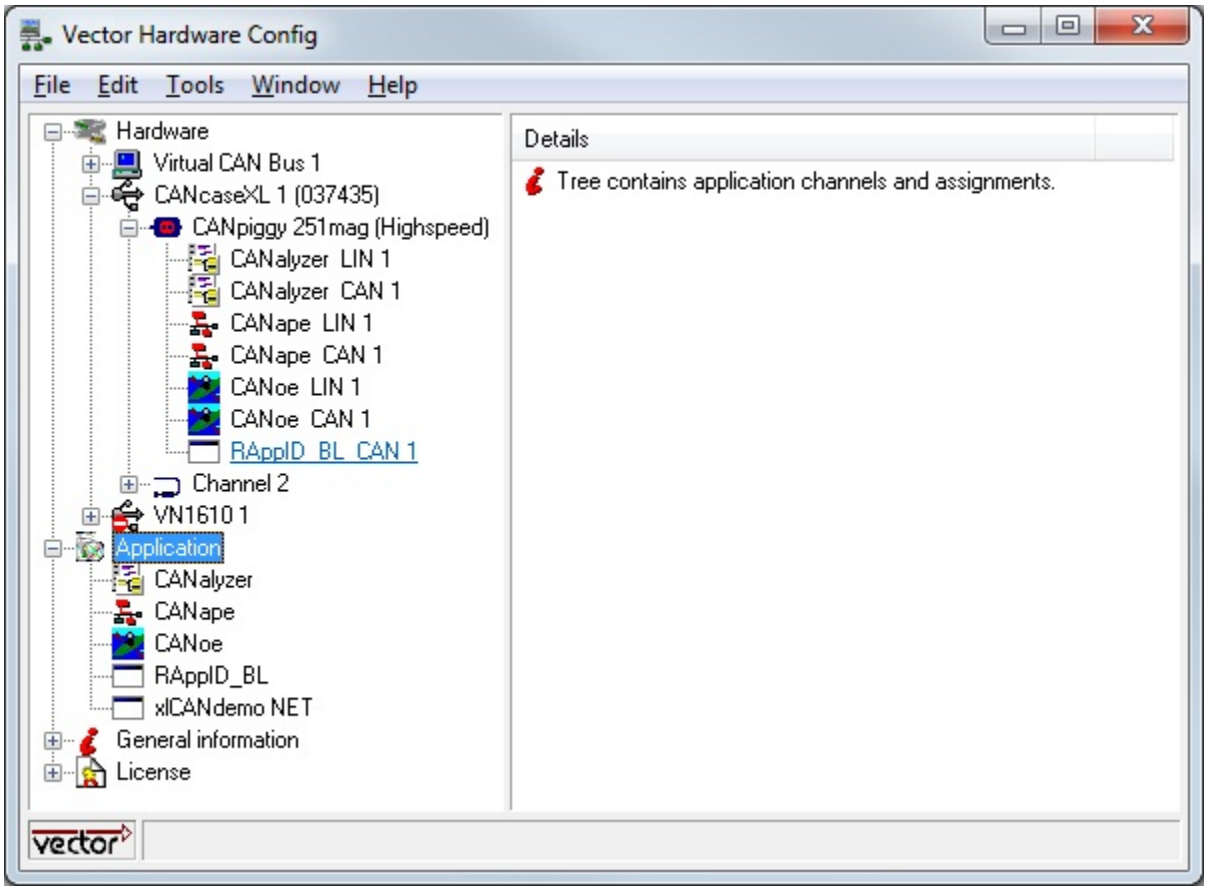

Figure 2-3. Vector Hardware Configuration for RAppID Boot Loader Utility

Chapter 3
General Product Description

The general product description includes:

- Downloading application code with the RAppID Boot Loader GUI
- Configuring the Boot Loader with a Configuration File
- Configuring the Boot Loader Manually (without loading an .rbl file)
- Operation Setup Description
- Programming Using Command Line Interface

3.1 Downloading application code with the RAppID Boot Loader GUI

Before downloading the application to the target ensure for following Hardware settings. This applies only if you are using an RBA supported Microcontroller.

1. Target is set in alternate boot configuration mode with below mentioned pin configuration :

Table 3-1. Configuration

MCU Pin	Pin Status MPC55xx	MCU Pin	Pin Status MPC56xx
N/A	N/A	FAB	1
BootCfg0	0	ABS0	0
BootCfg1	1	ABS2	1 (see 4)

   In this boot configuration mode the Microcontroller will scan FlexCAN and LINFlex (UART) interfaces while using the Auto Baud feature of the BAM.
2. UART (RS232) of the target board is connected with the COM port of the user machine or the CAN port of the Target board is connected to supported CAN hardware for third party vendor (ex. Vector or IXXAT).

3. After ensuring the above mentioned H/W connections, the target should be powered ON.
4. Some Microcontrollers do not support Auto Baud and therefore only use Fixed Baud in which case the ABS2 Pin status should be set to 0. Please see the release notes to determine which Microcontrollers have this and other limitations.

3.2 Configuring the Boot Loader with a Configuration File

To configure the Boot Loader with a configuration file, perform the followings steps.

1. Double-click on RBL tool executable.

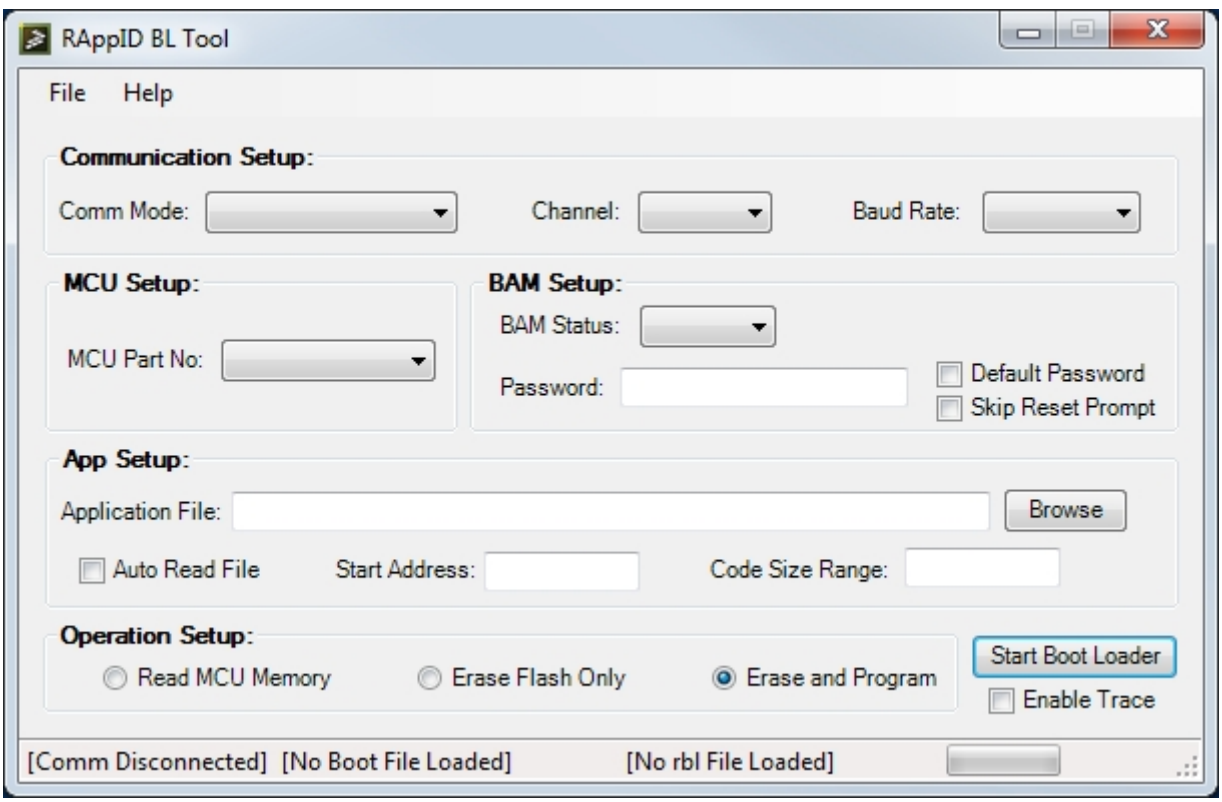

Figure 3-1. Default RAppID Boot Loader Screen

2. Load the RBL configuration file which will contain Microcontroller settings and point to the desired application code file:
   a. Select **File > Load RBL** file.

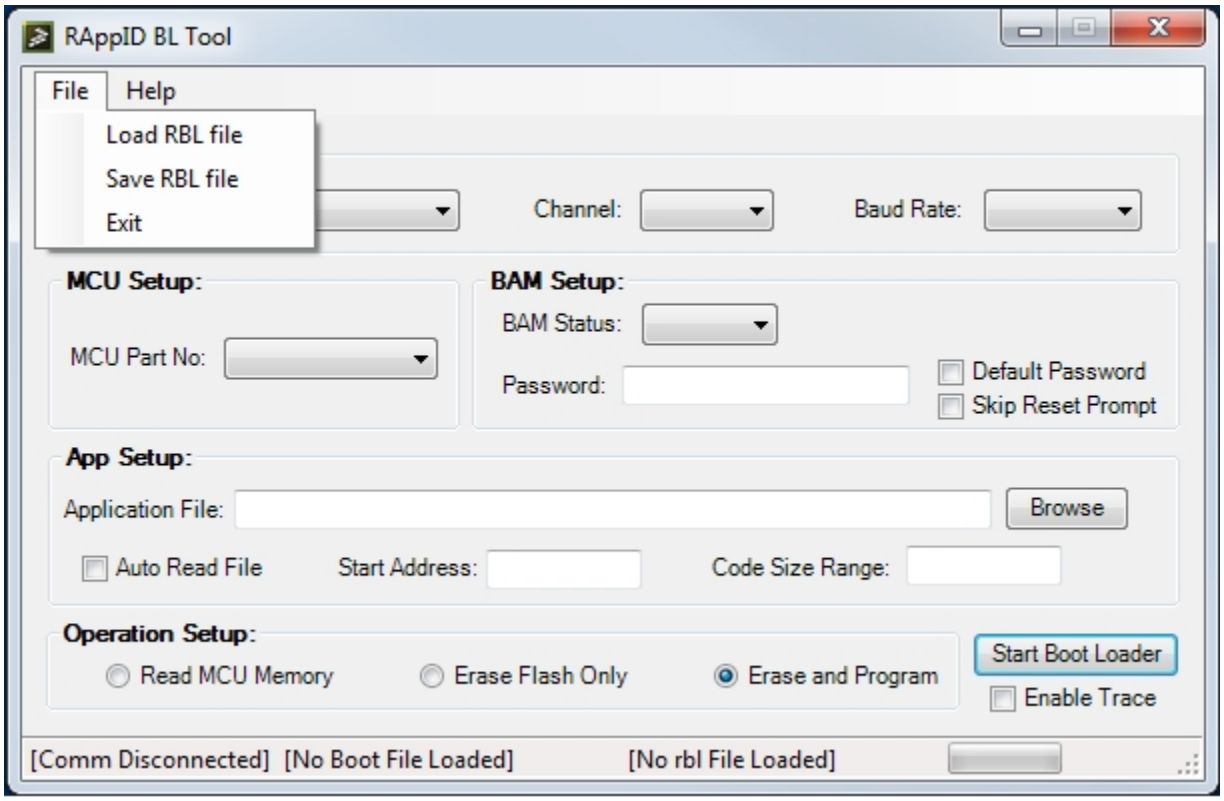

Figure 3-2. Selecting RBL File to Load

b. Navigate to the directory which contains desired RBL configuration file.
c. Select and open the RBL file

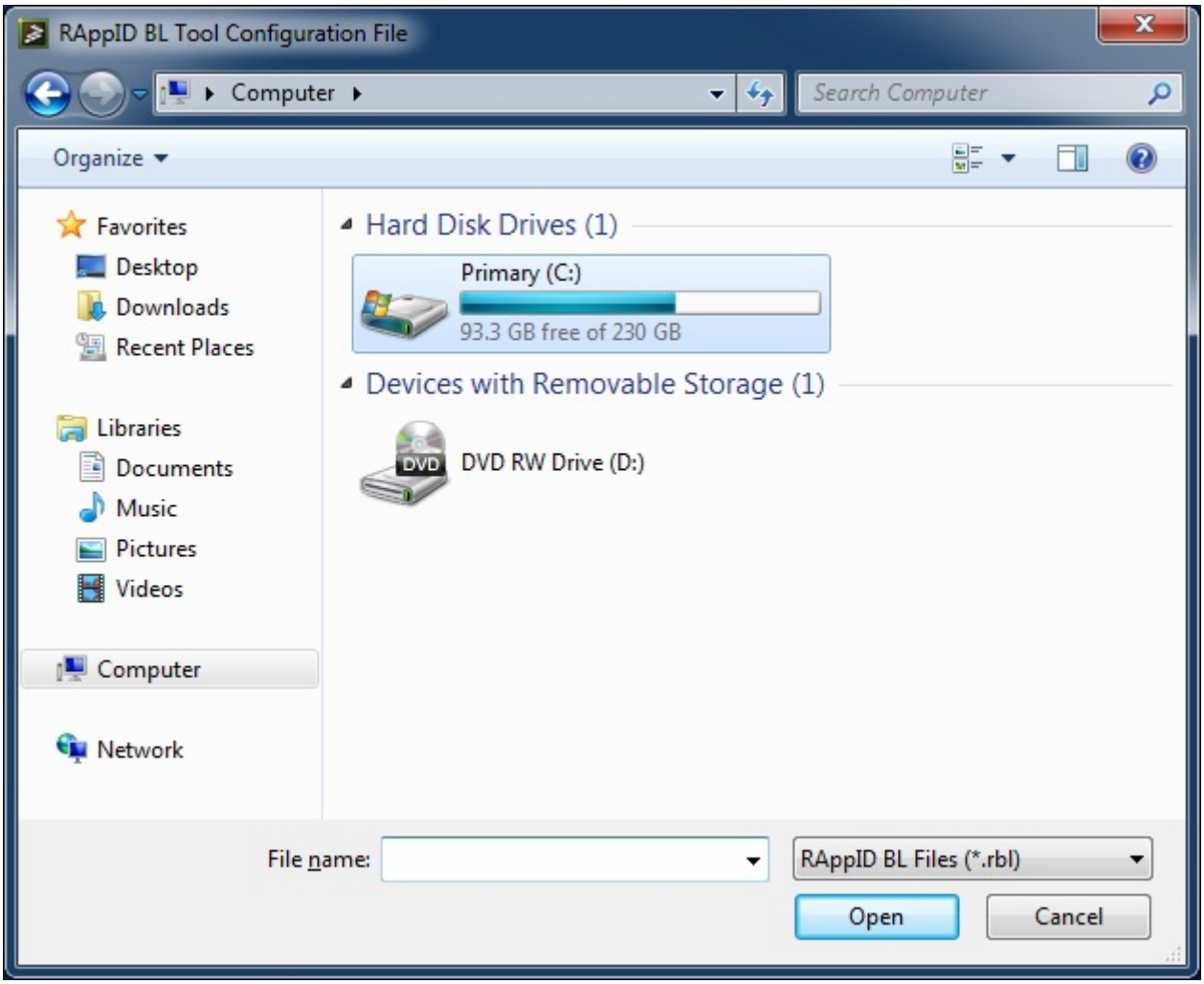

Figure 3-3. Browse to Load an RBL File

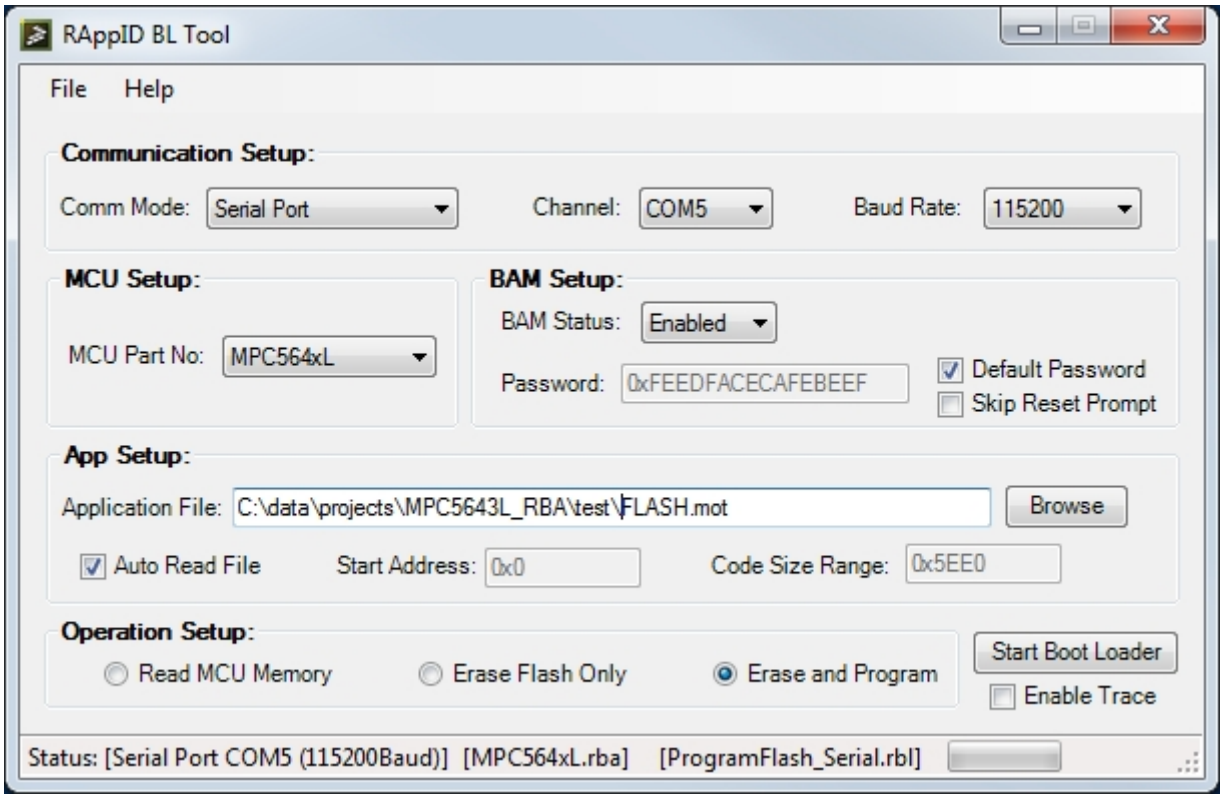

Figure 3-4. Boot Loader Screen after RBL is Loaded

3. Press the **Start Boot Loader** button to download the application to the target and then click **OK** at the prompt.

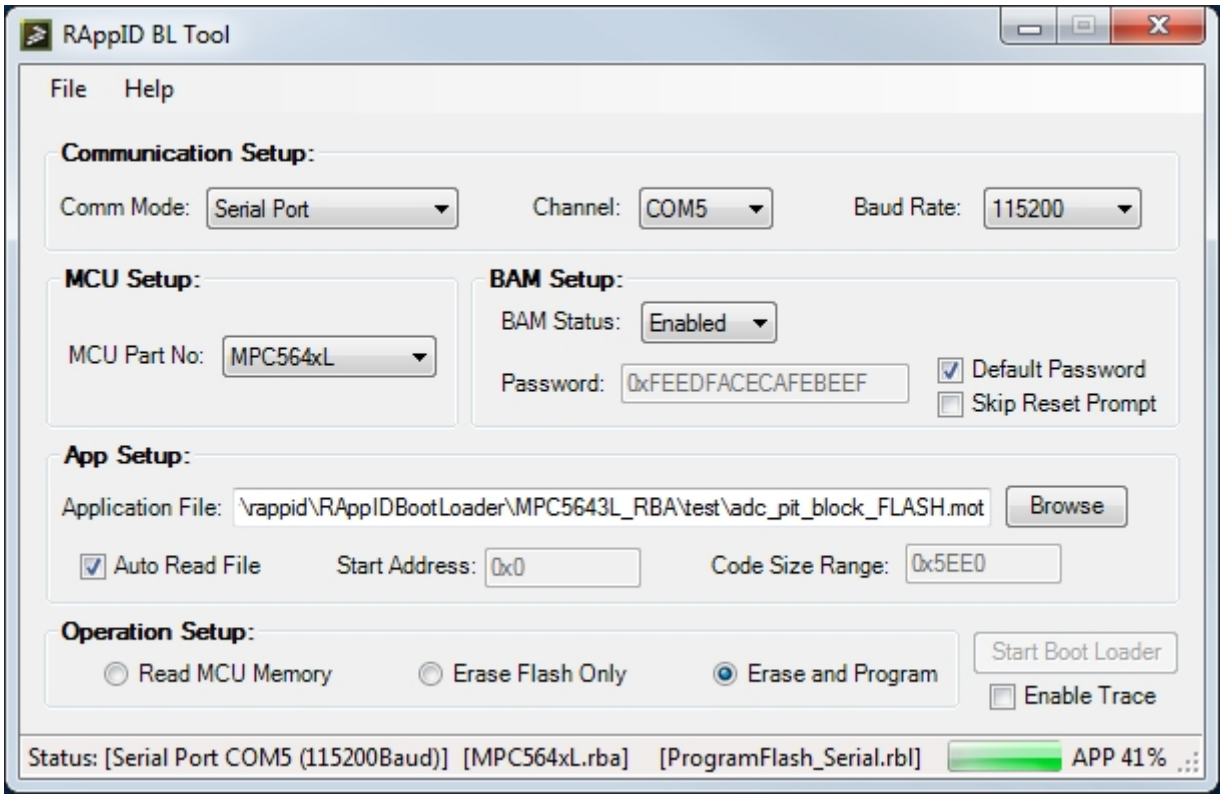

Figure 3-5. Boot Loader in Operation

Once the entire application is downloaded on the target memory boot loader will automatically jump to start location of the application and begin executing it. User can also configure the GUI manually (without loading RBL file) and download.

3.3 Configuring the Boot Loader Manually (without loading an .rbl file)

1. In communication Setup select **Comm Mode** as Serial Port.
2. Ensure that Serial Cable is connected to the port.

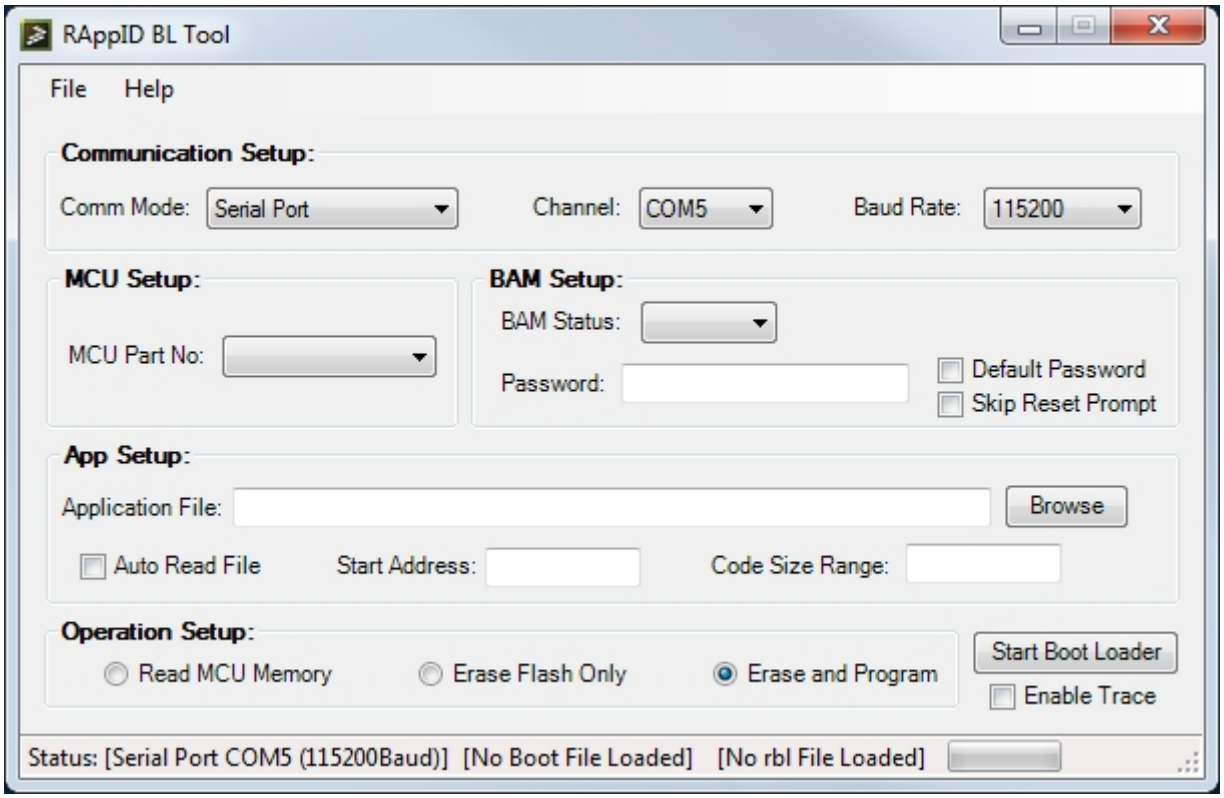

Figure 3-6. Configuring Communication Setup

3. Select desired **MCU Part No**.

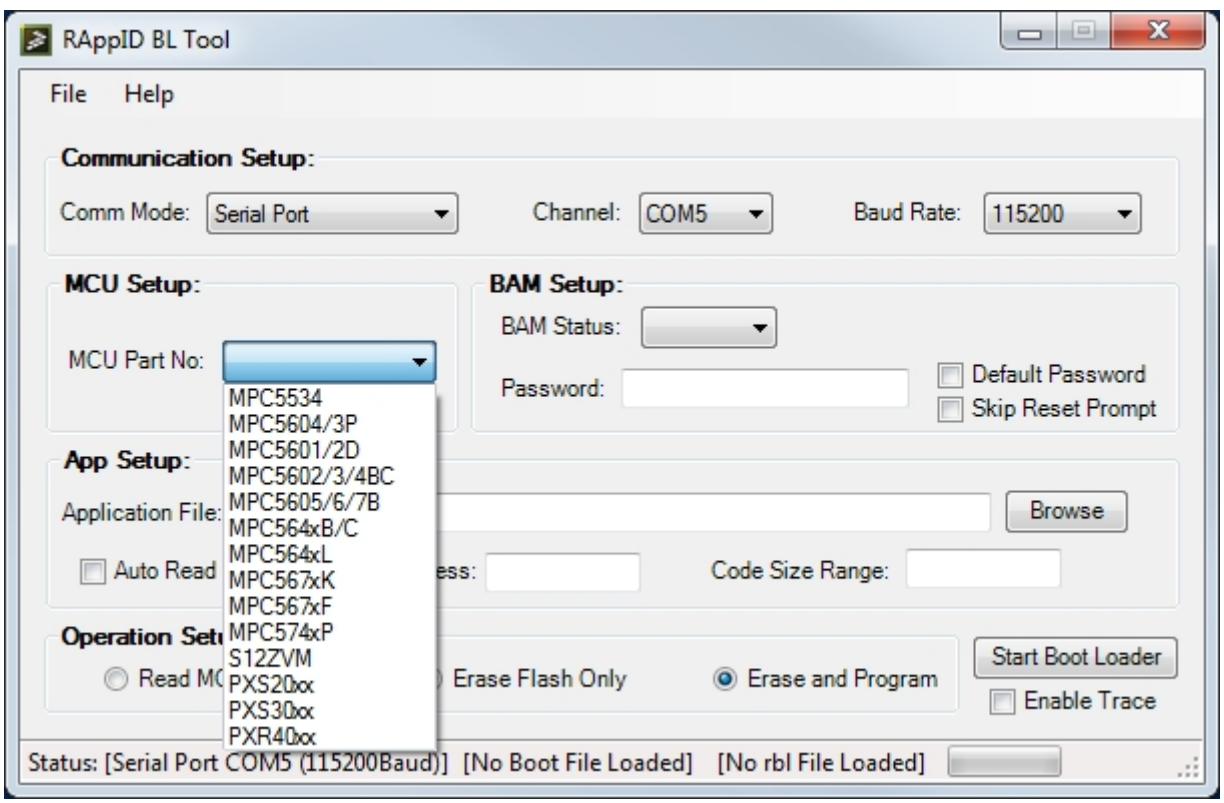

Figure 3-7. Selecting the MCU

4. Select **BAM Status** and **Set the Password**,

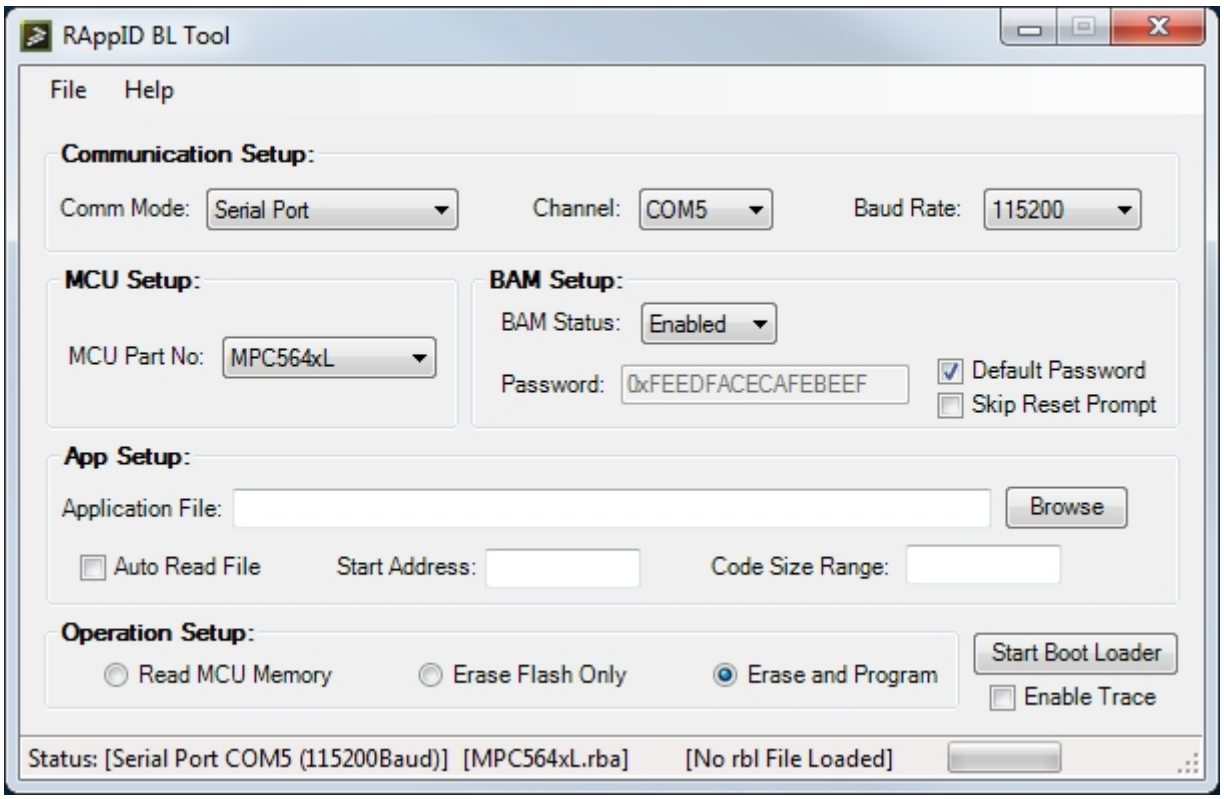

Figure 3-8. Selecting BAM Status & Password

When the BAM status is “Enabled”, the boot loader will download the RBA (boot loader algorithm) as well as the application file to the target memory. If BAM status is set to “Disabled” it will only download the Application to the target memory (boot loader code must be already running on the target). User can either set a specific Password or Check the “Default Password” check box. In both cases the Password should match with the one present in the target memory.

When the Boot Loader starts a prompt is given to remind the user to reset the MCU. This prompt can be skipped if it is not required (for example during batch testing) by checking the Skip Reset Prompt check box.

5. Navigate to the Application file (*.mot file)

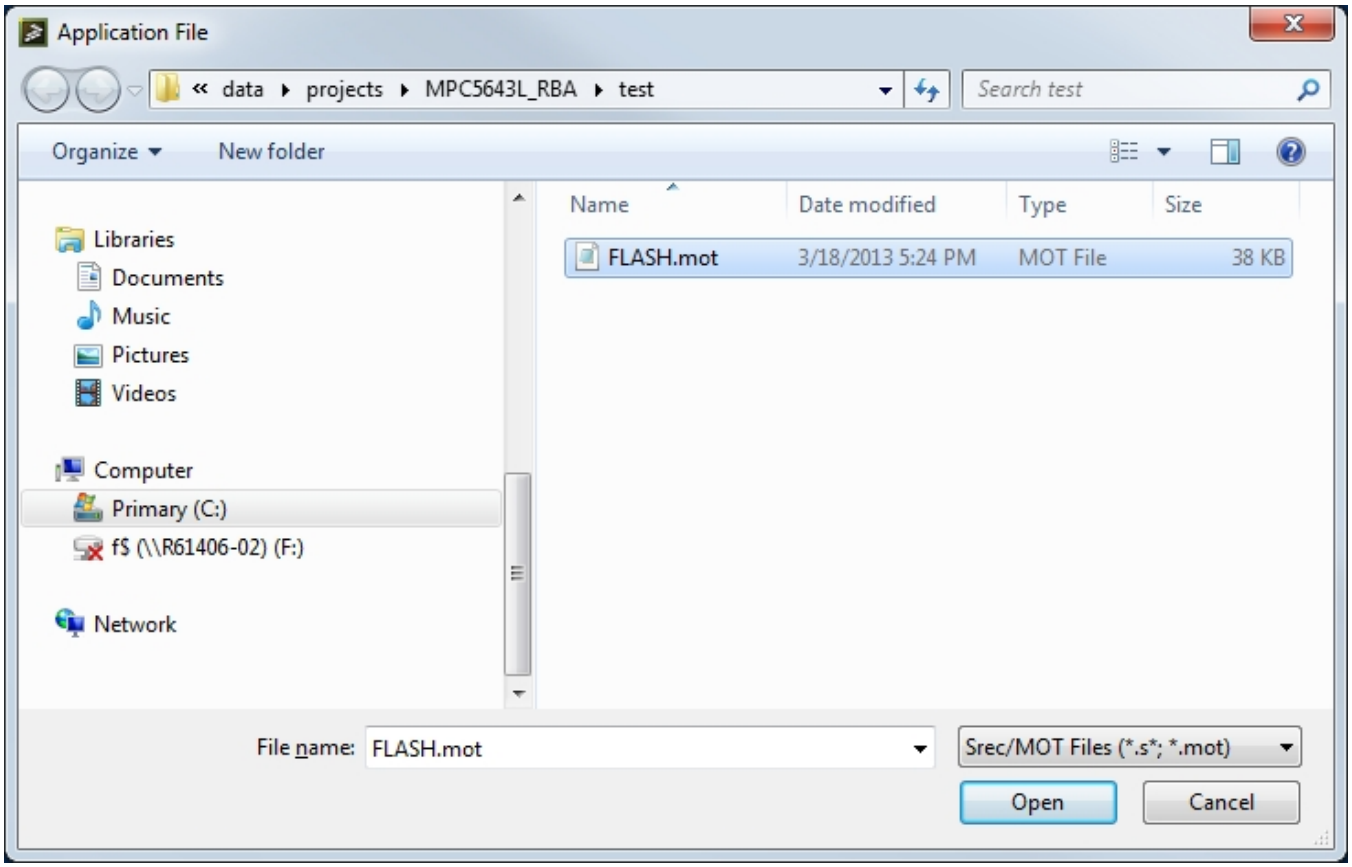

Figure 3-9. Browsing for the Application File

6. Auto Read File & Start Boot Loader.

NOTE

After loading the application file, check the **Auto Read File** check box which will get the Start Address and Code Size from the application (*.mot) file being loaded. Click **Start Boot Loader**.

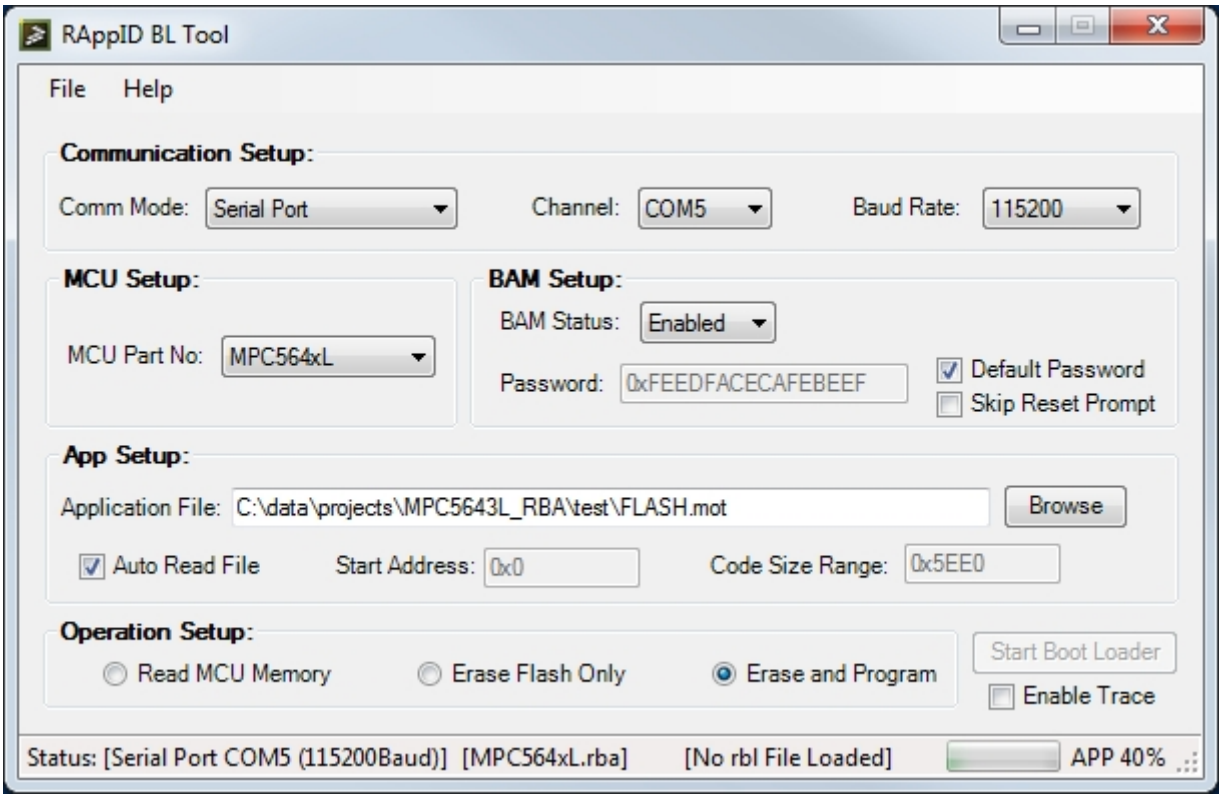

Figure 3-10. Check Auto Read File & Start Boot Loader

3.4 Operation Setup Description

The operation setup has the following options.

1. **Read MCU Memory** – This option enables you to read the Microcontroller memory. If there is no file specified in the App Setup section the RAppID BL Tool stores the memory read to a default file named “output_rbo.s19”. If there is a file name specified, it must have either a .s19, .mot, or .run extension.
2. **Erase Flash Only** – This option erases the entire Flash memory. There is no requirement to fill in the App Setup section for this option.
3. **Erase and Program** – This option programs Flash or RAM depending on the contents of the Application file specified.
4. **Enable Trace** – This option (under the “Start Boot Loader” button) can be selected for any of the three options mentioned above. When this check box is selected a trace window will open showing the communication messages exchanged between the PC and Microcontroller for either Serial or CAN interfaces. To save the contents simply select the text in the trace window and copy to the editor of your choice. The window

will reset on every new run and will only appear the check box is selected. The Enable trace feature is not available when using a command line to invoke the RAppID Boot Loader.

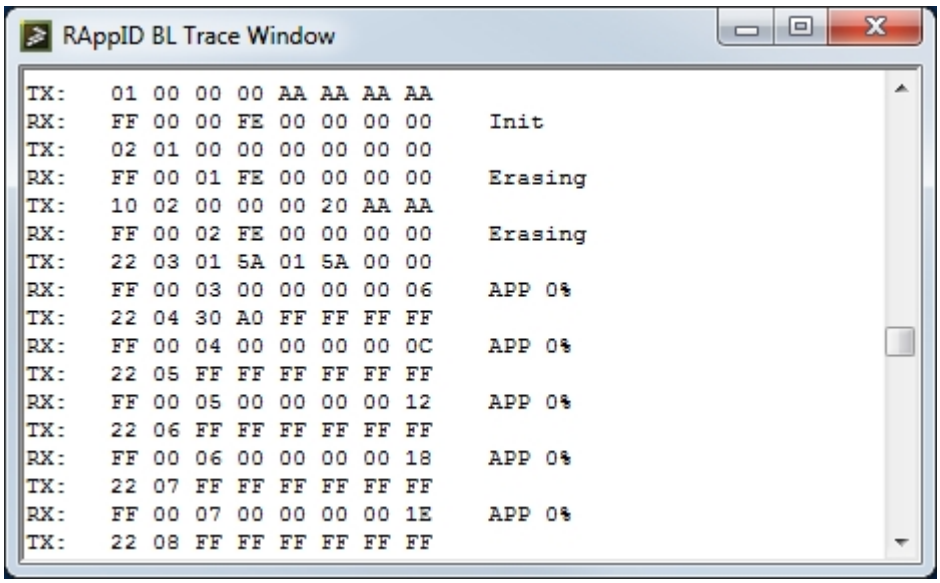

Figure 3-11. Trace Window screen shot

3.5 Programming Using Command Line Interface

Boot loader supports command line interface, which is useful while interfacing it with different tool chains like Matlab/Simulink, Eclipse Plug-In etc.

This section gives step wise description of using Boot loader command line

1. Launch the Windows command line.
2. Change directory to the BL Application directory
3. The boot loader command syntax is as follows

   <>RAppID_BL.exe "<boot loader executable directory path>\\" "<RBL file path with file name>"

   Take care of the command spacing while formatting the command.

Following screen shot shows RAppID bootloader command and boot loader progress bar after command is executed

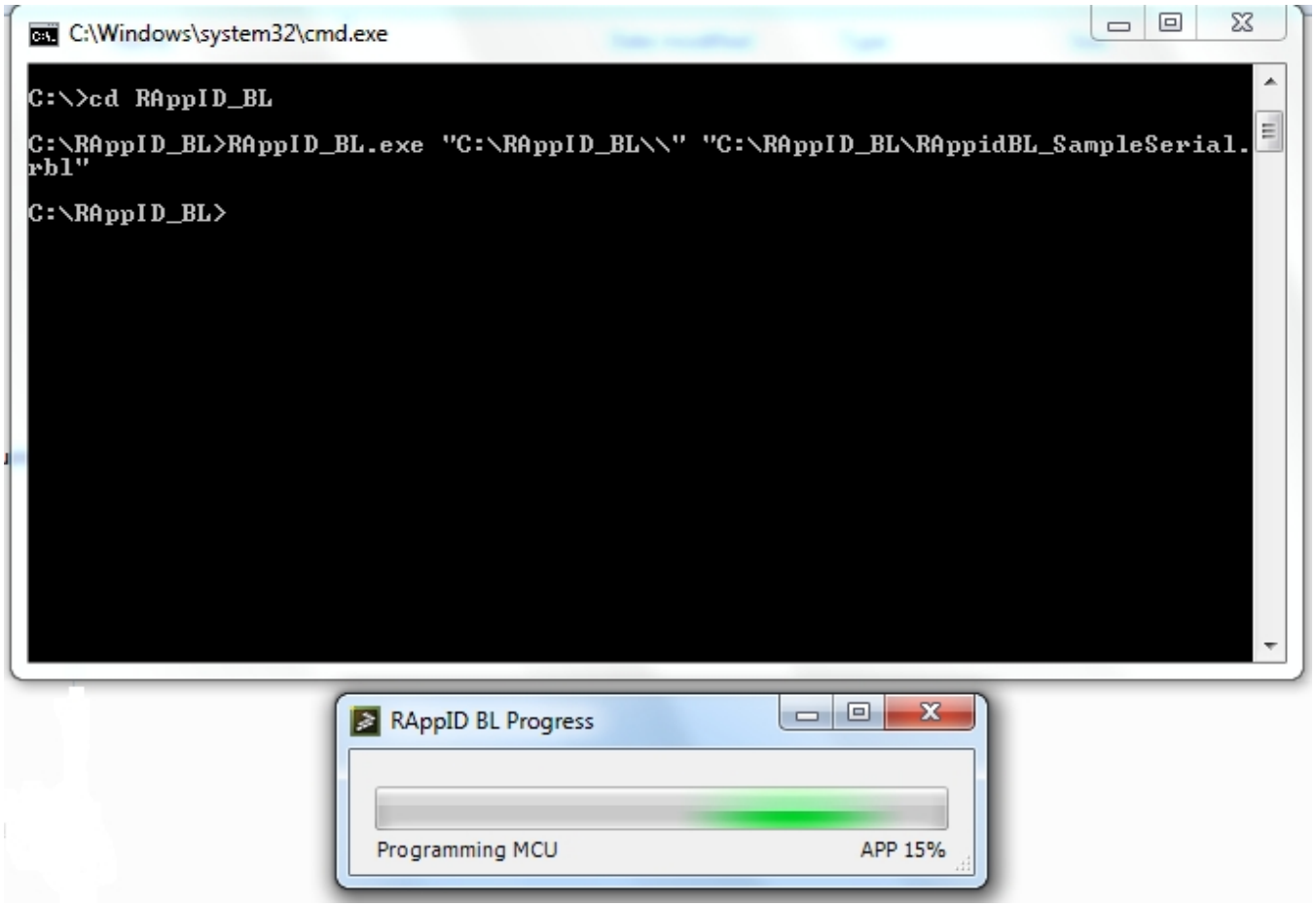

Figure 3-12. Command Line Option usage screen shot

Chapter 4
RAppID Boot Loader File Description

The RAppID Boot Loader has several input files that it works with for configuration and proper communication and programming of the Microcontroller.

4.1 Types of Files Used and Supported by RAppID Boot Loader

The following is the list of files used and supported by RAppID Boot Loader.

- **RAppID Boot Loader Algorithm (.rba) File**

  This file contains the software that performs the Boot Loader operation and is loaded by the BAM into RAM. There will be different rba files to support multiple MCU types. The rba file is in s19 format.

- **RAppID Boot Loader Flash Algorithm (.rbf) File**

  This file contains the software that performs the Boot Loader operation and is loaded by the user into Flash. There will be different rbf files to support multiple MCU types. The rbf file is in s19 format.

- **Application Code**

  This file is provided by the user and is the software that will be programmed by the Boot Loader into the FLASH or RAM. The code file is expected to be in s19 format.

- **RAppID Boot Loader Configuration (.rbl) File**

  Contains the application specific setup information for the Boot Loader so that it can know how to interface to the MCU and which application code will be programmed. This can be loaded in to configure the GUI and enables the use of a command line

interface to start the tool vs. using the GUI. Users can also configure the GUI and save their configuration to an .rbl file so that the same configuration can be used later.

- **RAppID MCU (.rbm) File**

  Contains MCU configuration of the MCUs supported.

4.2 Boot loader Configuration File (*.rbl) Format

Here is a sample .rbl file, which is part of Boot loader utility installation.

Listing: Sample .rbl File

```
*********************************************************************
* This file specifies the format version of the rbl file
* This is needed for compatibility checks between tool versions
* Do not manually change
*********************************************************************
[FORMAT_VERSION]

*********************************************************************
* COMMUNICATION_SETUP - Specifies the specifics of the network HW
*          and settings that are going to be used.
* 1. Hardware Type(Vector CANcaseXL, (Vector CANCardXL, IXXAT USBtoCAN, Serial Port)
* 2. Channel     (1, 2,...,13)
* 3. Baud Rate     (500K, 9600, ...)
*********************************************************************
[COMMUNICATION_SETUP]
Serial Port, COM4, 115200

*********************************************************************
* MCU_SETUP - Specifies the specifics of the ID of the MCU & the
*       location, size and content of the software to be flashed
* 1. MCU Part Number (MPC5604P, MPC5604B, MPC5604S, .....)
*********************************************************************
[MCU_SETUP]
MPC564xL

*********************************************************************
* BAM_SETUP - Specifies the specifics of the ID of the MCU & the
*       location, size and content of the software to be flashed
* 1. BAM Status  (Enabled, Disabled) Is the BAM going to be used
* 2. Password    (0xXXXXXXXX) Password required for BAM to unlock MCU
* 3. Check Box (If checked then use default password for MCU).
* 4. Skip Prompt If checked then do not prompt user to reset/POR MCU
*********************************************************************
[BAM_SETUP]
Enabled, 0xFEEDFACECAFEBEEF, Checked, Unchecked

*********************************************************************
* FILE_INFO - Specifies the file that contains software to be flashed
* 1. File      (The file that contains the hex info)
* 2. Start Address (Start Address of the area to be programmed)
* 3. Size       (Size of code to flash in bytes)
* 4. Check Box(If checked then auto determine start address and size to be programmed.
*********************************************************************
```

```
[FILE_INFO]
D:\data\DemoModels\DigitalIO_block_demo_rappid_rtw\DigitalIO_block_demo.mot, 0x0, 0x478C,
Checked

*********************************************************************
* Operation Setup - Specifies what operation(s) has been selected
* 1. Program/Erase (Erase, EraseProgram)
*********************************************************************
[OP_SETUP]
EraseProgram

*********************************************************************
* End of File
*********************************************************************
```

As shown the file content has six different sections, namely, Format Version, Communication Setup, MCU Setup, BAM Setup, File Info (Application Setup) and Operation Setup. These six sections are part of RAppID Boot loader GUI also. Data entered in the file is configured with respect to the GUI component, once the .rbl file is loaded in the utility.

4.3 Boot loader MCU File (*.rbm) Format

Here is a sample .rbm file, which is part of Boot loader utility installation.

Listing: Sample .rbm File

```
*********************************************************************
* This file specifies the format version of the rbl file
* This is needed for compatibility checks between tool versions
*********************************************************************
*********************************************************************
[FORMAT_VERSION]
3

*********************************************************************
* This file specifies which MCUs are supported and which RAppID
* Boot loader Algorithm (RBA) file shall be used to program
* application code in the flash.  The .rba file is in s19 format.
* 1. File name  (MPC5604P.rba, MPC5604B.rba, MPC5604S.rba, .....)
* 2. Start Address(Start Address of rba file)
* 3. Size   (Size of rba file in bytes)
* 4. RBA Baud  (Baud rate supported by rba file)
* 5. BAM Baud  (Baud rate supported by MCU BAM)
* 6. AutoBaud  (1 = AutoBaud; 0 = Fixed Baud)
* 7. BAM Type  (Type of BAM implementation designated by a number)
* 8. BAM Delay    (Delay in ms needed b/t bytes for BAM - inter byte gap)(if set to 0.00
full duplex enabled)
* 9. CCP Delay    (Delay in ms needed b/t msgs for CCP - inter message gap)(if set to 0.00
full duplex enabled)
*********************************************************************
[MPC5604/3P]
RBA_Files\MPC5604P.rba, 0x40000100, 0x3000, 115200, 9600, 0, 1, 0.00, 0.00

[MPC5605/6/7B]
RBA_Files\MPC5607B.rba, 0x40000100, 0x3000, 115200, 9600, 0, 1, 0.00, 0.00
```

```
[MPC564xL]
RBA_Files\MPC564xL.rba, 0x40000100, 0x2300, 115200, 115200, 1, 1, 0.00, 0.00

[MPC567xK]
RBA_Files\MPC567xK.rba, 0x40000100, 0x3000, 115200, 115200, 1, 1, 0.00, 0.00

[MPC567xF]
RBA_Files\MPC567xF.rba, 0x40000100, 0x3000, 115200, 115200, 1, 2, 0.00, 0.00

[PXS20xx]
RBA_Files\PXS20xx.rba, 0x40000100, 0x2300, 115200, 115200, 1, 1, 0.00, 0.00

[PXS30xx]
RBA_Files\PXS30xx.rba, 0x40000100, 0x3000, 115200, 115200, 1, 1, 0.00, 0.00

[PXR40xx]
RBA_Files\PXR40xx.rba, 0x40000100, 0x3000, 115200, 115200, 1, 2, 0.00, 0.00

*********************************************************************
* End of File
*********************************************************************
```

As shown the file content has two different sections, namely, Format Version and the MCUs supported. This file normally should stay untouched by the user but there may be configuration with certain USB to Serial Converters where the download rate is very slow do to chip sets and windows drivers. In these cases it is possible to switch the communication from full duplex to fix time delays. This can be used for both the download of the Bootloader to RAM through the BAM and download of the application to FLASH/RAM through the Bootloader using the CCP protocol stack. If this is desirable a user can change parameters 8 and 9 to get the desired download speed (1.0ms delay time is a good starting point). This may result in operation that is not as reliable so some tuning may be necessary.

Index

V

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Revision August 07, 2013