FTF FREESCALE TECHNOLOGY FORUM 2014

Overview of FreeMASTER and RAppID Bootloader

FTF-AUT-F0014

John H. Floros | Senior Software Engineer

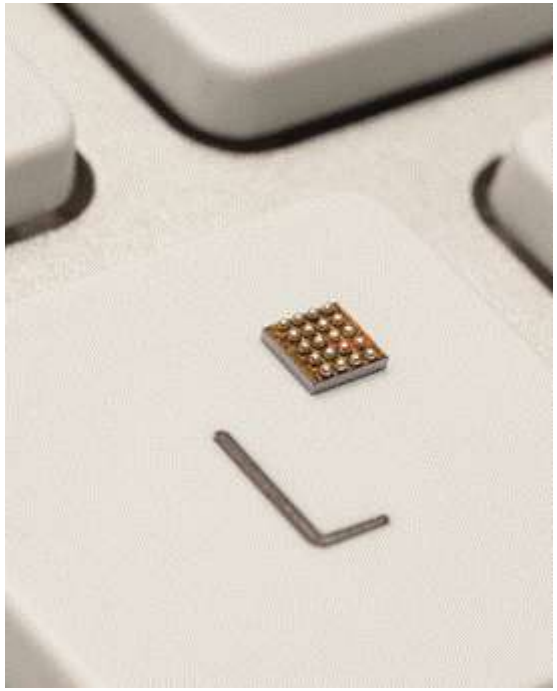
A P R . 2 0 1 4

freescale™

External Use

Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, Qorivva, SafeAssure, the SafeAssure logo, StarCore, Symphony and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airfast, BeeKit, BeeStack, CoreNet, Flexis, Layerscape, MagniV, MXC, Platform in a Package, QorIQ Qonverge, QUICC Engine, Ready Play, SMARTMOS, Tower, TurboLink, UMEMS, Vybrid and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2014 Freescale Semiconductor, Inc.

Introduction: What Do We Do?



- The Automotive Silicon Support Tools group's objective is to develop software enablement tools to assist our customers with rapid prototyping and accelerate algorithm development on their target Freescale MCU
- This includes software tools that automatically generate peripheral initialization code through GUI configuration, to generating peripheral driver code from a Model Based Design environment like Simulink™

FreeMASTER Overview
A Real-Time Monitor for Your Freescale MCU

Agenda

- FreeMASTER Part 1
- What is FreeMASTER
  - FreeMASTER Features
    - Real Time Monitor
    - Data Trace
    - Data Visualization
- Steps to integrate FreeMASTER in your Application
- FreeMASTER Demo

- RAppID Bootloader Part 2
- RAppID Boot Loader Overview
- RAppID Boot Loader Summary of Operation
- RAppID Boot Loader Input File Description
  - Configuration File (.rbl) Content
  - RAppID Boot Tool MCU File (.rbm) Content
- RAppID Boot Loader Setup Using GUI
- RAppID Boot Loader Future Enhancements

Timeline

- FreeMASTER Part 1
- Session: 35 minutes
  - Introduction and Objectives
  - What is FreeMASTER
  - FreeMASTER Features
    - Real Time Monitor
    - Data Trace
    - Data Visualization
  - Steps to integrate FreeMASTER in your Application
  - Freemaster Demo
- Summary and Q&A: 5 minutes

Objectives

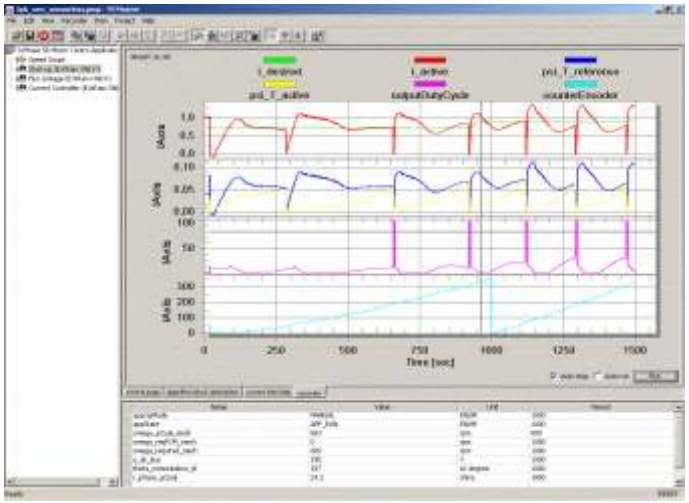
After this FreeMASTER Overview, you will know:

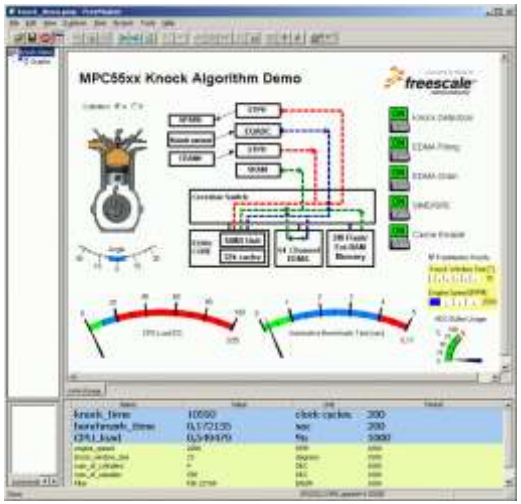
- **What FreeMASTER is and what features it contains for real time monitoring of your application on the MCU**
- **How to configure some of the features available in the FreeMASTER user interface**
- **The steps required to enable FreeMASTER in your application at a high level**

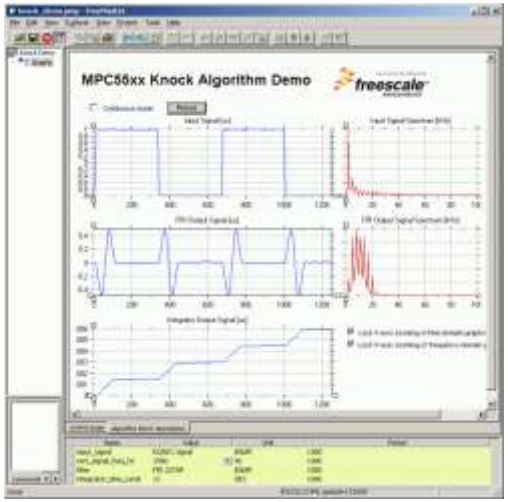
What is FreeMASTER?

- Real-time Monitor
- Graphical Control Panel
- Demonstration Platform & Selling Tool

FOR YOUR EMBEDDED APPLICATION





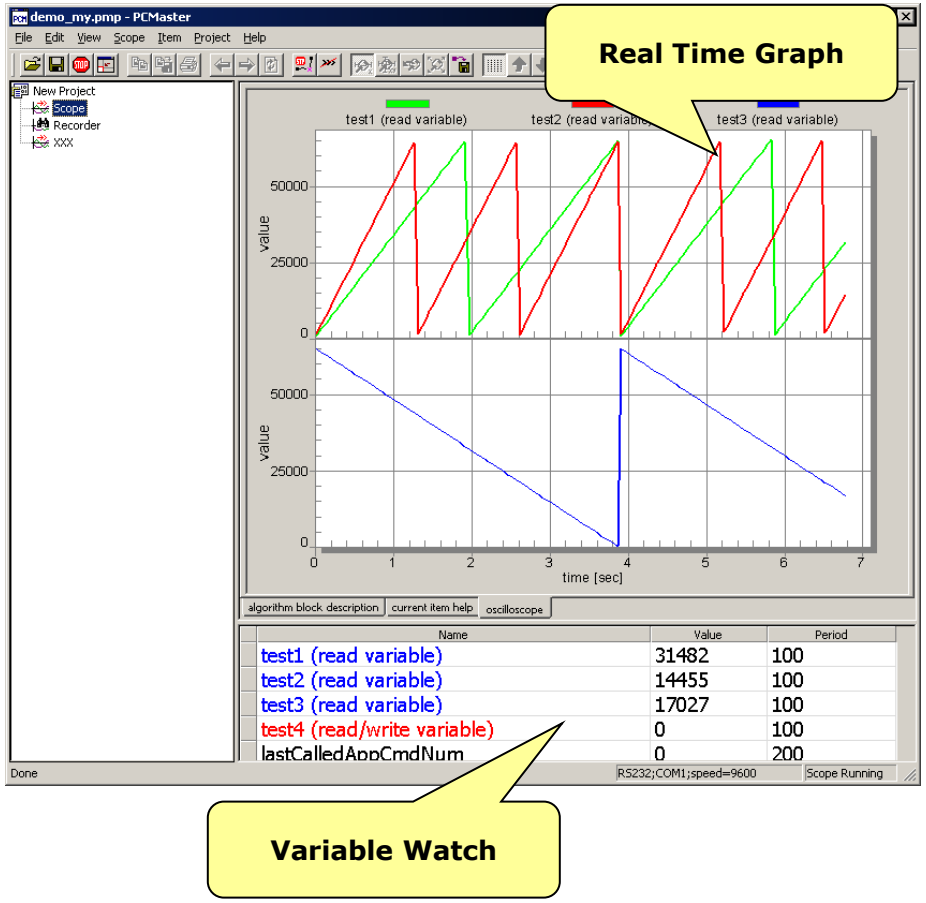


FreeMASTER as a Real-Time Monitor

- Connects to an embedded application
  - SCI, UART
  - JTAG/EOnCE (56F8xxx only)
  - BDM (HCS08, HCS12 only)
  - CAN calibration protocol
  - Ethernet, TCP/IP
  - Any of the above remotely over the network
- Enables access to application memory
  - Parses ELF application executable file
  - Parses DWARF debugging information in the ELF file
  - Knows addresses of global and static C-variables
  - Knows variable sizes, structure types, array dimensions etc.

FreeMASTER as a Real-Time Monitor

Displays the variable values in various formats:



- **Text**, tabular grid
  - Variable name
  - Value as hex, dec or bin number
  - Min, max values
  - Number-to-text labels
- **Real-time waveforms**
  - Up to 8 variables simultaneously in an oscilloscope-like graph
- **High-speed recorded data**
  - Up to 8 variables in on-board memory **transient recorder**

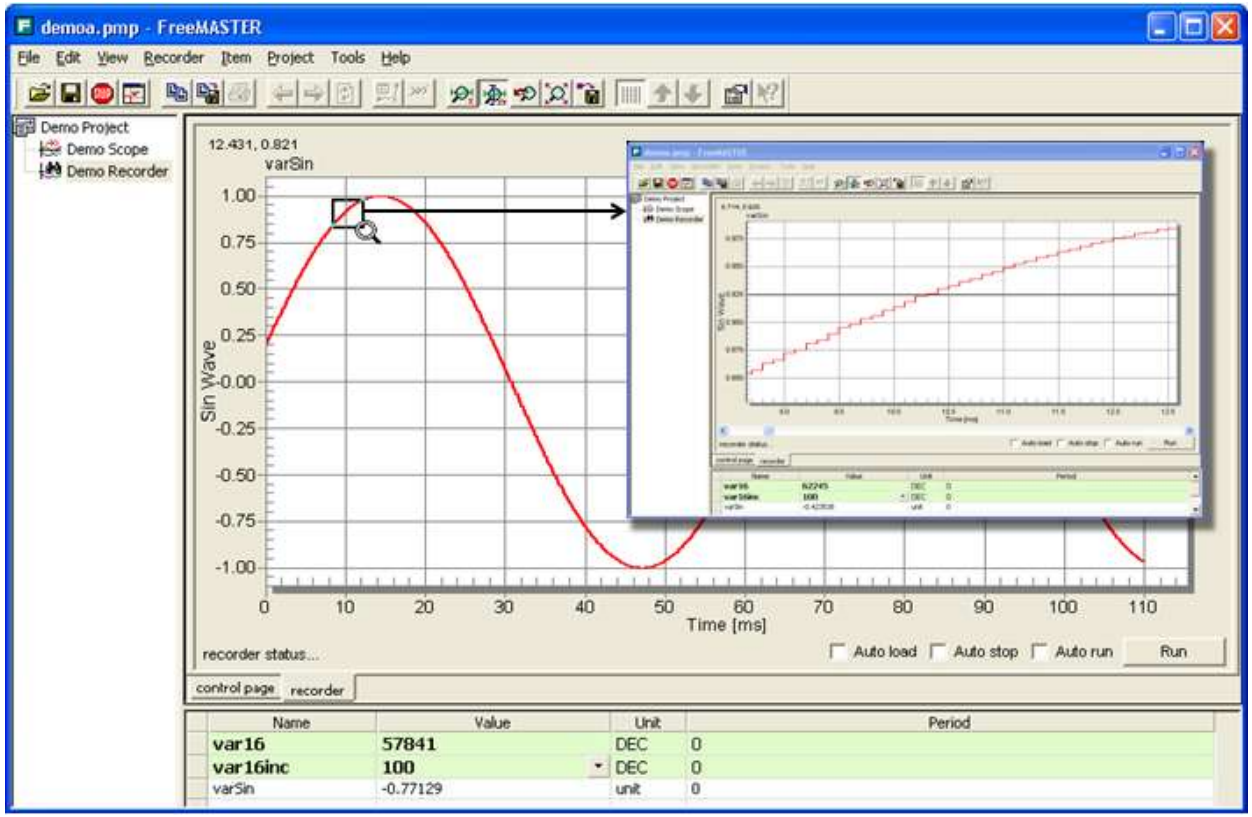
FreeMASTER as a Real-Time Monitor

Additional features:

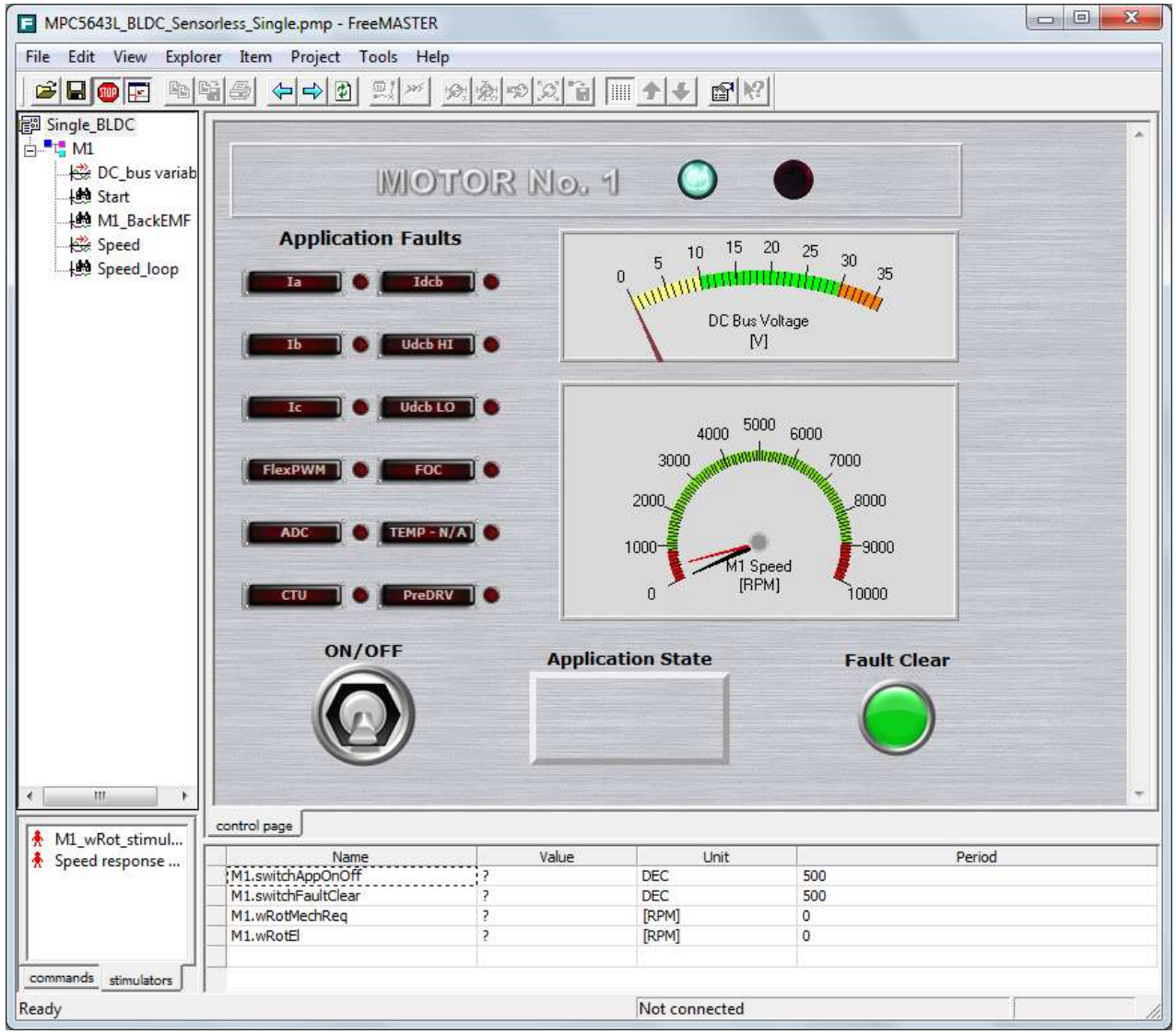
- Variable transformations
  - Variable value can be transformed to custom unit
  - Variable transformations may reference other variable values
  - Values are transformed back when writing a new value to variable
- Application commands
  - Command code and parameters are delivered to an application for arbitrary processing
  - After processing (asynchronously to a command delivery) the command result code is returned to PC
- Ability to protect memory regions
  - Describing variables visible to FreeMASTER
  - Declaring variables as read-write to read-only for FreeMASTER
    - The access is guarded by the embedded-side driver

FreeMASTER Data Trace

- **Establish a Data Trace on Target**
  - Set up buffer (up to 64KB), sampling rate and trigger
  - Near 10-µs resolution



FreeMASTER Data Visualization



The HTML-based data visualization area. The user can provide any collection of ActiveX-based instrumentation to create custom visual dashboards as complex or elegant as desired.

The data visualization area may also be used to display arbitrary information, such as presentations, help files and fact sheets.

Steps

1. Include the files under the **FreeMASTER Serial Communication Vxx\src_common** in your application code project with no changes.
2. One file changed in **FreeMASTER Serial Communication V1.6\src_platforms\MPC56xx** directory:
   a) renamed freemaster_cfg.h.example to freemaster_cgh.h
   b) Confiigure freemaster by changing macro definitions
3. Addition to main.c
   a) Add function call FMSTR_Init() after system init
   b) Add function call FMSTR_Poll(); in main loop
4. To build with FreeMASTER support for MPC56xx, include all files under **FreeMASTER Serial Communication V1.6\src_platforms\MPC56xx** and **FreeMASTER Serial Communication V1.6\src_platforms\MPC56xx** directories.

DEMO

Any Questions?

RAppID Bootloader Overview Programming for your Freescale MCU

Agenda

- RAppID Bootloader Part 2
- RAppID Boot Loader Overview
- RAppID Boot Loader Summary of Operation
- RAppID Boot Loader Input File Description
  - Configuration File (.rbl) Content
  - RAppID Boot Tool MCU File (.rbm) Content
- RAppID Boot Loader Setup Using GUI
- RAppID Boot Loader Future Enhancements

Timeline

- RAppID Bootloader Part 2
- Session: 15 minutes
  - Introduction and Objectives
  - What is the RAppID Bootloader Utility
  - Summary of Operation
  - RAppID Bootloader Utility Using the GUI
  - RAppID Bootloader Utility Trace
  - Demo
- Summary and Q&A: 5 minutes

Objectives

After this Bootloader Overview, you will know:

- What the RAppID Bootloader Utility is and what features it contains for programming your Freescale MCU.
- The steps required to use the RAppID Bootloader Utility in your application.
- How to configure some of the features available in the RAppID Bootloader Utility user interface.

RAppID Boot Loader Overview

The Boot Loader provides a streamlined method for programming code into FLASH or RAM on either target EVBs or custom boards using Freescale MCUs. Once programming is complete the application code automatically starts.

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, …).

Communication Interface to MCU

The Boot Loader is able to interface to the MCU with the supported communication interfaces supported by the BAM which are serial (UART) interfaces and CAN.

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 these setting to be brought in from a pre-set configuration file (.rbl) or to be able to save a configuration file based on the current GUI settings.

Features

Interfaces with the Boot Assist Module (BAM) on Qorivva MCUs. Other MCUs supported with Flash based bootloader.

Allows the programming of application code to flash or RAM with a specified S19 formatted hex file.

Allows the user to also perform a bulk erase and read memory.

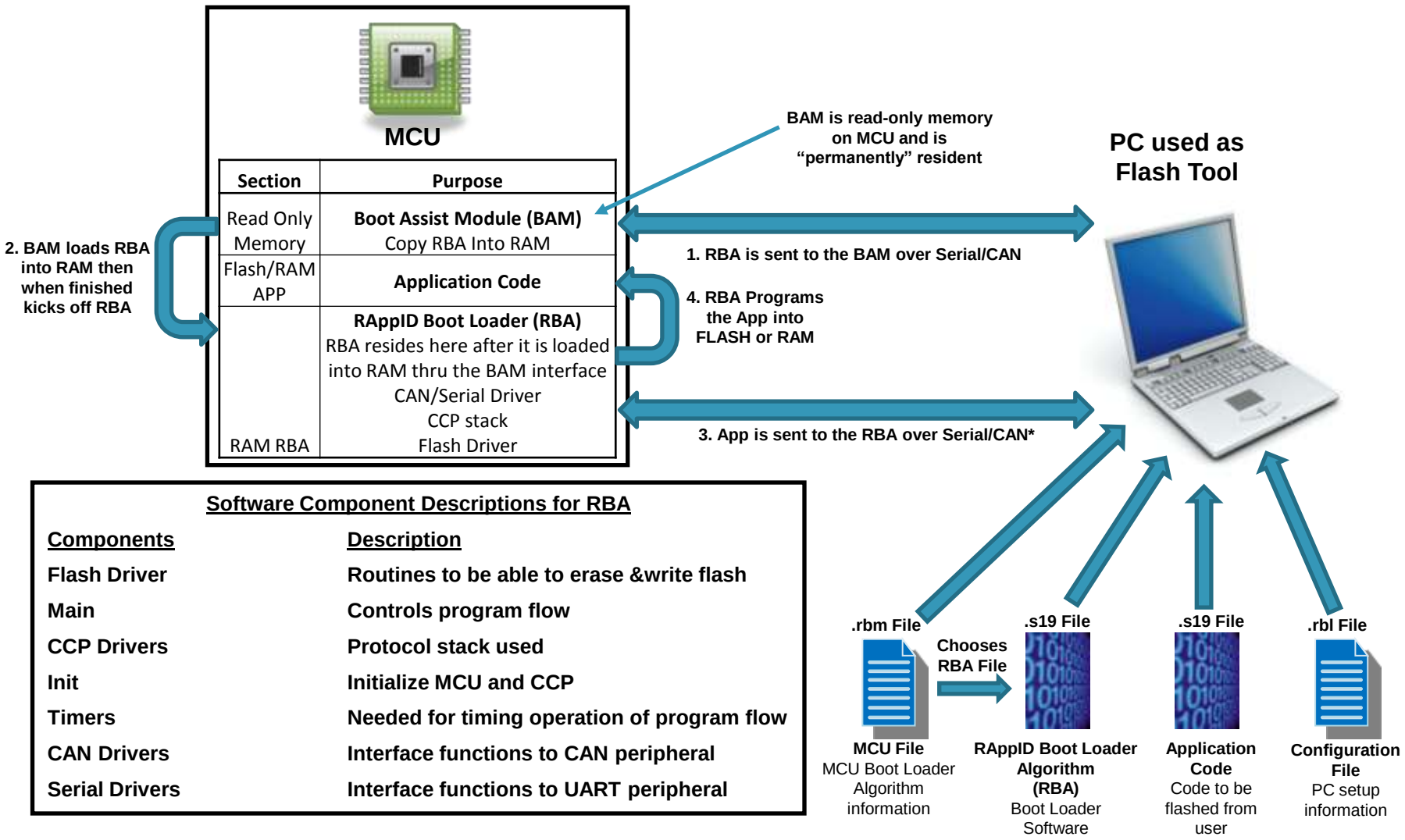
There is also to option to enable a trace window to see the communication b/t Host and target.

MCUs Supported

MPC5534, MPC5601/2D, MPC5602/3/4BC, MPC5605/6/7B, MPC564xB/C, MPC567xF, MPC567xK, MPC564xL, MPC5604/3P, MPC574XP*, S12ZVM*, KV10* and 56F82xx*

* Flash based Bootloader

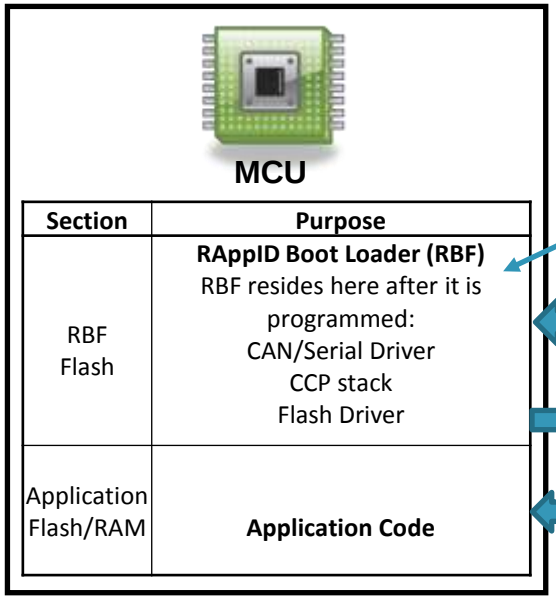
RAppID Boot Loader Summary of Operation



Software Component Descriptions for RBA

Components	Description
Flash Driver	Routines to be able to erase &write flash
Main	Controls program flow
CCP Drivers	Protocol stack used
Init	Initialize MCU and CCP
Timers	Needed for timing operation of program flow
CAN Drivers	Interface functions to CAN peripheral
Serial Drivers	Interface functions to UART peripheral

RAppID Boot Loader Summary of Operation



RBF is "permanently" resident once programmed in by user

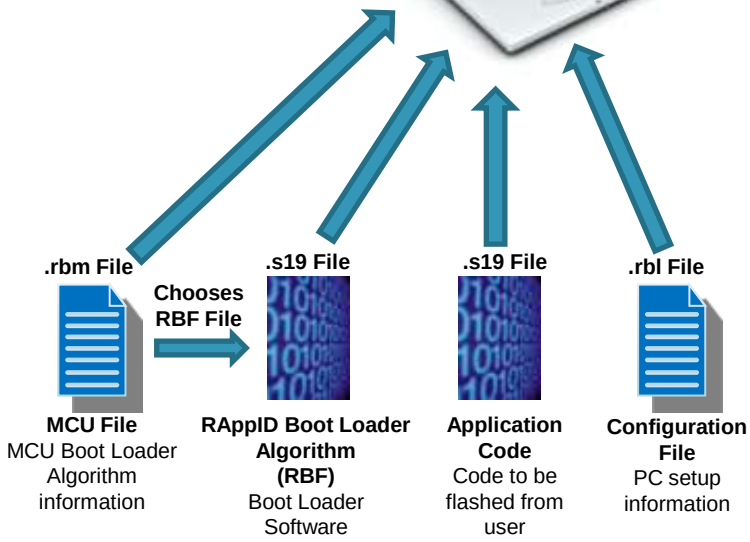
PC used as Flash Tool

1. App is sent to the RBF over Serial/CAN*

2. RBF Programs the App into FLASH or RAM

Software Component Descriptions for RBF

Components	Description
Flash Driver	Routines to be able to erase &write flash
Main	Controls program flow
CCP Drivers	Protocol stack used
Init	Initialize MCU and CCP
Timers	Needed for timing operation of program flow
CAN Drivers	Interface functions to CAN peripheral
Serial Drivers	Interface functions to UART peripheral



RAppID Boot Loader Input File Description

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

RAppID Boot Loader MCU (.rbm) File

Contains the information required to pick the correct algorithm file (.s19) to be used by the Boot Loader based on the MCU selected. Other details about the MCU can also be added to this file to extend the Boot Loader features.

RAppID Boot Loader Algorithm (.rba) File

The RBA is the target MCU-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 MCU selection. Also, based on the MCU selection, the RBM will automatically determine the appropriate interface protocol for the BAM, so that it can download the RBA into RAM through the BAM. There will be different .rba files to support multiple MCU types. The .rba file is in s19 format.

RAppID Boot Loader Algorithm (.rba) File

The RBF is the target MCU-specific algorithm which the GUI interfaces to for boot loader operation. What is different than the RBA is that it is a flash implementation where the user must program the RBF using a third party tool before using the GUI. The Boot Loader application is structured so that when using an RBF it will skip the BTL download process and start programming the application code. When using an RBF supported MCU there are some parameters that need to be considered to be included into the application — specifically a parameter defining the delay time to jump to the application (optional) and an application key (required). 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 Boot Loader Config File (.rbl) Content

```
*********************************************************************
*                RAppID Boot Tool Configuration 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]
3
*********************************************************************
* 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, COM7, 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).
*********************************************************************
[BAM_SETUP]
Enabled, 0xFEEDFACECAFEBEEF, Checked
*********************************************************************
* 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\projects\DigitalIO_block_demo_rappid_rtw\DigitalIO_block_demo.mot, 0x40001000, 0x8400, Checked
*********************************************************************
* Operation Setup - Specifies what operation(s) has been selected
*                   1. Program/Erase    (Erase, EraseProgram)
*********************************************************************
[OP_SETUP]
EraseProgram

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

RAppID Boot Loader MCU File (.rbm) Content

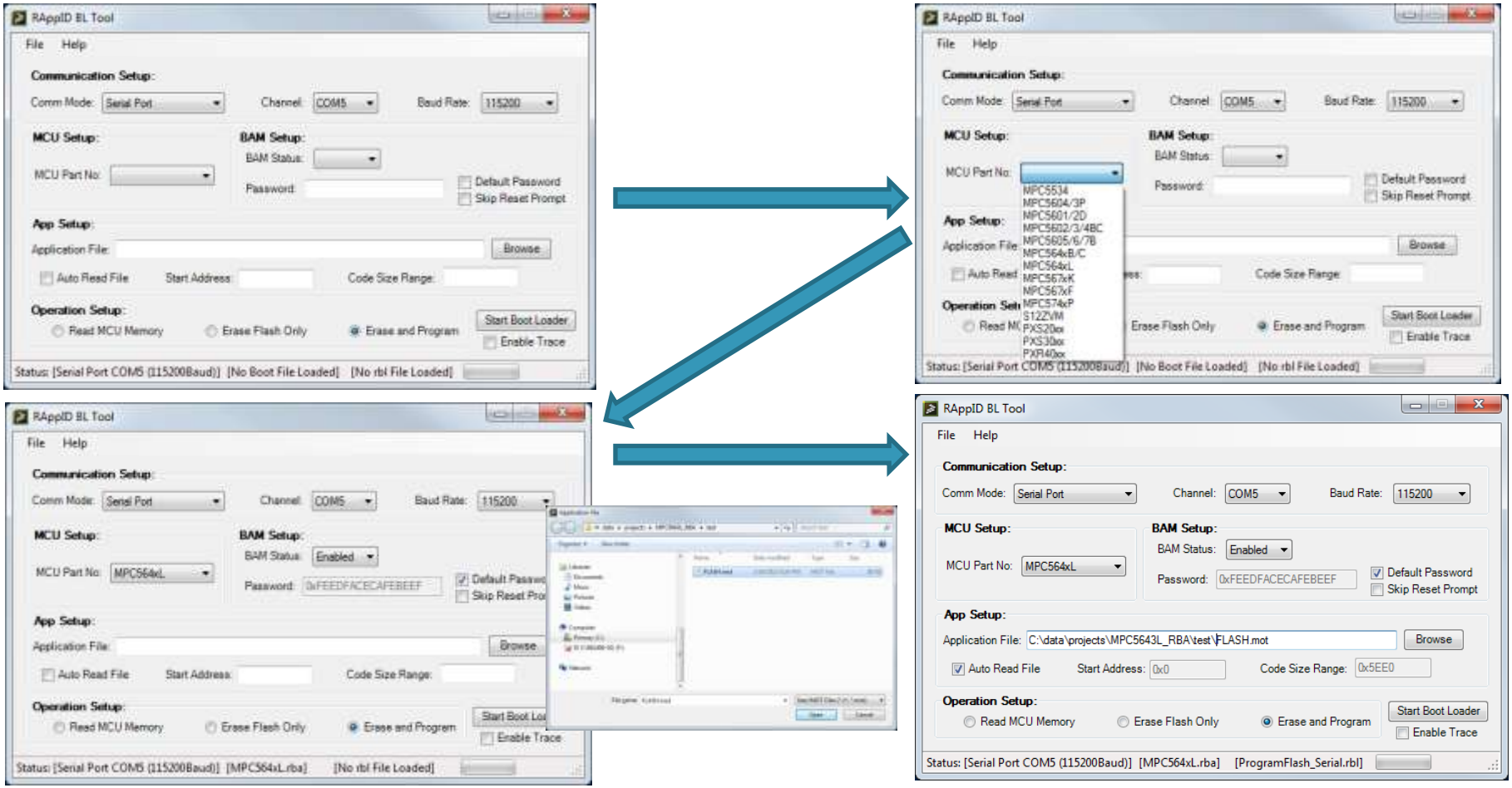
```
**********************************************************************
*                  RAppID Boot Tool MCU File
**********************************************************************
**********************************************************************
**********************************************************************
* 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. 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)
**********************************************************************
**********************************************************************
[MPC564xL]
RBA_Files\MPC5604P.rba, 0x40000100, 0x3000, 115200, 9600, 0, 1, 0.00, 0.00
.
.
.
.
.
.
.
.

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

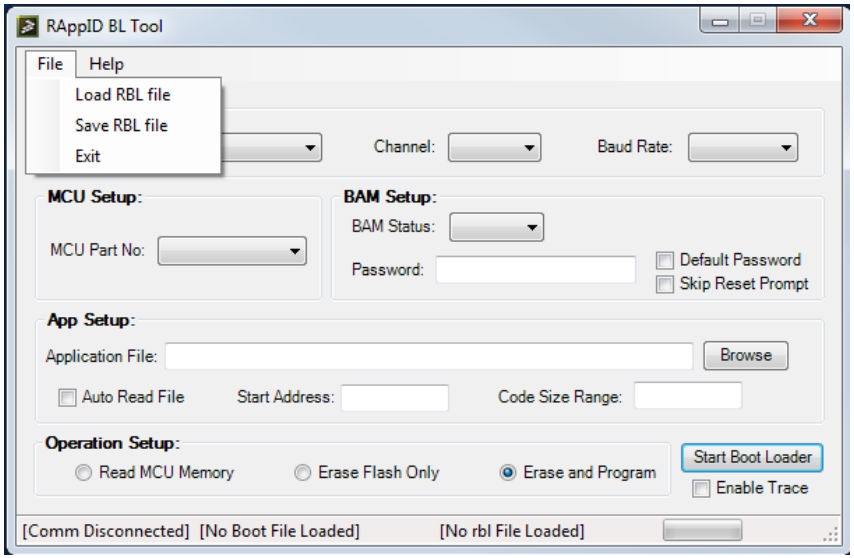
RAppID Boot Loader Setup Using GUI

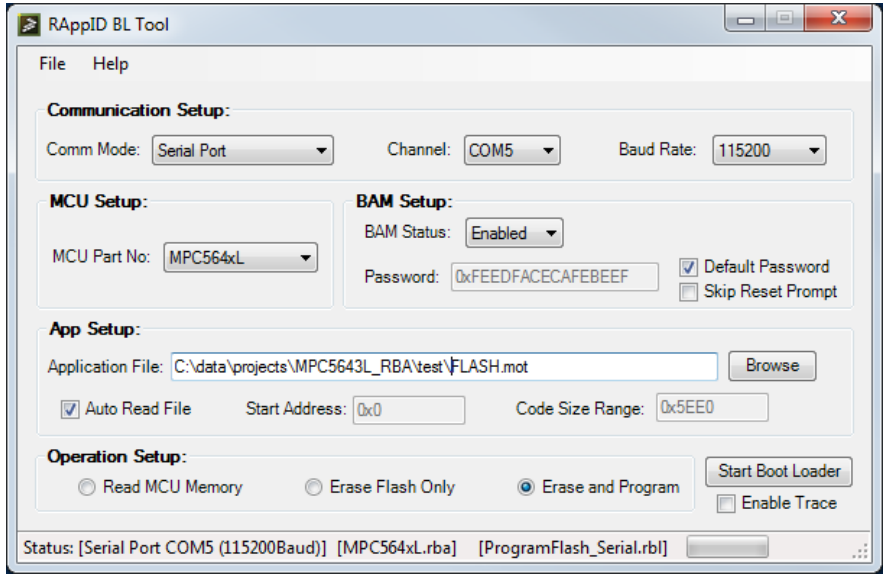
- With the GUI the user can change the settings in the GUI after loading a .rbl file or start configuring without loading a .rbl file.



RAppID Boot Loader Setup Using RBL File

- With the GUI you will be able to load in the RBL file that contains the configuration information that is set up in the file.

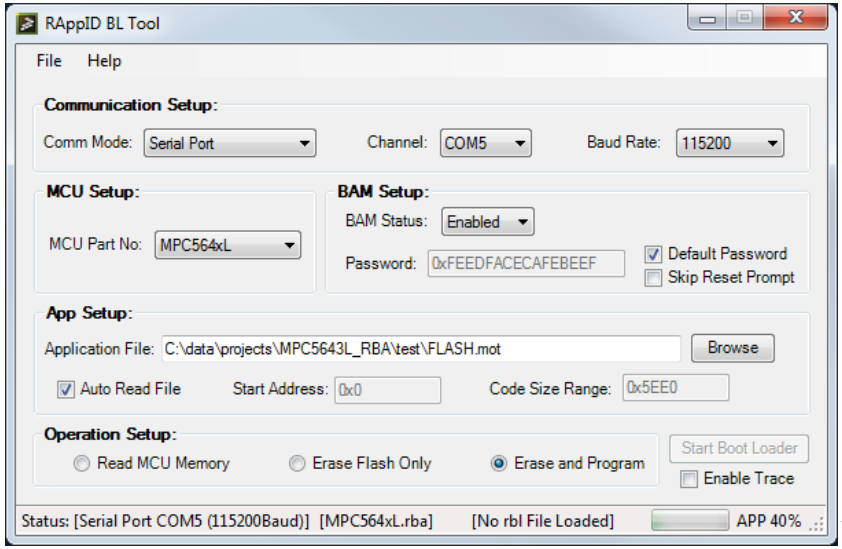




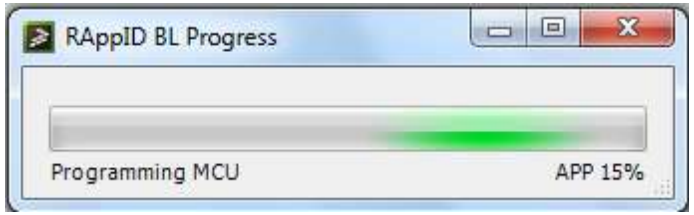
RAppID Boot Loader Status Information

- When programming is started, whether with the GUI or with the Command Line, a real-time status update shows the percentage complete.

Status information when using GUI



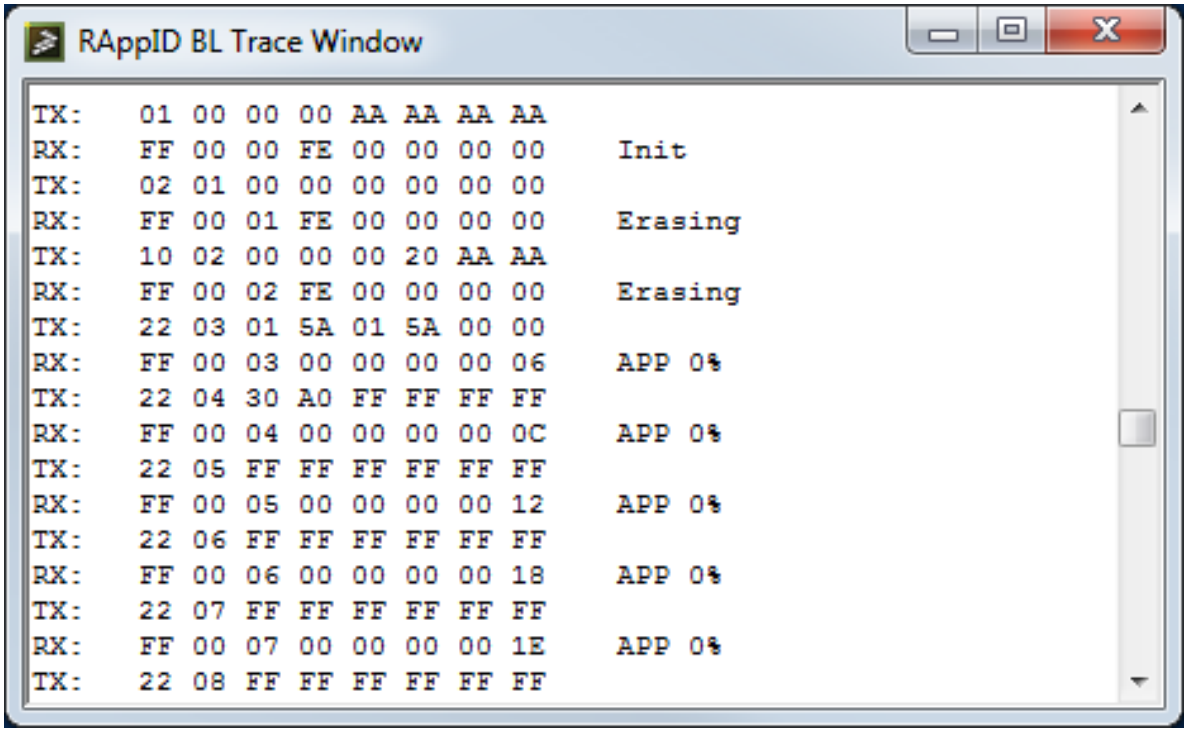
Status information when using Command Line



Status given in two stages Boot Loader Download and then Application Programming

RAppID Boot Loader Trace Window

- 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 MCU 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 when the check box is selected.



Example Trace Window Capture

Demo

Any Questions?

Summary: More Information

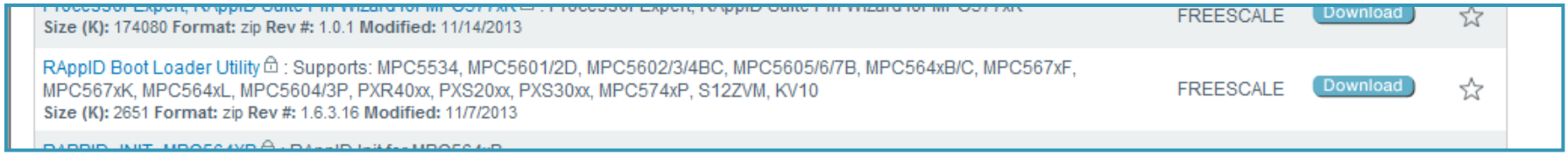
For Freemaster, visit the Freescale website:

www.freescale.com/Freemaster

To Download the RAppID Bootloader Utility:

www.freescale.com/rappid

- Then click the download tab and find the following:



Summary

Any Questions?

Please Fill Out Your Surveys.

Thank you for your time.

www.Freescale.com

© 2014 Freescale Semiconductor, Inc. | ***External Use***