RAppID Boot Loader Utility
Version 1.6.7.35 Release Notes

Targets Supported by RBA: MPC5534, MPC5601/2D, MPC5602/3/4BC, MPC5605/6/7B, MPC564xB/C, MPC567xF, MPC567xK, MPC564xA, MPC5605/6/7BK, MPC564xL, MPC5604/3P, PXR40xx, PXS20xx, PXS30xx.

Targets Supported by RBF: MPC574xP, MPC5746R, MPC5746C, MPC5748G, MPC5777C, MPC5775K, S12ZVC, S12ZVL, S12ZVM, S12VR, KEAZN16/32/64, KEAZ64/128, S32K144, KV10, MC56F82xx, MC56F84xx, KV31F512, KV31F256, KV31F128, KV30F128, KV4xFxxx.

Targets Supporting Serial communication: All

Targets Supporting CAN communication: MPC5534, MPC567xK, MPC564xL, MPC564xA, MPC5601/2D, MPC5602/3/4BC, MPC5605/6/7BK, MPC56075/6/7B, MPC564xB/C, MPC5604/3P, MPC574xP, MPC5746R, MPC5746C, MPC5748G, MPC5777C, MPC5775K, S12ZVC, S12ZVL, S12ZVM, KEAZN16/32/64, KEAZ64/128, S32K144, PXS20xx, PXS30xx.

Known Limitations:

> The MPC57xx bootloader occupies the following memory locations:
The MPC574xP and MPC5775K RBF bootloader occupies 16K memory block starting at 0x00F98000.
The MPC5746R RBF bootloader occupies 2 16K memory block - 0x00F9C000 and 00xFA0000
The MPC5748G and MPC5746C RBF bootloader occupies 16K memory block starting at 0x00F8C000.
The MPC5777C RBF bootloader occupies 16K memory block starting at 0x00000000.

The user should take care not to generate application code in this address space. To start the application, the bootloader expects following boot header information to be present at the start location of any of the small, medium or large blocks other than block occupied by bootloader:

Size (bytes)	Offset (bytes)	Value	Description
-----	------	-------	-----------
4	0	0x0000005A	RCHW
4	4	Address	Address of start of application
4	8	Delay	Delay in micro seconds before bootloader launches application on powerup
4	12	App key address	The location of application key. The application key value should be 0x55AA55AA

> The MPC5746R RBF support a 20MHz crystal input. Please contact RAppID@freescale.com for support of other crystal frequency inputs for this MCU.

> The MC56F84xx bootloader will start the application as long as it sees a valid RCHW at any 2048-byte aligned Flash location starting at 0x1800. A valid RCHW is 0x015A015A.
Then the next 4-bytes shall contain the start location of the application, then the next 4-bytes shall contain the delay before the bootloader start the application, and then the next 4-bytes contains the location of the application key. Below is an example of the necessary c-code (also see section 2.7 in the user manual for more information):

```
#pragma define_section p_flash_ROM_data "appkey.text" RX
#pragma section p_flash_ROM_data begin
const volatile uint32_t APPKEY = 0x55AA55AA;
#pragma section p_flash_ROM_data end
#pragma define_section boot_info "boot_info.text" RX
#pragma section boot_info begin
const volatile uint32_t RCHW = 0x015A015A;
const volatile uint32_t APP = (uint32_t)__entry_point;
const volatile uint32_t DELAY = 5000000;
const volatile uint32_t APPLOC = (uint32_t)&APPKEY;
#pragma section boot_info end
```

> The KV3xFxxx and KV4xFxxx bootloaders will start the application as long as it sees a valid RCHW

at any 2048-byte aligned Flash location starting at 0x2000. A valid RCHW is 0x015A015A.
Then the next 4-bytes shall contain the start location of the application, then the next 4-bytes shall contain the delay before the bootloader start the application, and then the next 4-bytes contains the location of the application key. Below is an example of the necessary c-code (also see section 2.7 in the user manual for more information):

```
const volatile uint32_t APPKEY __attribute__ ((section(".appkey"))) = 0x55AA55AA;
const volatile uint32_t RCHW __attribute__ ((section(".cfmconfig"))) = 0x015A015A;
const volatile uint32_t APP __attribute__ ((section(".cfmconfig"))) = (uint32_t)__thumb_startup;
const volatile uint32_t DELAY __attribute__ ((section(".cfmconfig"))) = 5000000;
const volatile uint32_t APPLOC __attribute__ ((section(".cfmconfig"))) = (uint32_t)&APPKEY;
```

> Please reduce as much as possible the possibility of a reset or power being removed during any erase (at the beginning of programming or just an erase). An incomplete erase can leave the flash in an incorrect state causing the bootloader to fail. If there is a need to have the bootloader operate when the flash is in an incorrect state please contact rappid@freescale.com.
> When an erase occurs for Erase Flash only or Erase and Program all of Flash memory is erased.
> There is a MC56F82xx bootloader which specifically supports the Tower card TWR-56F8200.
> The MC56F82xx bootloader will start the application as long as it sees a valid RCHW at any 1024-byte aligned Flash location starting at 0x1400. A valid RCHW is 0x015A015A.
Then the next 4-bytes shall contain the start location of the application, then the next 4-bytes shall contain the delay before the bootloader start the application, and then the next 4-bytes contains the location of the application key. Below is an example of the necessary c-code (also see section 2.7 in the user manual for more information):

```
#pragma define_section p_flash_ROM_data "appkey.text" RX
#pragma section p_flash_ROM_data begin
const volatile uint32_t APPKEY = 0x55AA55AA;
#pragma section p_flash_ROM_data end
#pragma define_section boot_info "boot_info.text" RX
#pragma section boot_info begin
const volatile uint32_t RCHW = 0x015A015A;
const volatile uint32_t APP = (uint32_t)__entry_point;
const volatile uint32_t DELAY = 5000000;
const volatile uint32_t APPLOC = (uint32_t)&APPKEY;
#pragma section boot_info end
```

> The KV10 RBF will only support a 10MHz crystal input. Please contact RAppID@freescale.com for support of other crystal frequency inputs for this MCU.

> The KV10 bootloader will start the application as long as it sees a valid RCHW at any 1024-byte aligned Flash location starting at 0x1800. A valid RCHW is 0x015A015A.
Then the next 4-bytes shall contain the start location of the application, then the next 4-bytes shall contain the delay before the bootloader start the application, and then the next 4-bytes contains the location of the application key. Below is an example of the necessary c-code (also see section 2.7 in the user manual for more information):

```
const volatile uint32_t APPKEY __attribute__ ((section(".appkey"))) = 0x55AA55AA;
const volatile uint32_t RCHW __attribute__ ((section(".cfmconfig"))) = 0x015A015A;
const volatile uint32_t APP __attribute__ ((section(".cfmconfig"))) = (uint32_t)__thumb_startup;
const volatile uint32_t DELAY __attribute__ ((section(".cfmconfig"))) = 5000000;
const volatile uint32_t APPLOC __attribute__ ((section(".cfmconfig"))) = (uint32_t)&APPKEY;
```

The KEAZN64, KEAZ128 and S32K144 bootloaders will start the application as long as it sees a valid RCHW at any 2048-byte (KEAxxx) or 512-byte (S32K144) aligned Flash location starting at 0x2000. A valid RCHW is 0x015A015A.
Then the next 4-bytes shall contain the start location of the application, then the next 4-bytes shall contain the delay before the bootloader start the application, and then the next 4-bytes contains the location of the application key. Below is an example of the necessary c-code

```
const volatile uint32_t APPKEY __attribute__ ((section(".appkey"))) = 0x55AA55AA;
const volatile uint32_t RCHW __attribute__ ((section(".cfmConfigure"))) = 0x015A015A;
const volatile uint32_t APP __attribute__ ((section(".cfmConfigure"))) = (uint32_t)__thumb_startup;
const volatile uint32_t DELAY __attribute__ ((section(".cfmConfigure"))) = 5000000;
const volatile uint32_t APPLOC __attribute__ ((section(".cfmConfigure"))) = (uint32_t)&APPKEY
```

> The S12ZVM, S12ZVC, S12ZVL and S12VR bootloader will start the application as long as it sees a valid RCHW at any 512-byte aligned Flash location. A valid RCHW is 0x015A015A. Then the next 4-bytes shall contain the start location of the application, then the next 4-bytes shall contain the delay before the bootloader start the application, and then the next 4-bytes contains the location of the application key. Below is an example of the necessary c-code for S12ZVM, S12ZVC for S12ZVL:
const uint32_t APPKEY @0xFFDFF0 = 0x55AA55AA; /* Application Key at end of App code */
const uint32_t RCHW @0xFE0000 = 0x015A015A; /* Valid RCHW Reset Config Half Word */
const uint32_t APP @0xFE0004 = 0xFFFE0013; /* start of application */
const uint32_t DELAY @0xFE0008 = 5000000; /* delay 5 seconds to start application */
const uint32_t APPLOC @0xFE000C = 0xFFFFDFF0; /* Location of Application Key */

> The MPC55534 RBA will only support a 8MHz crystal input. Please contact RAppID@freescale.com for support of other crystal frequency inputs for this MCU.

> The MPC5604/3P can only run at the fixed baud setting (ABS2 = 0) and only with an 8MHz crystal input. The baud rate selection must be set for 9600 baud for proper operation.

> The MPC5601/2D, MPC5602/3/4BC, MPC56075/6/7B & MPC564xB/C can only run at the fixed baud setting (ABS2 = 0) and with an 8MHz, 12MHz, & 16MHz crystal input. The baud rate selection must be set for 9600, 14400, & 19200 baud (based on the crystal input) for proper operation.

> The MPC567xF RBA will only support a 40MHz crystal input. Please contact RAppID@freescale.com for support of other crystal frequency inputs for this MCU.

> For MCUs using RBA support the SRAM between 0x40000000 to 0x40005000 is not available for applications that run out of SRAM. This area of SRAM is used by the Bootloader. This restriction does not apply for applications that run from FLASH memory.

> Data Flash programming is not supported.

> For MPC564xL the Boot Loader can't program SRAM memory locations after address 0x4000FFFF (after first 64KB of SRAM). As the memory location after this is not initialized by the boot loader.

> For MPC5674K the Boot Loader can't program SRAM memory locations after address 0x4004000 (after first 256KB of SRAM). As the memory location after this is not initialized by the boot loader.

> For MPC5673K the Boot Loader can't program SRAM memory locations after address 0x4002000 (after first 128KB of SRAM). As the memory location after this is not initialized by the boot loader.

Notes:
> Supports Windows XP, 7.
> It is required to use .NET Framework Version 4.0.
> Vector Hardware required to communicate over CAN.
> Required to have a RS232 COM Port to communicate with the target hardware.
> For Tower systems the serial card used for testing was the TWR-IND-CTRL.
> The code size range displayed in the GUI is the difference between the Highest Flash/RAM Address and Lowest Flash/RAM Address in the MOT (S-record) file. It is not necessarily the number of bytes that will be programmed.
> The start address displayed is the start address for programming and not necessarily the application start address for execution.

New with 1.6.7.35
Added support for MPC5748G, MPC5746C, MPC5777C, S12ZVC, S12ZVL, S12ZVM, S12VR, KEAZN16/32/64, KEAZ64/128, S32K144

New with 1.6.7.26
Added support for MPC5775K.

New with 1.6.6.25
Added support for MPC5746R.

New with 1.6.6.24
Added support for MC56F84xx. Added support for 1M CAN for BTL download of RBA file.

New with 1.6.5.23
Added support for KV4xFxxx. Uses UART1.

New with 1.6.5.22
Added support for MPC564xA.

New with 1.6.5.21
Added support for KV30F128. This is a flash based bootloader.

New with 1.6.5.20
Added support for KV31F512, KV31F256, and KV31F128. These are a flash based bootloader.

New with 1.6.4.17
Added support for MC56F82xx. This is a flash based bootloader.
Updated rba file for the MPC5604P.
New with 1.6.3.16
Added support for KV10. This is a flash based bootloader.
Added support for MPC5605/6/7B.
New serial baud rate selections available.
Added Hex file filter for browse file menu.
Added CANCardXL support.

New with 1.6.2.14
Added support for S12ZVM. This is a flash based bootloader.
MPC567xK, PXS30xx and MPC5604/3P now support CAN.
New configuration item to allow user to avoid reset MCU request prompt if not needed.

New with 1.6.1.13
Modified Vector Hardware Configuration to use the name "RAppID_BL"
to recongnize the RAppID Boot Loader Utility instead of "xlCANdemo NET". What
drove this change is that the latest Vector Hardware driver does not support
spaces in the application name.

New with 1.6.0.13
Added the ability to change the baud rate for CAN Vector tools.
Added trace window to allow monitoring of communication traffic.
Added CAN support for MPC564xL, MPC5602/3/4BC, MPC56075/6/7B & MPC564xB/C.
Added support for MPC5601/2D

New with 1.5.0.12
Added support for MPC5534.
Added CAN support for Vector tools
Added new memory read feature.

New with 1.4.6.11
Added support for MPC574xP. This is a flash based bootloader which is new method of support.
A user must take the RBF file in the RBF folder and first program the flash based bootloader into flash with a separate tool. Once flashed the bootloader will operate until it is erased by a separate tool.

New with 1.4.5.10
Added support for MPC564xB/C.

New with 1.4.5.9
Added support for MPC5602/3/4BC.

New with 1.4.5.8:

Updated so that user can change both CCP and BAM from full duplex to fixed time delays to accomodate the differences b/t USB to Serial converter cable chip sets. This can be done by changing parameters in the .rbm file.

New with 1.4.4.8:
Added capability to select Erase only or Erase and Program.
New with 1.4.2.7:
Added Cancel option in dialog box that appears before programming starts.

Contact Information
For feedback, send comments to:
Freescale Semiconductor Inc.
Automotive Silicon Support Tools
RAppID@freescale.com