ΣyG Bootloader User's Manual

Summary:

This manual covers how to program the bootloader, creating boot loadable images, building a file system, and testing procedures. It provides detailed instructions for integrating and configuring the bootloader within your system, setting up storage media for boot loading, and validating functionality for reliable boot processes.

Target Device(s):

- SYG-70CP ..... Website

Required Software:

- Renesas e2 Studio ..... Website
- Python 2.7 ..... Provided
- JLink (Latest) ..... Website
- GIMP ..... Website

Provided Files:

- SYG_BOOTLOADER.h
- srec_cat.exe
- Python/python27-32-venv
- Python/python-2.7.16.msi
- Python/README.txt
- r_fl_mot_converter.py
- srec_cat.exe
- PostProcessBCH.bat
- Scripts.zip
  - Bootloader HEX file w/ JLink scripts for automated programming of Bootloader.
  - Scripts related to testing.

For more information on FDI or our products please visit www.teamfdi.com.

NOTE: The inclusion of vendor software in the FDI software does not imply an endorsement of the product by Future Designs, Inc.

FDI PN: MA00109

Revision: 1.2, 06/10/2024

Printed in the United States of America

Contents

1. Notes about QSPI / Dataflash

After programming an image with or without a QSPI section, the bootloader writes a validation value to the first block of the SYG unit's on-chip data flash. The bootloader reserves a 64-byte block at 0x40100000. **User applications should NOT write to this area**.

2. Programming the Bootloader

1. Connect SYG unit with JLink and provide the SYG power.
2. Run "Scripts/JLink/program_bootloader.jlink"

3. Creating Boot Loadable Images

When creating boot loadable images, the start address of the application must be moved from 0x0 to 0x40000, and a special header must be put into the application for the bootloader to properly recognize it.

For development purposes, it might be desirable to maintain the ability to create images which start at 0x0 to make testing without the bootloader simpler. For this reason, the following instructions discuss how to make the project suit both purposes.

In either case, you will need to add files, duplicate and update the linker file and configure the e2studio project to automate image generation.

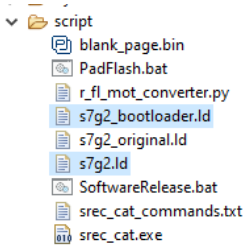
a. Updating Application Code

1. Add the provided header file into the include path of your application project:
   a. SYG_BOOTLOADER.h
2. Add the following code to your application project, updating the **version_number** and **image_identifier** as needed:

```
#include "SYG_BOOTLOADER.h"

const sf_firmware_image_file_header_t g_image_file_header
BSP_PLACE_IN_SECTION_V2(".sf_firmware_image_file_header") =
{
    .valid_mask = (uint8_t)SF_FIRMWARE_IMAGE_VALID_MASK_8_BIT,
    .version_number = (uint8_t)1,
    .image_identifier = (uint16_t)1,
    .num_blocks = 0U,
    /* The following items not used for sf_firmware_image_internal, but struct is used
       by bootloader, so they need to be initialized and the memory reserved */
    .size_bch = 0U,
    .size_raw = 0U,
    .maximum_block_size = 0U,
    .checksum_bch_file = 0U,
    .checksum_fw_image = 0U,
    .first_block_address = 0xFFFFFFFFU,
    .application_start_address = 0U,
    .num_memory_instances = 1U,
    .p_app_memory_instance = NULL,
    .successfully_stored = 0xFFFFFFFFU,
    .checksum_bch_file_header = 0U,
};
```

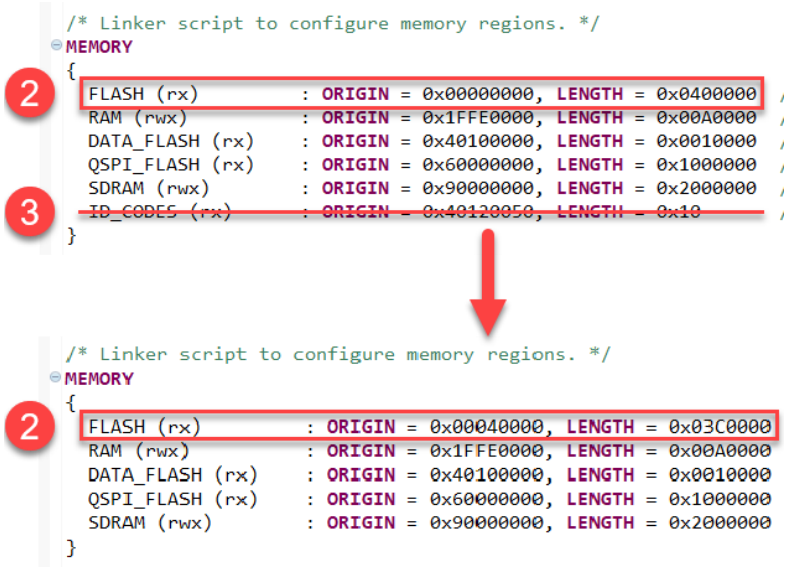
b. Updating Linker file

1. Duplicate the existing linker file to create a new linker file suitable for generating boot loadable images:



2. Update the new linker file to move the ORIGIN of the FLASH section to 0x40000, and subtract 0x40000 from the FLASH section's LENGTH.

3. Remove the "ID_CODES" section.



4. Add the following into the linker file directly before *(.text*) to place the previously defined image header at the 0x800 address:

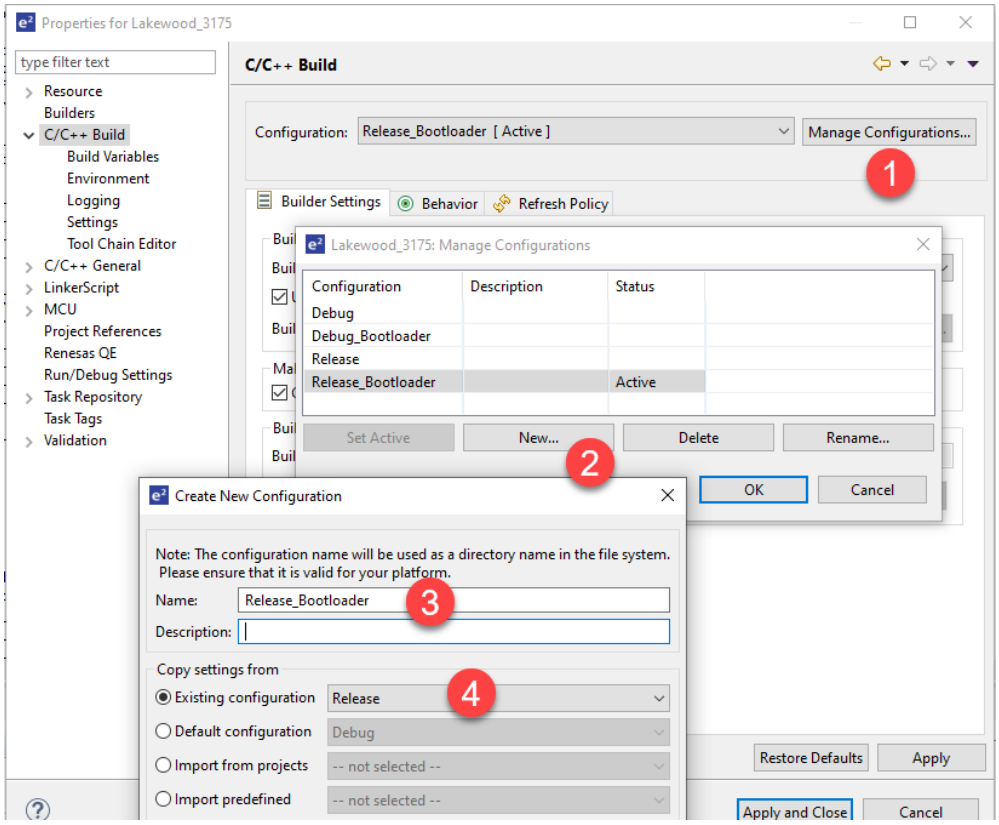
```
/* Flash Loader App Header at 0x800. This offset is fixed in sf_firmware_image.h and in XML */
. = __ROM_Start + 0x800;
KEEP(*(.sf_firmware_image_file_header*))

*(.text*)
```

5. Remove the section which places the id_codes:

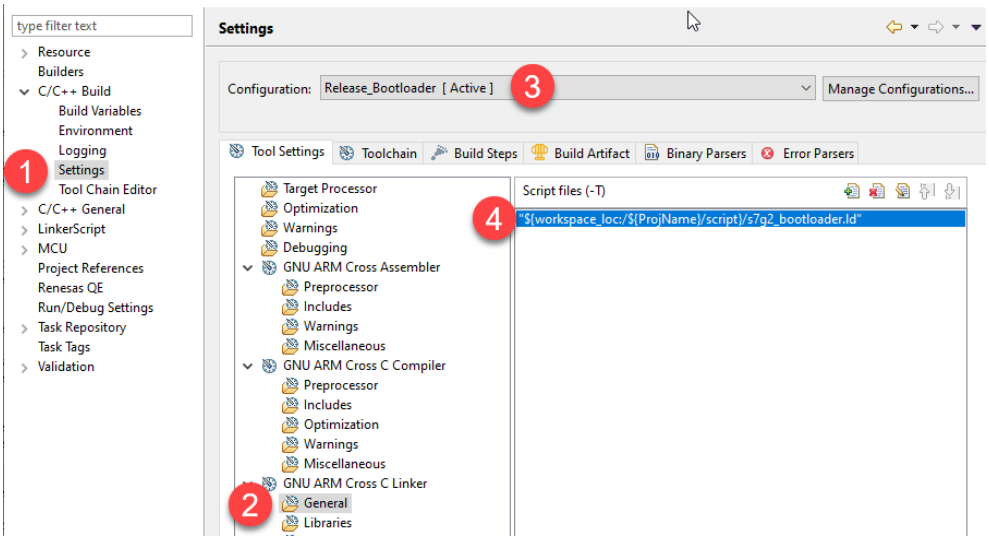
```
.id_code_1 :
{
    __ID_Codes_Start = .;
    KEEP(*(.id_code_1*))
    __ID_Codes_End = .;
} > ID_CODES
```

c. Updating e2studio Project Build Configuration

1. Duplicate an existing project build configuration to create one suitable for generating boot loadable images:



2. Update only the new bootloader project configuration(s) to point to the edited boot loadable image linker file.



d. Creating Custom Builder in e2studio

Several files have been provided to assist in the automation of image generation. Each is summarized below:

1. **r_fl_mot_converter.py**
   Python script responsible for parsing an SREC file and converting it to the bootloader-required BCH format.

2. **Python/python27-32-venv**
   Pre-configured Python virtual environment with the correct versions of the required packages needed by **r_fl_mot_converter.py**.

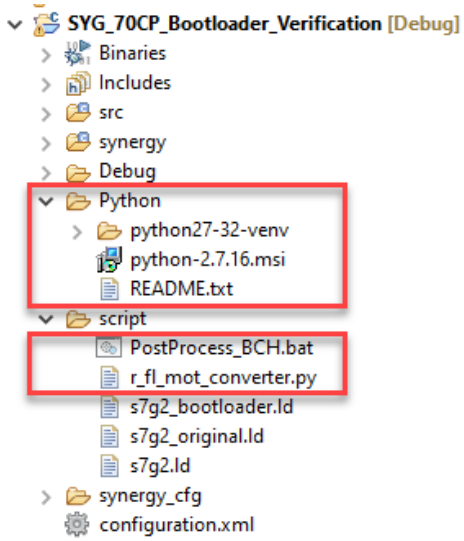
3. **Python/python-2.7.16.msi**
   Windows installer for Python 2.7

4. **Python/README.txt**
   The instructions on where to install Python to ensure the virtual environment points to it correctly.

5. **PostProcessBCH.bat**
   This is a batch script which handles the calls to **Python** which will be used by the e2studio build system to automate image generation.

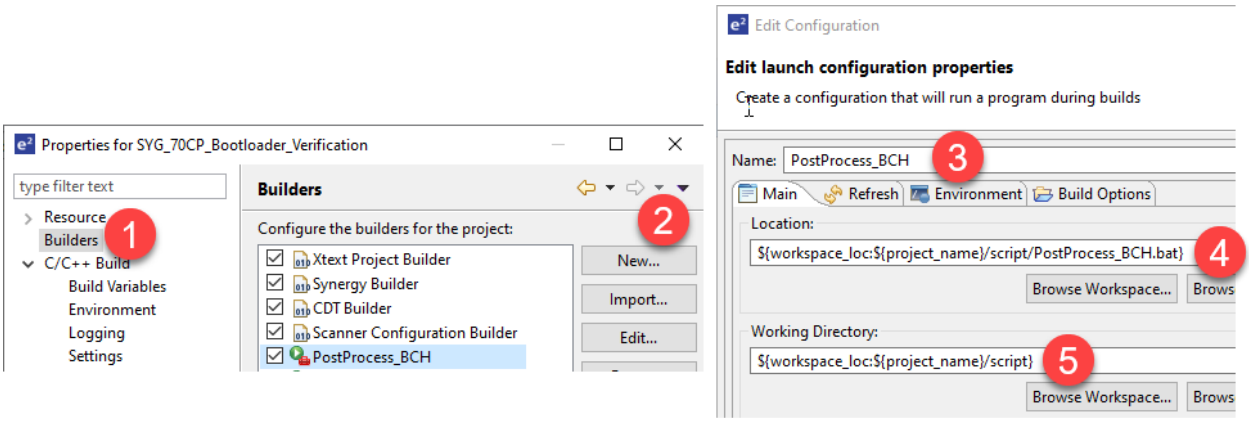
To implement the automatic post-build conversion process,

1. Add the provided files in the project at the locations shown below.



2. Install Python using the instructions in README.txt.

3. Create a new Builder in the e2studio project configuration:



4. *Creating File System*

The bootloader requires the device's file system to contain certain files for its operation.

- [PRODUCT_NAME]
  - [FILE_NAME]
  - INSTALL.INI (Product INI)
- INSTALL.INI (Root INI)

a. Root INI

The Root INI tells the bootloader where it can get a boot loadable image using the following settings. This must be placed into the root of the filesystem.

The Number (1 in this case) represents the "Image Identifier" which is chosen in the application's file header variable. This allows for having a single file system with images for multiple devices if needed.

The rest of the file path represents the folder that the bootloader can find the Product INI which will be used to determine certain options during the bootload process.

NOTE: Make sure to include a newline at the end of the file

```
[Products]
1 = SYG_70CP/INSTALL.INI
```

b. Product INI

The Product INI provides several configurable options for the bootloader.

- Image Identifier
  This should match the image identifier in the Root INI as well as what is defined in the application's file header variable.
- Path
  This should be the path to the boot loadable image file output by the conversion process.
- Attended (Default = false)
  This allows for configuring the bootloader to provide on-screen buttons for choosing various actions during the bootload process.
- Rename (Default = true)
  When setting Rename to true, after the boot load process has completed successfully, the root INSTALL.INI is renamed to INSTALL.FIN. This prevents the bootloader from re-programming the unit on every power up. When setting Rename to false, the bootloader will program the application on each power-up.

- Version Protection (Default = false)
  When setting version protection to true, the version set in the application's header file variable is compared with the version of the currently programmed version, preventing the device from programming (older? Same?) versions.
  When setting version protection to false, any image can be programmed.

NOTE: Make sure to include a newline at the end of the file

```
[App]
Image Identifier = 1
Path = SYG_70CP/SYG_70CP.bch
Attended = false
Rename = true
Version Protection = false
```

5. *Customizing Splash Screen*

When starting up, the bootloader will display a custom splash screen. Due to the lightweight nature of bootloaders, the GUI is limited in features to keep its code size small. To customize the splash screen, there are specific post-processing requirements for the image that you wish to use.

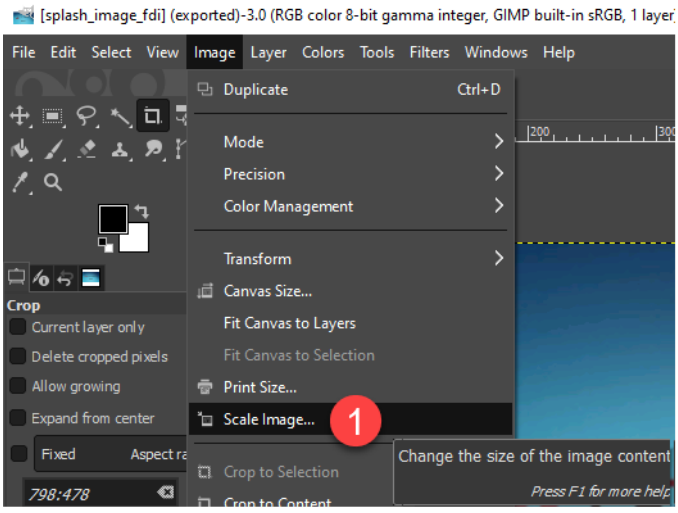
Requirements:

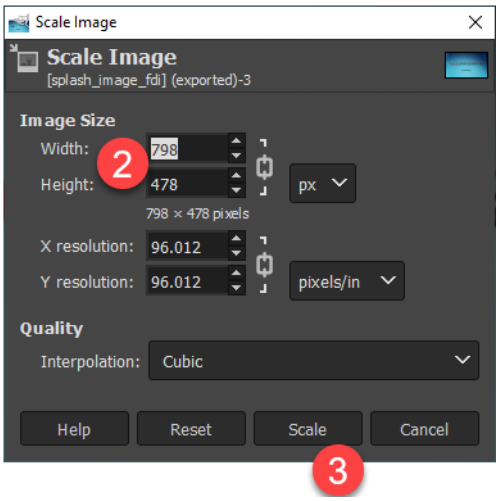
1. Image file must be named “splash_image.bmp”.
2. Image file must be on the root of the SD card.
3. Image must be 798 x 478.
4. Image must be BMP format, 16 bits per pixel.
5. Image must be flipped vertically.

The free image manipulation software GIMP can be used to modify an existing image to meet all these requirements. With the desired image opened in GIMP, follow the steps below:

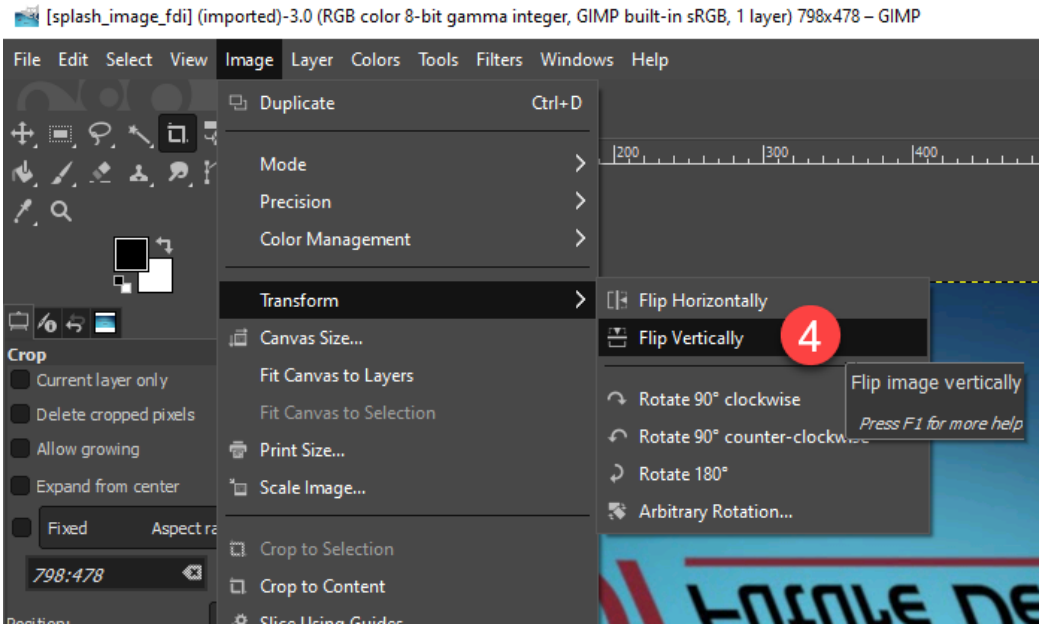
1. Select Image > Scale Image…
2. Update the Image Size Width and Height to 798 x 478.
3. Click Scale to resize the image.

NOTE: Depending on the original size of the image, you might need to try different “Interpolation” methods to get the quality you want.

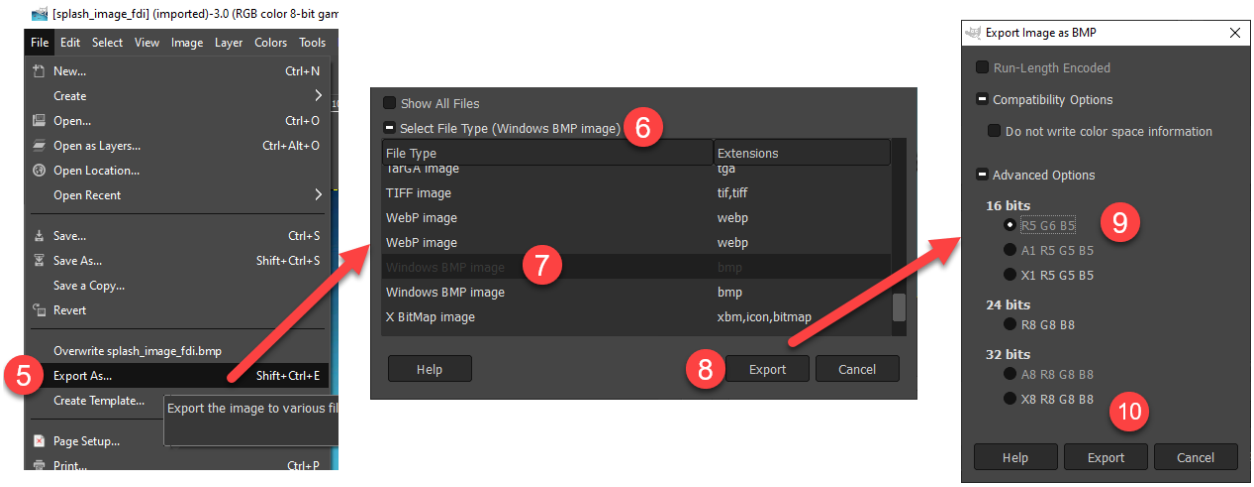




4. Select Image > Transform > Flip Vertically.



5. Select File > Export As…
6. Select “Select File Type” to expand the list of options.
7. Select “Windows BMP Image”
8. Click “Export”
9. Select “Advanced Options” and choose “16 bits”
10. Click “Export” to save the file.



6. Testing

Proper bootloader testing involves testing a variety of cases and can take a long time to test manually. For this reason, we have developed automated testing which tests the following test cases.

Information and files for how to reproduce these automated tests are provided on purchase of the bootloader and are included within the Scripts directory.

1. No Valid Image & No SD Card
2. No Valid Image & SD Card
3. No Valid Image & SD Card with Empty install.ini
4. No Valid Image & SD Card with Filled QSPI
5. No Valid Image & SD Card with Filled QSPI – Corrupted QSPI Image
6. No Valid Image & SD Card with Image ID Mismatch
7. No Valid Image & SD Card with Valid Update - Interrupt flash write
8. No Valid Image & SD Card with Invalid Application Path
9. No Valid Image & SD Card with Invalid Install Path
10. No Valid Image & SD Card with Invalid Update – Rev 1
11. No Valid Image & SD Card with Invalid Update – Rev 2
12. No Valid Image & SD Card with Missing install.ini
13. No Valid Image & SD Card with Missing Product BCH file
14. No Valid Image & SD Card with Missing Product Directory
15. No Valid Image & SD Card with Missing Product INI file
16. No Valid Image & SD Card with Missing Splash Image
17. No Valid Image & SD Card with Valid Update – Rev 1
18. No Valid Image & SD Card with Valid Update – Rev 2
19. Valid Image & No SD Card
20. Valid Image & SD Card with Valid Update - Same Version
21. Valid Image & SD Card with Empty install.ini
22. Valid Image & SD Card with Filled QSPI
23. Valid Image & SD Card with Filled QSPI – Corrupted QSPI Image
24. Valid Image & SD Card with Image ID Mismatch
25. Valid Image & SD Card with Valid Update - Interrupt flash write
26. Valid Image & SD Card with Invalid Application Path
27. Valid Image & SD Card with Invalid Install Path
28. Valid Image & SD Card with Invalid Update – Rev 1
29. Valid Image & SD Card with Invalid Update – Rev 2
30. Valid Image & SD Card with Missing install.ini
31. Valid Image & SD Card with Missing Product BCH file
32. Valid Image & SD Card with Missing Product Directory
33. Valid Image & SD Card with Missing Product INI File
34. Valid Image & SD Card with Missing Splash Image

35. Valid Image & SD Card with Valid Update – Version Downgrade
36. Valid Image & SD Card with Valid Update – Version Upgrade