Release Notes
For
Unmanaged Software Development Kit

UM 3.8.0

Table of Contents

Section 1: About This Document

These are the Release Notes for the Broadcom Unmanaged Software Development Kit, Release 3.8.0.
This document provides a general description of the release, the chips and platforms supported by the release.

Section 2: New Devices Added to This Release

Support for the following devices has been introduced in this release.

Family	*Devices*	*Description*
BCM56070	BCM56072 A0	Ethernet Switch (embeds ARM CPU) supports up to a maximum of 3x (4x10Q Serdes core) and 4x (4x25G Serdes core)
	BCM56071N A0	Ethernet Switch (embeds ARM CPU) supports up to a maximum of 3x (4x10Q Serdes core) and 4x (4x25G Serdes core). And this SKU only supports option 5 and option 6.

Section 3: Device and Platform Support

Section 3.1: SOC/Switch Devices

The following devices are supported in this software release.

Family	*Device*	*Description*
BCM56070	BCM56072 A0	Ethernet Switch (embeds ARM CPU) supports up to a maximum of 3x (4x10Q Serdes core) and 4x (4x25G Serdes core)
	BCM56071N A0	Ethernet Switch (embeds ARM CPU) supports up to a maximum of 3x (4x10Q Serdes core) and 4x (4x25G Serdes core). And this SKU only supports option 5 and option 6.

Section 3.2: Reference Design

The following Switch Reference Design is available from Broadcom and is supported in this software release.

Platform	*Description*
BCM956070K	5607x Switch Ref Design

Section 4: Feature Support

The Broadcom BCM56071N/BCM56072 devices are a class of high-performance, non-blocking network switching devices supporting up to a maximum of 3x (4x10Q Serdes core) and 4x (4x25G Serdes core), as well as various combinations of these port configurations.

This UM release only supports features to let BCM56071N/BCM56072 devices be used as a channelized adjunct line card port fan-out switch. This UM release supports the UM+ package only. That means the UM-Web is not supported in this UM release. So most UM-WEB features (for example, the LAG feature and the Mirror feature) are not supported in this UM release. Below list shows features supported in this release. And the complete UM feature support list is available in the file “UM-3.8.0 feature support list.pdf” under the directory $UM/RELDOCS in the release package.

BTW, the QSGMII/QXGMII port configuration is not supported in this UM release.

Section 4.1: BCM56071N/BCM56072 A0 Feature Maturity Level

Feature	***Maturity***
UART with 115200 bps	GA
PHYMOD Driver	GA
CPU TX/RX	GA
Flash Driver	GA
Persistence	GA
Loader	GA
Dual Image	GA
Vendor Config	GA
VLAN APIs	GA
Statistic APIs	GA
MMU Setting per Customer Requirement	GA
Various Port Configs by Vendor Config	GA
PORT APIs	GA
QoS	GA
COE	GA
Pass Through Mode	GA
Broadsync	GA
SyncE APIs	GA
1-Step and 2-Step TC APIs	GA
Interrupt Mode	GA
GPIO APIs	GA
M0 Code for LED	GA
M0 Code for Hardware Linkscan	GA
Exception Handler	GA
WatchDog Mechanism	GA

Temperature Query API	GA
Example Code of Simple SPI Management	GA

Section 5: API Support

Complete API documentations are available in the file “UM-3.8.0 API function support list.pdf” and “API_GUIDE ” under the directory $UM/RELDOCS in the release package.

Broadcom does not guarantee API default values set within the UM SW and changes to default values may be made between releases. If an API default value is required for application software to work properly, it must be explicitly set.

Section 6: Things to Note

Section 6.1: Build Steps

Section 6.1.1 Toolchain

The toolchain (GNU GCC ARM Embedded 4.8 update, https://launchpad.net/gcc-arm-embedded/4.8/4.8-2014-q1-update) is used to develop this UM SW for BCM5607X platform.

Customers can follow below steps to use the new version of toolchain.
1. Download the new version of toolchain and install it.
2. Set the environment TOOLCHAIN_DIR to the location where the toolchain is installed. And set the environment LD_LIBRARY_PATH also if necessary.

For example, if the toolchain has been downloaded and installed in the /projects/ntsw-tools/gnu/gcc-arm-none-eabi-7-2018-q2-update folder. And $UM is the UM working directory.

```
setenv TOOLCHAIN_DIR /projects/ntsw-tools/gnu/gcc-arm-none-eabi-7-2018-q2-update
setenv LD_LIBRARY_PATH /projects/ntsw-tools/lib/glibc-2.14/lib
cd $UM/systems/bcm95607x
make target=umplus
```

Section 6.1.2 Build The Image File for BCM5607X UM+

1. set $UM to your UM directory

2. Go to the $UM/systems/bcm95607x directory and create an output directory (outputs).

```
cd $UM/systems/bcm95607x
mkdir outputs
```

3. Type the following to build the loader (bcm95607x-loader.bin) and firmware (bcm95607x-umplus.flash), and save the loader and firmware to the output directory.

```
make target=loader
cp bcm95607x-loader.bin outputs/
make target=umplus
cp bcm95607x-umplus.flash outputs/
```

4. Go to the output directory and generate a single image (bcm95607x-umplus.image) for the BCM5607X platform.

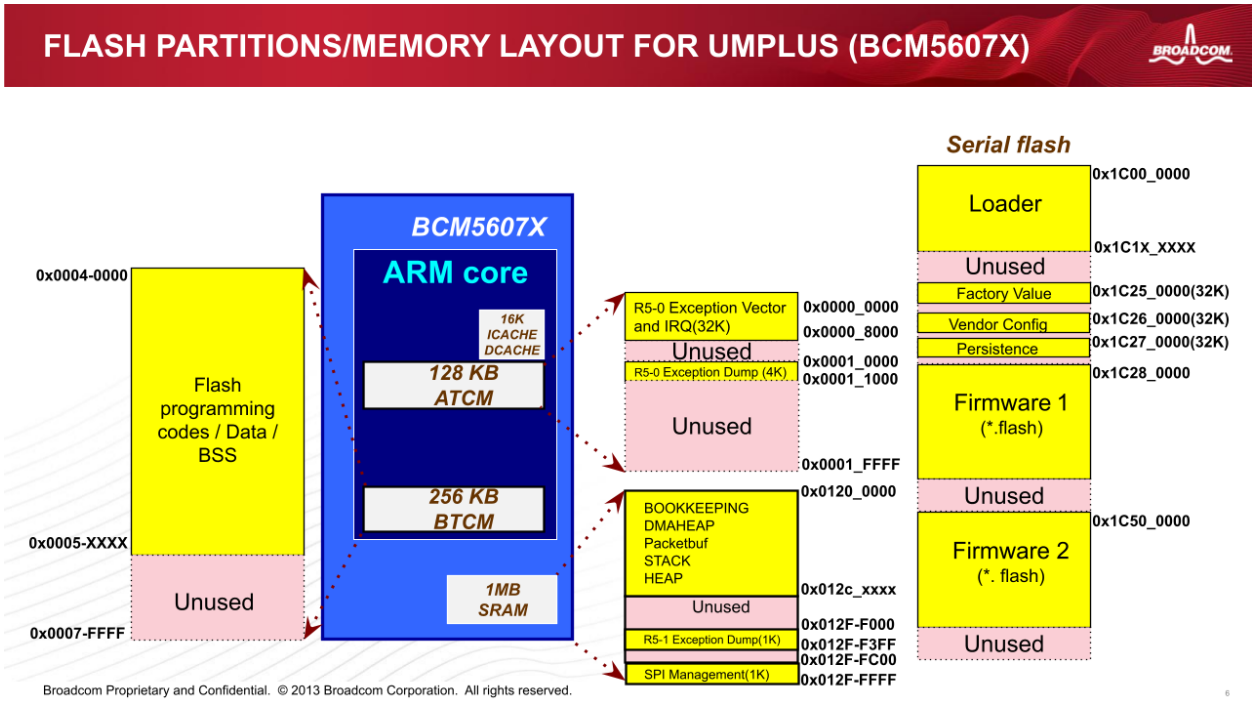
```
cd outputs/
../../../tools/mkflashimage.pl bcm95607x-loader.bin bcm95607x-umplus.flash
bcm95607x-umplus.image
```

5. (Optional) Select/edit a vendor config and insert it into the image. Assume the name of the vendor config file is config.um which is copied and renamed from $UM/systems/bcm95607x/vendor_configs/config.um.956070K_op05-10x10g_2x100g.

```
cp ../vendor_configs/config.um.956070K_op05-10x10g_2x100g config.um
../../../tools/um_config_insert.pl -c config.um -i bcm95607x-umplus.image -f
```

Section 6.2: Memory Layout and Flash Partition

The UM image (loader and firmware) is programmed in external QSPI flash. The on-chip RAM is used for the runtime data, the stack, the heap, exception handler and flash programming code. The overall memory layout and flash partition for UM software is shown as below.



Section 6.3: Using The Configuration Insert Tool

The Unmanaged Software provides a tool to insert the configuration into the precompiled firmware image (*.image). The firmware image may perform different settings in the initialization based on the configuration.

Section 6.3.1 Support Different Port Configurations by Config File

Besides the default port configuration "10P 10G (front panel, merlin core) + 2P 100G (backplane, falcon core)", the following port configurations are also supported in this release.

4P 10G (front panel, merlin core) + 8P 10G (backplane, falcon core)
4P 25G (front panel, falcon core) + 8P 25G (backplane, falcon core)
4P 25G (front panel, falcon core) + 2P 100G (backplane, falcon core)
12P 10G (front panel, merlin core) + 2P 100G (backplane, falcon core)
12P 10G (front panel, merlin core) + 1P 100G + 1P 50G (backplane, falcon core)

If customers would like to run different port configurations, they can refer to config files in the $UM/systems/bcm95607x/vendor_configs directory in the release package. The file includes lanemap configs of BCM956070K SVK board if the file name starts with "config.um.956070K".

Section 6.3.2 Using The Configuration Insert Tool

$UM/tool/um_config_insert.pl is the configuration insert tool. When given the firmware image and the configuration file, this tool will translate this configuration file into binary and insert it into the firmware image. An example of the tool is shown below.

$UM/tool/um_config_insert.pl -c config.um -i bcm95607x-umplus.image -f

The file bcm95607x-umplus.image is the firmware image. The file config.um is the configuration file. The option, -force, will insert the configuration regardless of the existence of the previous inserted configuration. For more information, execute “$UM/tool/um_config_insert.pl -h” to see the usage of the insert tool.

Usage: um_config_insert.pl -image <image_file> [options]:
-image, -i: This specifies the image file where the configuration will be inserted.
-config, -c: This optionally specifies the configuration file name. The default name is config.um.
-force, -f: This forces the previous configuration in the image file to be overwritten.
-verbose: This shows the debug log.
-h: This shows the configuration file usage.

Section 6.4: Dual Image Support

Two firmware images (bcm95607x-umplus.flash and bcm95607x-umplus-2.flash) are generated by “make target=umplus” for BCM5607X UM+. They are used for “firmware 1 partition” and “firmware 2 partition” respectively. These two images are built with separate linking base addresses CFG_TEXT_START and CFG_TEXT_START_2 which are defined in config_umplus.old. And the image ID (CFG_IMAGE_1_ID or CFG_IMAGE_2_ID) is added in the image header by “mkheader.pl”. The image ID will be checked when the loader tries to load the firmware image. Of course, the customer’s firmware upgrade application needs to check the image ID before it tries to program the firmware on flash.

Section 6.5: IFP TCAM Slice Arrangement

As in the table below, this section describes IFP TCAM rules and slices arrangement for "QoS feature", "COE mode" and "pass-through mode".

FP types	Qualifier	Slice	Entry index
COE/ Passthru mode	SrcPort CustomHeaderPkt CustomHeaderData InterfaceClassPort	0~3 double wide	31(0~30: 10 for upstream; 20 for downstream; 1 for default drop) for config.um.956070K_op05-10x10g_2x100g Note: May be more than 31 entries when downstream have multiple channels.
VLAN 802.1P	OuterVlanPri	4 single wide	0-7
MPLS EXP	ETHERTYPE DATA	4 single wide	8-15
	CustomHeaderPkt ETHERTYPE DATA	5 single wide	0-7

Section 6.6: WatchDog

The watchdog timeout timer is defined via CFG_WDT_TIMEOUT (5 seconds currently) in config_umplus.h. Max supported timeout is 51513851 microseconds (around 51.5 seconds). If a task or function code busily runs for longer than CFG_WDT_TIMEOUT, calling board_wdt_ping to keep watchdog alive is required. Otherwise watchdog will trigger reboot. In case watchdog

timeout occurs, a watchdog reset flag will be set in the HW register and triggers reset. Watchdog reset flag will be kept until the next power cycle.

Section 6.7: Broadsync Support

To build this UM SW with Broadsync support for BCM95607X platform:

1. Get the Broadsync FW release package (FIRMWARE-BCM56070_1_UM_BS-4.3.17.tar.gz) from docSAFE. Untar this package and get the Broadsync FW (BCM56070_1_um_bs.bin in binary format).
2. Create “broadsync” directory under $UM/systems/bcm95607x/binfs.
3. Copy the Broadsync FW into the directory $UM/systems/bcm95607x/binfs/broadsync
4. Uncomment “#define CFG_BROADSYNC_INCLUDED” in $UM/systems/bcm95607x/include/configs/config_umplus.h.
5. Then follow the build steps in section 6.1.2.

Broadsync FW is automatically initialized as part of the boot-up sequence with the following configuration:

- Mode: Broadsync Slave
- Bit_Clk frequency: 10Mhz
- HB frequency: 4Khz

Customers can use the BroadSync command to get the BroadSync FW version, change the default configuration, and enable debug logs or 1PPS output.

Section 6.8: COE Support

The COE chassis mode and pass through mode are supported in this release.

For customers’ reference, we provide 2 scenarios for pass through mode and 4 scenarios for chassis mode.

[Pass through mode]
Scenario 1 : 4P 10G (front panel) + 8P 10G (backplane)
The config file is config.um.56070_op05-4x10g_8x10g or
config.um.956070K_op05-4x10g_8x10g.
Scenario 2 : 4P 25G (front panel) + 8P 25G (backplane)
The config file is config.um.56070_op03-4x25g_8x25g or
config.um.956070K_op03-4x25g_8x25g.
[Chassis mode]
Scenario 3 : 4P 25G (front panel) + 2P 100G (backplane)
The config file is config.um.56070_op03-4x25g_2x100g or
config.um.956070K_op03-4x25g_2x100g.

Scenario 4 : 10P 10G (front panel) + 2P 100G (backplane)
The config file is config.um.56070_op05-10x10g_2x100g or
config.um.956070K_op05-10x10g_2x100g.
Scenario 5 : 12P 10G (front panel) + 2P 100G (backplane)
The config file is config.um.56070_op05-12x10g_2x100g or
config.um.956070K_op05-12x10g_2x100g.
Scenario 6 : 12P 10G (front panel) + 1P 100G + 1P 50G (backplane)
The config file is config.um.56070_op05-12x10g_1x100g_1x50g or
config.um.956070K_op05-12x10g_1x100g_1x50g.

Customers can get related config files from the directory $UM/systems/bcm95607x/vendor_configs. The file includes lanemap configs of BCM956070K SVK board if the file name starts with "config.um.956070K"

Section 6.8.1. Chassis Mode

Customers need to add the config variable “coe_scenario” in the config file to support COE on these 4 scenarios.

For example of scenario 4, customers need to add “coe_scenario=4” in the config file and re-insert it into the image “bcm95607x-umplus.image” again (please refer to the step #5 in section 6.1.2) before programming the flash image.

Section 6.8.2. Pass Through Mode

For COE passthrough mode, customers need to re-insert the scenario 1 or scenario 2 config file into the image “bcm95607x-umplus.image” (please refer to the step #5 in section 6.1.2) before programming the flash image. Please note that the BumpOnWire MMU is used for COE passthrough mode, the “mmu_mode=passthru” config should be included in the config file.

Section 6.9: Customer LED Firmware

Section 6.9.1 Examples of Customer LED Firmware and LED API

Customers can refer to cmicx_customer_led.c at $UM/tools/led/example for CMICx LED firmware coding. Follow the steps in section 6.9.2 to generate the binary file of LED firmware.

To built-in the customer LED firmware into UM image, the cmicx_custom_led.bin should be copied into $UM/systems/bcm5607x/binfs/cmicfw. The LED firmware will be built into the UM image after compilation.

Please refer to $UM/systems/bcm5607x/src/board_init.c for the way to load and start the customer LED firmware load in UM. During the runtime, UM will load the built-in cmicx_customer_led.bin and let the customer LED firmware run at the UM init stage by default.

As for the board LED APIs, customers can find the example of usage in $UM/src/appl/unittest/utled/utled.c. The LED API list is at $UM/include/boardapi/led.h.

Section 6.9.2 How to Build Customer LED Firmware

1. Go to $UM/tools/led/example/ and type “make” to compile the customer LED example.
2. Copy the binary of customer LED example (cmicx_customer_led.bin) to the directory $UM/systems/bcm95607x/binfs/cmicfw. This binary will be built-in automatically after the compilation of BCM5607X UM+.

Section 6.10: Interrupt Mode and GPIO

The polled and true interrupt mode both are supported in this UM release. Customers can use CFG_POLLED_INTR to select interrupt mode at compile time. The default mode is true interrupt mode.

In true interrupt mode, functions used in interrupt context should be reviewed strictly by customers to ensure they are safe to be invoked or to be reentrant in interrupt context.

The API board_intr_handling_disable and board_intr_handling_enable can be used to disable/enable the entire interrupt system. The API board_gpio_intr_enable_set can be used to disable/enable per gpio interrupt.

Basic register/memory access can be supported in the interrupt context.

The flash programming can not be used in the interrupt context. The interrupt will be disabled during flash programming.

The test case in $UM/src/appl/unittest/utgpio demonstrates how to use GPIO APIs and how to hook the GPIO interrupt handler.

Section 6.11: Hardware Linkscan

Customers need to hook a hardware linkscan handler and configure which port needs to support hardware linkscan. After configuration, the hardware linkscan will be turned on automatically. Any link change happens on hardware linkscan ports, the hardware linkscan module will send interrupts to the system. Then the system will invoke customer’s hardware linkscan callback.

Please find more details of the hardware linkscan APIs in $UM/include/boardapi/link.h. For the more details of usage, please check $UM/src/appl/unittest/utled.c.

Section 7: UM Externally Licensed Software Components

The UM contains a number of third-party externally licensed software components. This appendix contains information regarding these components, the license for each of these components, and where these components are used in UM.

Component	*Origin*	*Location in Source Tree*
EDITLINE	https://github.com/troglobit/editline	src/appl/editline

Section 7.1 EDITLINE License Terms and Conditions

```
--------------------------------------------------------------------------------

This package was obtained in 1999 and modified to fit the Broadcom
SDK. In 2015 it was modified further to perform terminal I/O through
call-backs, and several unused FSF compatibility functions were
removed. For SDK purposes, the library can still be replaced by the
FSF readline library.

The original library is maintained at GitHub:
https://github.com/troglobit/editline

--------------------------------------------------------------------------------

ORIGINAL DESCRIPTION

This is a line-editing library.  It can be linked into almost any
```

```
program to provide command-line editing and recall.

It is call-compatible with the FSF readline library, but it is a
fraction of the size (and offers fewer features).  It does not use
standard I/O.  It is distributed under a "C News-like" copyright.

------------------------------------------------------------------------------

ORIGINAL COPYRIGHT

 Copyright 1992,1993 Simmule Turner and Rich Salz.  All rights reserved.

 This software is not subject to any license of the American Telephone
 and Telegraph Company or of the Regents of the University of California.

 Permission is granted to anyone to use this software for any purpose on
 any computer system, and to alter it and redistribute it freely, subject
 to the following restrictions:
 1. The authors are not responsible for the consequences of use of this
    software, no matter how awful, even if they arise from flaws in it.
 2. The origin of this software must not be misrepresented, either by
    explicit claim or by omission.  Since few users ever read sources,
    credits must appear in the documentation.
 3. Altered versions must be plainly marked as such, and must not be
    misrepresented as being the original software.  Since few users
    ever read sources, credits must appear in the documentation.
 4. This notice may not be removed or altered.

------------------------------------------------------------------------------
```