iROS-ONU 3.0 SDK Manual

January, 2011
Document Number:
Revision 1.00

Cortina Systems, Inc. Confidential

This document contains information proprietary to Cortina Systems, Inc. (Cortina). Any use or disclosure, in whole or in part, of this information to any unauthorized party, for any purposes other than that for which it is provided is expressly prohibited except as authorized by Cortina in writing. Cortina reserves its rights to pursue both civil and criminal penalties for copying or disclosure of this material without authorization.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH CORTINA SYSTEMS® PRODUCTS.

NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT.

EXCEPT AS PROVIDED IN CORTINA'S TERMS AND CONDITIONS OF SALE OF SUCH PRODUCTS, CORTINA ASSUMES NO LIABILITY WHATSOEVER, AND CORTINA DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO THE SALE AND/OR USE OF CORTINA PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Cortina products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Cortina Systems® and the Cortina Systems logo are the trademarks or registered trademarks of Cortina Systems, Inc. and its subsidiaries in the U.S. and other countries. Other names and brands may be claimed as the property of others.

Copyright © 2007-2011 Cortina Systems, Inc. All rights reserved. (Use Date Range of publication including Intel and Immenstar publication dates if applicable.)

Revision History

Revision List
Rev 1.00 Draft – 14th Jan, 2011

Contents

1. Introduction

IROS is a distributed operating system that provides the management and control of Cortina EPON system. IROS-ONU is one of the key software component which running on Cortina ONU SoC. This document describes the detail information of iROS-ONU version 3.0 which is used for Cortina CS8016 SoC.

Following applications are supported in iROS-ONU 3.0 package

- EPON 1GE/1FE SFU
- EPON 4FE SFU
- EPON PowerGrid ONU

Cortina provide 2 separated packages for 1-Port (FE/GE) ONU and 4-Ports (4FE) ONU. Both packages can support terminal server for power grid application. These 2 packages have similar source tree and compile/build/Firmware upgrade procedures. Except switch and GPIO configuration, this document provides general information for these 2 packages.

It is recommended that the readers of this document also refer to the accompanying CS8016C hardware datasheet which documents the hardware specifications and features of the chip, and the API Manual which specifies a detailed list of the driver APIs.

1.1 Document Organization

This document is written to provide an overview of iROS-ONU 3.0.
Section 2 introduces the overall IROS software architecture spanning the OLT, ONU and SC.
Section 3 describes the detail information of iROS-ONU 3.0 software architecture and operation.
Section 4 describes development guidelines for developers.

1.2 Abbreviations

EPON	Ethernet over Passive Optical Network
OLT	Optical Line Termination
ONU	Optical Network Unit
ODN	Optical Distribution Network
UNI	User Network Interface
NNI	Network Node Interface
iROS	Cortina Operating System
API	Application Programming Interface
HA	High Availability
VLAN	Virtual Local Area Network
FDB	Filtering Database
OAM	Operation, Administration and Management

Cortina Systems, Inc. Confidential

MIB	Management Information Base
QoS	Quality of Service
DiffServ	Differential Services
DSCP	Differential Service Code Points
DBA	Dynamic Bandwidth Allocation
IGMP	Internet Group Multicast Protocol
RED	Random Early Detection
WRED	Weighted Random Early Detection
EAP	Extensible Authentication Protocol
SAL	System Abstraction Layer
IFM	Interface Manager

2. iROS System Architecture

iROS is a distributed operating system that provides the management and control of Cortina EPON system. The iROS system architecture is illustrated in

Figure 1. From the perspective of management and control, there are two domains of operation, host system and EPON system. The host system resides in the head-end system that provides the EPON services. The EPON system is the access network that is built upon the Cortina EPON chipset. iROS is responsible for the management and control of the EPON system and provides the application programming interface to the host system. iROS consists of four software components that work in concert to provide the management and control functions.

- iROS-SC: this software component referred to as system control running in the host system context is the central point of the distributed system. It communicates with the other two software components, and provides the application programming interface to the EPON applications resident in the host system.

- iROS-OOB: this software component referred to as system control running in the host system context is the central point of the distributed system. It communicates with the IROS-ONU software components via connection through the management or UNI port of ONU, and provides the application programming interface to the EPON applications resident in the host system connected to the ONU.

- iROS-OLT: this software component running on the built-in microprocessor at the CS8021 OLT device, is a gateway to the remote CS8016 ONU devices. The built-in microprocessor provides the supporting environment for iROS-OLT. In addition to the control of the OLT device, it communicates directly with iROS-SC and iROS-ONU. Typically, one instance iROS-SC will manage multiple instances of iROS-OLT. The communication between iROS-SC and iROS-OLT is taking place via one or more Ethernet management interface. This is known as an out-band management channel.

- iROS-ONU: this software component running on the built-in microprocessor at the CS8016 ONU device, is at the remote end of the iROS distributed system. The built-in microprocessor that provides the supporting environment for iROS-ONU. It communicates with iROS-OLT, and controls the operation of the ONU device. Typically, one instance iROS-OLT will manage multiple instances of iROS-ONU. The communication between iROS-OLT and iROS-ONU is taking place via the EPON channel. This is known as an in-band management channel. In addition, for iROS 3.0, the iROS ONU also exposes a System Abstraction Layer, for system vendors to directly call ONU APIs from the ONU CPU.

Cortina Systems, Inc. Confidential

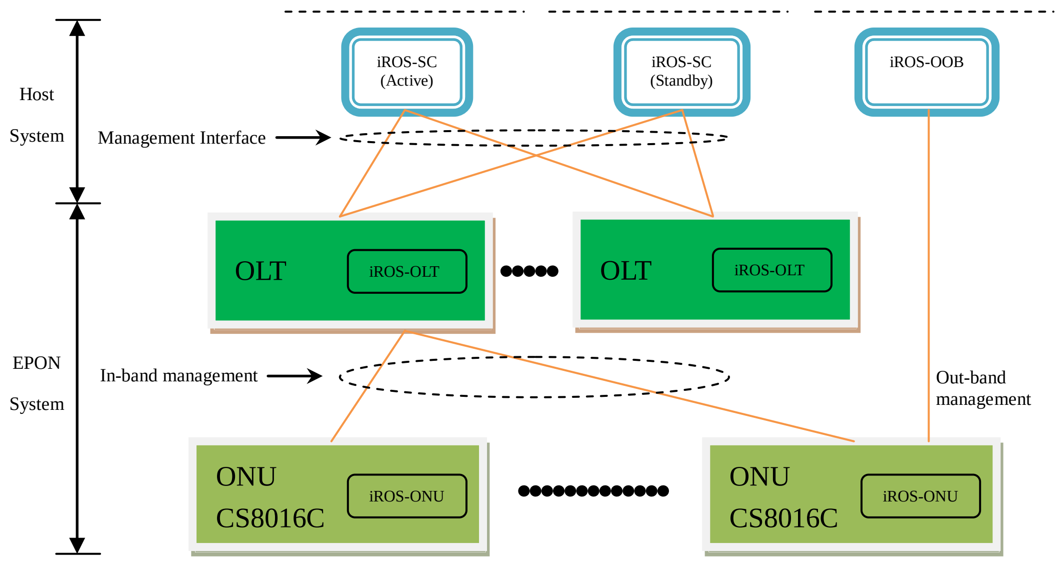


Figure 1: iROS System Architecture

iROS is designed to be fault tolerant as part of the host system. Thus there could be two instances of iROS-SC running in the host system. A typical scenario is that each iROS-SC instance runs on a separate controller card in the host system. In this configuration, if a running iROS-SC instance failed, caused by any other events (controller card failure) or its own software errors, the standby iROS-SC will take over the control of the system.

Communication between the iROS-SC and an iROS-OLT is achieved through the internal management infrastructure in the host system, typically in a switched Ethernet environment. For this purpose, each OLT device is equipped with two management Ethernet ports. Thus, an instance of iROS-OLT could interact with two iROS-SC instances simultaneously. The communication between an iROS-OLT instance and an iROS-ONU is purely in-band, through Ethernet OAM frames and vendor specific control frames. Under certain circumstances, e.g., ONU software upgrade, iROS-SC may need to exchange Ethernet frames directly with ONU. In these cases, iROS-OLT serves as an internal gateway that forwards the received frames to a desired destination within the iROS operating environment. The internal control frame between iROS components is built on the layer 3 structure. As a result, when the control frame is received from iROS-SC, iROS-OLT can simply forward the control frame, rather than terminating it, if it is destined for a remote iROS-ONU. This is because the control frame contains the destination information in the layer 3 structure. This capability will further enhance the system performance.

Cortina Systems, Inc. Confidential

3. iROS-ONU 3.0 SW Package

3.1 iROS 3.0 ONU Software Components

Figure 2 illustrate the software architecture of iROS-ONU 3.0. The iROS-ONU 3.0 software consists of three major components: platform software, ASIC Adaptation Layer (AAL), and Applications, running on the embedded ARM946 ONU CPU. In addition iROS-ONU 3.0 also provides build tools and environments for system vendors to run applications and invoke the iROS SAL APIs on the ONU CPU.

- Platform software

iROS uses the open-source eCOS real-time operating software to provide platform services(thread schedulers, system calls, interrupt service routines, synchronization constructs and message queues) to the Application and AAL components. Optionally the platform software also provides IP stack, Web Server and File System functionality for system vendor applications.

- ASIC Adaptation Layer (AAL)

The AAL is a low-level software layer that interfaces between iROS and the underlying ONU ASIC, and provides device-driver functionality for the iROS-ONU Application above. The AAL interacts with ASIC using memory-mapped registers and interrupts. The AAL is not visible to external applications and as such is released in binary-format only.

- Reference applications

There are two parts in iROS-ONU Application software. The first part contains the sample high-level protocol implementations developed by iROS-ONU such as IGMP snooping, Rapid Spanning Tree Protocol, 802.1x, Rapid Spanning Tree Protocol (RSTP), and also contains high-level functionality that is needed to interface with system vendor applications and external SC messages. It also contains other sample application implementations that are vendor-specific, such as sample vendor-specific OAM extensions, and 4-port switch drivers. The iROS-ONU Application component also contains the System Abstraction Layer(SAL) which is a common set of high-level APIs that can be used to manage various ONU features like switching, QoS, MPCP, encryption, authentication, MDIO/GPIO access, Interface Management etc. The SAL APIs can be used in two cases:

1) It can be directly used by system vendors applications running on the ONU CPU to manage the ONU functions, in place of using SC and SC-OOB and

2) It is used by iROS-ONU Applications for cases where it is needed to communicate with the ONU ASIC. This includes the case where the iROS-ONU Application receives messages from SC and SC-OOB to configure/retrieve ONU parameters, and also iROS features such as IGMP and 802.1x running on the ONU CPU that might need to configure the ONU ASIC. Essentially, the entire iROS application layer now needs to go through the SAL whenever it wants to communicate with the ONU ASIC. Internally, the SAL API uses the AAL layer for configuration/retrieval purposes.

Cortina Systems, Inc. Confidential

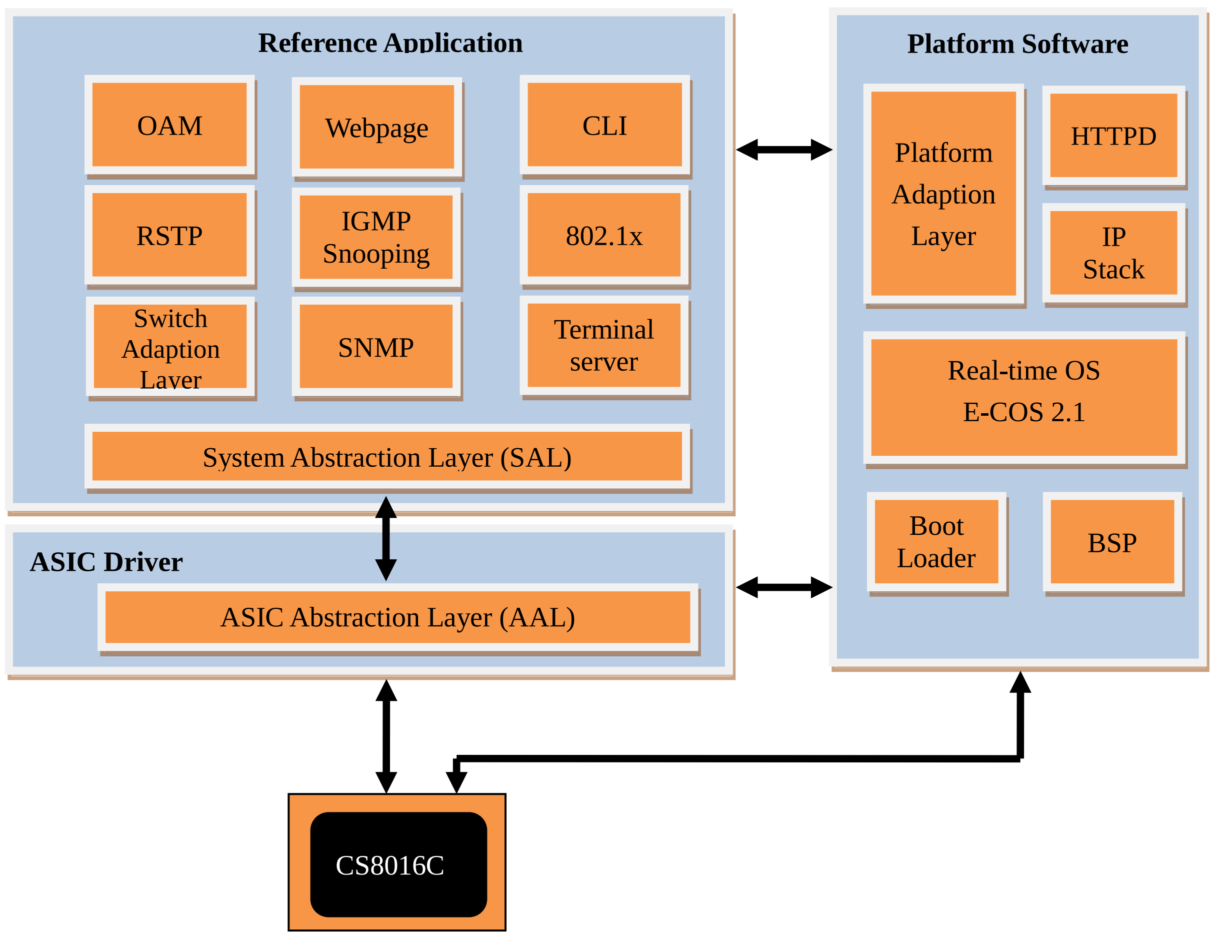


Figure 2 iROS-ONU 3.0 Software Architecture

3.2 1 Port and 4 Ports ONU Packages

Cortina provide 2 software packages in iROS-ONU 3.0:

- 1-Port ONU package
  - Target for Cortina 1-Port reference board
  - Support GE/FE UNI interface
  - Support terminal server for Power Grid application
- 4-Ports ONU package
  - Target for Cortina 4-Ports reference board
  - Support sample driver for external switch
  - Support terminal server for Power Grid application

These 2 packages have similar source tree and the compile/build/Firmware upgrade procedures are the same. Except switch and GPIO configuration, all the information in this document is shared by these 2 packages.

Cortina Systems, Inc. Confidential

3.3 iROS-ONU 3.0 Source Tree Introduction

```
$IROS-ONU-ROOT
|-- ecos                                      ECOS 2.1 source files
|-- iros_onu                                  ONU target images
|-- src                                       All source codes
      |-- cmn                                 Common application and platform codes
      |    |-- apps
      |    |    |-- clause57                  802.3ah OAM
      |    |    |-- event                     Event handler functions
      |    `-- plat                           including  peripheral  driver
      |-- include                             System wide header files
      |-- onu                                 ONU source files
      |    |-- aal                            AAL driver (header only)
      |    |-- apps
      |          |-- dot1x                    802.1x source code
      |          |-- stp                      spanning tree source code
      |          |-- marvell                  Marvell switch driver (4 ports package only)
      |-- sc
      |--tools                                Scripts and utilities
```

- $IROS-ONU-ROOT

  This is the root directory of iROS-ONU 3.0 package.
- eCos RTOS

  iROS-ONU 3.0 is based on eCOS 2.1 , more detail information can be found on eCOS website: http://ecos.sourceware.org/docs-1.2.1/tutorials/ecos-tutorial/ecos-tutorial.html
- ONU source files

  Aal – AAL header files ($IROS-ONU-ROOT/src/onu/aal)

  Apps – application and protocol task source files ($IROS-ONU-ROOT/src/onu/app)
  - bridge.c (bridge configuration API interface logic)
  - config.c (ONU general configuration API interface logic)
  - frame.c (CPU port frame receiving and processing logic)
  - hello.c (CPU port hello protocol handling logic)
  - **main.c (ONU Application task main loop)**
  - message.c (ONU RPC message dispatch logic)
  - oamp.c (glue logic to OAM state machine)
  - port_dot1x.c (glue logic to 802.1x state machine
  - port_mcast.c (glue logic to IGMP state machine)
  - security.c (802.1x configuration API interface logic)
  - tm.c (traffic management configuration API interface logic)
  - **api.c (SAL API interface logic)**

Cortina Systems, Inc. Confidential

- 802.3ah OAM
  All OAM related files are located at: $IROS-ONU-ROOT /src/cmn/apps/clause57
  - oam_client.c (OAM frame processing logic)
  - oam_control.c (OAM state machine logic)
  - oam_core.c (OAM to application glue logic)
  - oam_env.c (OAM event generation logic)
  - oam_mux_par.c (OAM loopback handling logic)
  - oam_timer.c (OAM related timer logic)
- 802.1x
  All 802.1x related files are located at: $IROS-ONU-ROOT/src/onu/apps/dot1x
  - eap.c (EAP frame processing logic)
  - eapmd5.c (MD5 codec logic)
  - eapol.c (EAPOL frame processing logic)
  - statemachine.c (802.1x supplicant state machine logic)
- IGMP Snooping
  All multicast related files are located at: $IROS-ONU-ROOT/src/cmn/apps/mcast
  - mfdb.c (multicast FDB get/add/delete logic)
  - snoop.c (IGMP snooping related logic)
- Switch adaption driver
  For 4-Ports ONU package, the sample switch driver is located at:
  $IROS-ONU-ROOT/src/cmn/apps/marvell
  - onu_marvell_sample_drv.c (High level switch API)
  - onu_marvell_sample_hwcfg.c (includes all basic functions to configure switch register or table)
- Platform software
  Platform related files are located at: $IROS-ONU-ROOT/src/cmn/plat
  - cli.c (serial CLI, debug command handler logic)
  - cpuload.c (cpu load monitoring logic)
  - eeprom.c (EEPROM interface logic)
  - “flash” folder(Flash interface logic)
  - i2c.c (I2C interface logic)
  - ifm.c (Interface manager logic)
  - logger.c (System log logic)
  - mdio.c (MDIO read & write logic)
  - term_server.c (Terminal server logic for Power Grid application)
  - webpage.c (webpage handling logic)
  - **main.c (ONU APP root task logic, all other tasks are spawned by root task)**
- ONU images
  ONU images will be generated at : $IROS-ONU-ROOT/iros_onu
- Tools
  All tools are located at: $IROS-ONU-ROOT/tools/support/tools/bin
  - eeprom_onu_gen (Generate startup configuration file)
  - flash_onu_gen (Assemble loader ,startup_config_data ,ONU app firmware and jffs2 image to a single image)
  - mkfs.jffs2 (Make a JFFS2 filesystem image from an existing directory tree)

3.4 IROS-ONU 3.0 SW Module Customization

Customer might want to customize their software package (ie. add/delete features) before compile and build. In the iROS-ONU 3.0 release package, the iros_config.h contains the module control MACROs which are used for enable/disable the related SW module. There are three sections in iros_config.h.

The 1st section contains the basic modules which are required for minimum systems. It has below format.

```
/***********************************************************************
  * Basic MACRO (Don't edit it)                                         *
  ***********************************************************************/
#define    HAVE_ONU_SPECIAL_PKT_RATE_LIMITER       1
#define    HAVE_ORGSPEC_OAM_HANDLER                  1
...
/* End Basic MACRO ***************************************************/
```

The contents in this section are controlled by Cortina Software team. It not allowed editing by customers.

The second section contains the optional modules which can be customized. Customers are allowed to edit the module control MACROs in this section to get the special propose they want. The format is displayed as below:

```
/**********************************************************************
  *Optional MACRO(Below module can be enable/disable by the MACROS) *
  **********************************************************************/
/* Enable CTC ext OAM support                                          */
#define    HAVE_CTC_OAM                          1
/* Enable CUC ext OAM support                                          */
#define    HAVE_CUC_OAM                          1
...
/*End Optional MACRO *************************************************/
```

Customs can add/remove the MACROs to enable/disable the related modules. Please refer to section 5.1 for all Marcos that iROS-ONU 3.0 supported.

Note: If you want to disable one module, you should remove or comments out the controls MACROs.
For instance, either one of following methods would disable the CTC OAM module:

```
/* #define    HAVE_CTC_OAM                          1 */
or
// #define    HAVE_CTC_OAM                          1
```

Following is a wrong example (to define the MACRO as 0), this would cause compile error:

```
#define HAVE_CTC_OAM                          0
```

Cortina Systems, Inc. Confidential

The 3rd section contains the module definition format check and module dependences relationship check. It also controlled by Cortina software team and not allowed to be edit by customers. The format is displayed as below:

```
/*********************************************************************
  *MACROS depends check. Don't edit it                                      *
  *********************************************************************/
/* Check MACRO define Format */
#if (defined(HAVE_CTC_OAM)              && !HAVE_CTC_OAM)               ||\
(defined(HAVE_ONU_SPECIAL_PKT_RATE_LIMITER)&&!HAVE_ONU_SPECIAL_PKT_RATE_L
IMITER)
        #error HAVE_XXX_XXX MACROs should be define to 1 or comments out, Please fix it.
#endif
/* Check IP Stack depends                                                   */
#ifndef HAVE_ONU_IP_STACK
        #if defined(HAVE_TELNET_CLI)
                #error    IP Stack is required by HAVE_TELNET_CLI.          Please fix it at first
        #elif    defined(HAVE_TERMINAL_SERVER)
                #error    IP Stack is required by HAVE_TERMINAL_SERVER. Please fix it at first
        #elif    defined(HAVE_ONU_SNMP)
                #error    IP Stack is required by HAVE_ONU_SNMP.             Please fix it at first
        #endif
#endif
/*End MACROS depends check ****************************************/
```

3.5 Compile the IROS-ONU 3.0 Package.

The IROS-ONU 3.0 develop environment is base on Fedora 4, and a full package installation is recommended. A compilation toolchain is the set of tools that allows compiling code for your system. It contains arm-elf-gcc, arm-elf-as and arm-elf-ar etc which used to build the arm-arch image.

3.5.1 Install Toolchain

Cortina provide the toolchain tar ball for customer. So customer doesn't need to re-build it on host system. Toolchain tar ball can be downloaded through Cortina Customer support website. After download the tar ball, customer can install it into any directory (need to have read/execution privilege).

Here is an example to install the toolchain at /opt/toolschain:

```
cd /opt
mkdir toolschain
cd toolschain
tar xvzf iros-tools-101507.tgz
```

Cortina Systems, Inc. Confidential

3.5.2 Build Boot loader image

Step1: Untar the Boot loader package.

```
mkdir ~/myproject
cd    ~/myproject
tar xvzf iros-zloader-02.08.01-1254301956.tgz
```

Step2: Set the environment variable before compile and build

```
cd iros-zloader-02.08.01-1254301956
export SWHOME=$PWD
export TOOLSHOME=/opt/toolschain/iros-tools
make
```

Step3: Get the Boot loader image file
The compiled Boot loader image file (*.img) is generated under following directory:
$SWHOME/main_flash/onu_zloader.img

3.5.3 Build ONU image

Step1: Untar the iROS-ONU source package.

```
mkdir ~/myproject
cd    ~/myproject
tar xvzf iros-std-onu-3.10.15.0-Dec-10-2010.tgz
```

Step2: Set the environment variable before compile and build

```
cd iros-std-onu-3.10.15.0-Dec-10-2010
export SWHOME=$PWD
export TOOLSHOME=/opt/toolschain/iros-tools
make
```

Step3: Get the ONU image file
The compiled ONU image file (*.zblob) is generated under following directory:
$SWHOME/iros_onu/ iros_onu_asic_rom_lynxb_ip.zblob

3.6 Images Upgrade

This section describes how to upgrade boot loader and ONU image through UART interface.

Figure 3 illustrate the image upgrade environment.

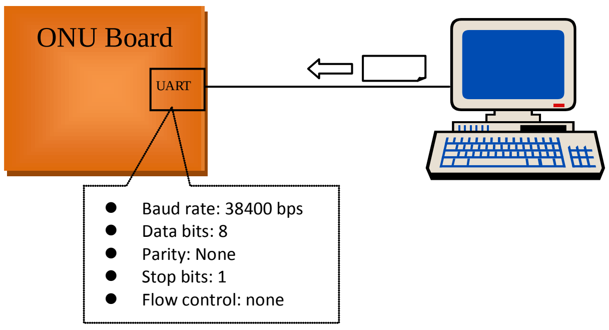


Figure 3 Image upgrade environment

3.6.1 Boot Loader Update

Since boot loader is executed in flash, so it cannot update itself when running. Cortina provide a special loader which can be run in memory to upgrade target boot loader back to flash. Figure 4 illustrate the
So you need to prepare following 2 images before boot loader upgrade.

- upgrade_zloader-*.img

  This is the pre-loader which is available in boot loader package.

- onu_zloader-*.img

  This is the target boot loader that you get after compile.

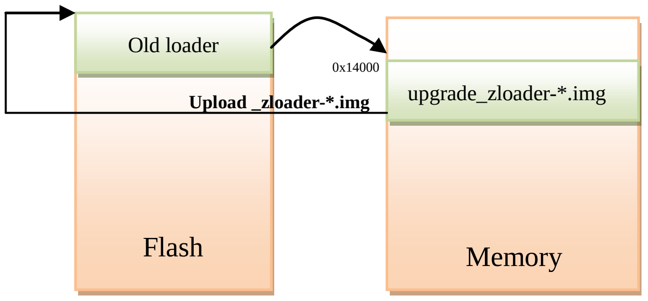


Figure 4 Boot loader update procedure

Here are the steps to update the ONU boot loader:

Step1: Power up the board and press ‘d’ or ‘D’ to enter the loader shell.

Step2: Load upgrade_loader image to memory

iROSBoot>l m 0x14000;

Send “upgrade_zloader-*.img” by Xmodem protocol

Step3: Run upgrade_loader image in memory

iROSBoot>g 0x14000

Step4: Upload ONU boot loader image

iROSBoot>u l

Send boot loader image “onu_zloader-*.img” by Xmodem protocol

Step5. Reset the ONU to run the new loader

Cortina Systems, Inc. Confidential

3.6.2 ONU Image Update

Here are the steps to update the ONU image:
Step1: Power up the board and press ‘d’ or ‘D’ to enter the loader shell.
Step2: Upload ONU image

```
iROSBoot> u a;
Send ONU image *.zblob by Xmodem protocol
```

Step3: Reset ONU to run the new ONU image

3.6.3 Startup Configuration File Update

Default startup configuration file is provided by Cortina. Customer can modify it base on their application. Please refer to section 5.2 for the detail definition of startup configuration. Cortina also provide a tool to let customer generate their own configuration file. The usage of this tool is described in section 5.4.1.
Here are the steps to update startup configuration in flash partition:
Step1: Erase flash blocks to prepare space for startup configuration data.

```
iROSBoot>r f 0x2f010000
```

Step2: Load the configuration data.

```
iROSBoot> l f 0x2f010000
Send the startup configuration file (ex:onu_eeprom.dat) by xmodem protocol.
```

Step3: Verify the configuration data in flash

```
iROSBoot>d m 0x2f010000
```

After the startup configuration file is loaded into the flash and verified, you need to reboot the system to apply the new startup configuration file in flash.

3.6.4 File system Update

File system is used by customer to store their configuration or html resource files. Please refer to section 5.4.2 for JFFS2 image generation.
Here are the steps to upgrade root file system in flash partition
Step1: Check JFFS2 start address in partition table.
The partition table will be displayed on console when the ONU bring up.
Example:

```
Partition Index 0x0 ID 0x77770001 Type 0x1 Start 0x2f000000 Size 0x00010000 BOOT
Partition Index 0x1 ID 0x77770006 Type 0x3 Start 0x2f010000 Size 0x00010000 STARTUP_CONFIG
Partition Index 0x2 ID 0x77770003 Type 0x8 Start 0x2f020000 Size 0x00010000 SUPER_BLOCK
Partition Index 0x3 ID 0x77770002 Type 0x7 Start 0x2f030000 Size 0x00130000 XIP
Partition Index 0x4 ID 0x77770004 Type 0x2 Start 0x2f160000 Size 0x000c0000 BLOB
Partition Index 0x5 ID 0x77770005 Type 0x2 Start 0x2f220000 Size 0x000c0000 BLOB
Partition Index 0x6 ID 0x77770008 Type 0x5 Start 0x2f2e0000 Size 0x00010000 DATA
Partition Index 0x7 ID 0x77770009 Type 0x6 Start 0x2f2f0000 Size 0x00010000 CORE_DUMP
Partition Index 0x8 ID 0x77770007 Type 0x4 Start 0x2f300000 Size 0x00100000 JFFS2
```

In this example, JFFS2 start address is 0x2f300000 and its size is 0x100000(1024KB).
Please note that your file system image must smaller than the size of JFFS2 partition.

Step2: Power up the board and press ‘d’ or ‘D’ to enter the loader shell.

Step3: Erase flash blocks to prepare space for file system.

    iROSBoot>r f <JFFS2 start address>   (Note: the JFFS2 start address is got from step 1.)

Step4: Load the file system.

    iROSBoot> l f <JFFS2 start address>   (Note :the JFFS2 start address is got from step 1.)

    Send the file system image (ex: cortina-home_jffs2.img) by xmodem protocol.

Step5: Reset ONU

    iROSBoot> r o

Step5: Check file system after system bring up

```
ONU> enable
ONU# config terminal
ONU(config)# debug
ONU(config-debug)# legacy
ONU(config-debug)# x fcli
onu_fs>ls      ➔This will list all files in file system.
```

3.7 ONU Flash Partition

3.7.1 Flash Partitions introduction

Cortina define 8 types of partitions in a flash. Here are the descriptions of each type of partition:

1. Boot loader

   The boot loader initializes the board and downloads the operation image (Blob Image).The boot loader reads the 2 hardware strap pins IMAGE_DOWN_0 and IMAGE_DOWNLOAD_1 and determines the boot mode. Two modes are supported:

   • UART

   • FLASH

   The UART boot mode provides a debug shell for various functions including memory/register access, EEPROM dump, ONU status dump, and image load and upgrade. The FLASH boot mode selects a valid blob image and relocates the image to the execution area according to the Loading Start pointer in the blob header.

2. Startup Configuration

   Startup configuration will be loaded into memory when system startup. (Please refer to section 3.7.)

3. Super Block and Partition Table

   Super Block is used to store information and status of the partitions. When firmware upgrade, boot loader will check the partition table in super block and write the image to the right location.

4. Blob0/1

   A blob image is the operation image running on Cortina Systems® EPON SoC. The blob image uses the Cortina Systems® proprietary format: .blob.

   To support image protection and redundancy, there are two areas in the flash to store two independent blob images:

   • Blob Image 0

   • Blob Image 1

   Only one active image will be run, another backup image will be replaced when new image upgrade.

5. Image Execution Area

   The blob Image Execution Area is used for running the operation image if the image is built to run from

Cortina Systems, Inc. Confidential

the flash. When the system boots up, one of blob images will be extracted to this area and execute.

6. User Data

   This partition is used by Cortina to store runtime user configuration.

7. Core Dump

   This partition is used by Cortina to record the system running info when system is crash.
   For example, the register contents, the thread stacks etc.

8. JFFS2 File system

   This partition is used by customer to store their configurations or html files (please also refer to section 4.4 about web page development guide.).

Please note that not all these partitions are needed for every application. Cortina provide different partition definition base on flash size, those are defined in section 3.7.2.

3.7.2 Flash Partitions for different flash size

iROS-ONU 3.0 can support various kinds of flash size (2MB/4MB/8MB/>8MB), but the partition definitions are different in each flash size. The detail definition are displayed as below:

- Flash partition for 2MB flash

Address	Size (bytes)	Usage
0x2f000000	0x10000(64KB)	Boot code
0x2f010000	0x10000(64KB)	Startup Configuration
0x2f020000	0x10000(64KB)	Super Block and Partition Table
0x2f030000	0xb0000(704KB)	Execution Area
0x2f030000	0x90000(576KB)	Blob0
0x2f170000	0x90000(576KB)	Blob1

- Flash partition for 4MB flash

Address	Size (bytes)	Usage
0x2f000000	0x10000(64KB)	Boot code
0x2f010000	0x10000(64KB)	Startup Configuration
0x2f020000	0x10000(64KB)	Super Block and Partition Table
0x2f030000	0x130000(1216KB)	Execution Area
0x2f160000	0xc0000(768KB)	Blob0
0x2f220000	0xc0000(768KB)	Blob1
0x2f2e0000	0x10000(64KB)	User Data
0x2f2f0000	0x10000(64KB)	CORE Dump
0x2f300000	0x100000(1024KB)	JFFS2 File System

Cortina Systems, Inc. Confidential

- Flash partition for 8MB flash or larger

Address	Size (bytes)	Usage
0x2f000000	0x10000(64KB)	Boot code
0x2f010000	0x10000(64KB)	Startup Configuration
0x2f020000	0x10000(64KB)	Super Block and Partition Table
0x2f030000	0x260000(2432KB)	Execution Area
0x2f290000	0x180000(1536KB)	Blob0
0x2f410000	0x180000(1536KB)	Blob1
0x2f590000	0x10000(64KB)	User Data
0x2f5a0000	0x10000(64KB)	Core Dump
0x2f70_0000	0x100000(1024KB)	JFFS2 file system

3.8 ONU Startup Sequences

Figure 5 illustrate the ONU initialization sequences in boot loader and eCos. A startup confile file will be loaded into memory during system initialization. The startup config file can be stored in EEPROM or flash. It is up to customer whether they need an EEPROM in their applications. iROS-ONU init process will check EEPROM first then Flash. If there is no valid startup configuration file in both EEPROM and Flash. A default one will be generated by software.

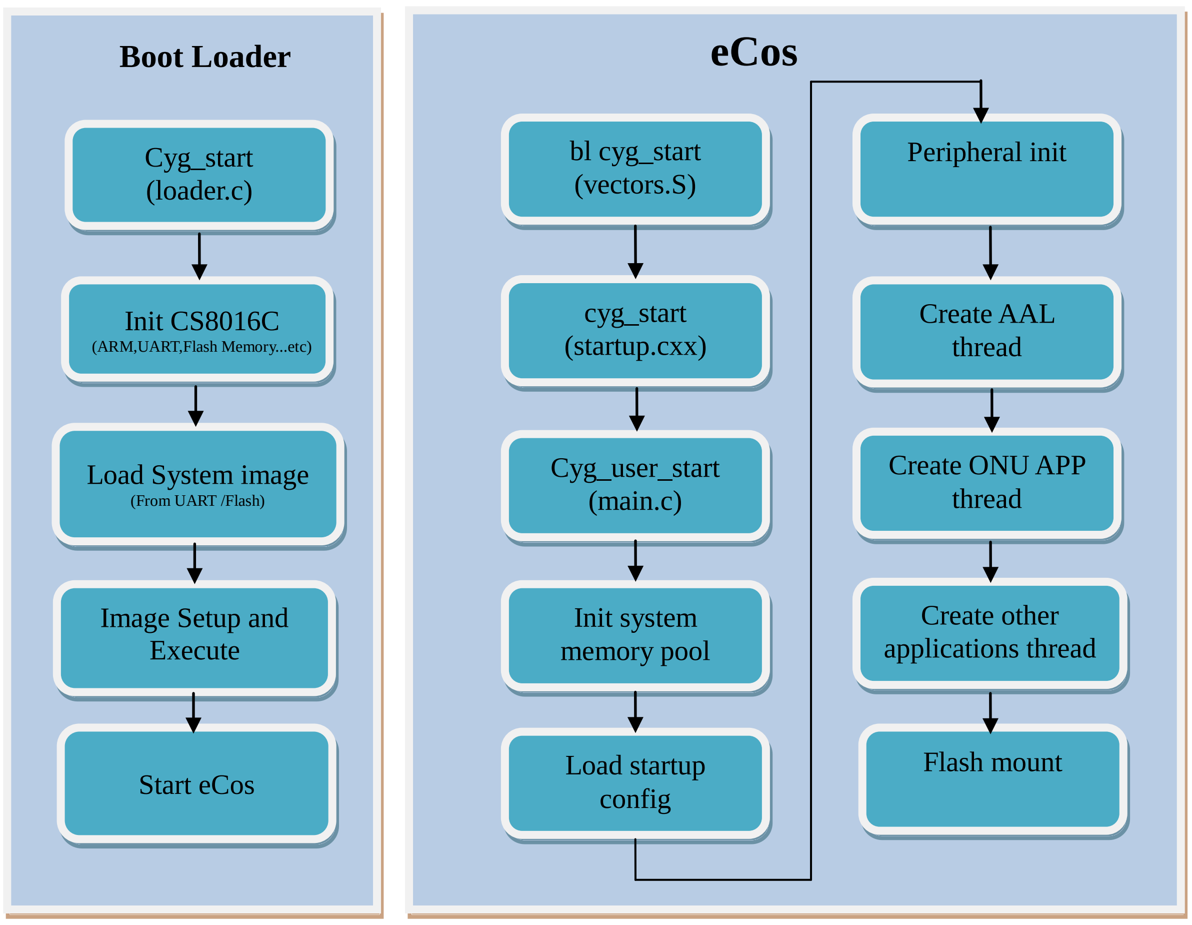


Figure 5 ONU Initialization Sequences

ONU APP is the main task to handle MPCP & OAM registration.

Figure 6 illustrate how ONU APP handles the MPCP and OAM discovery.

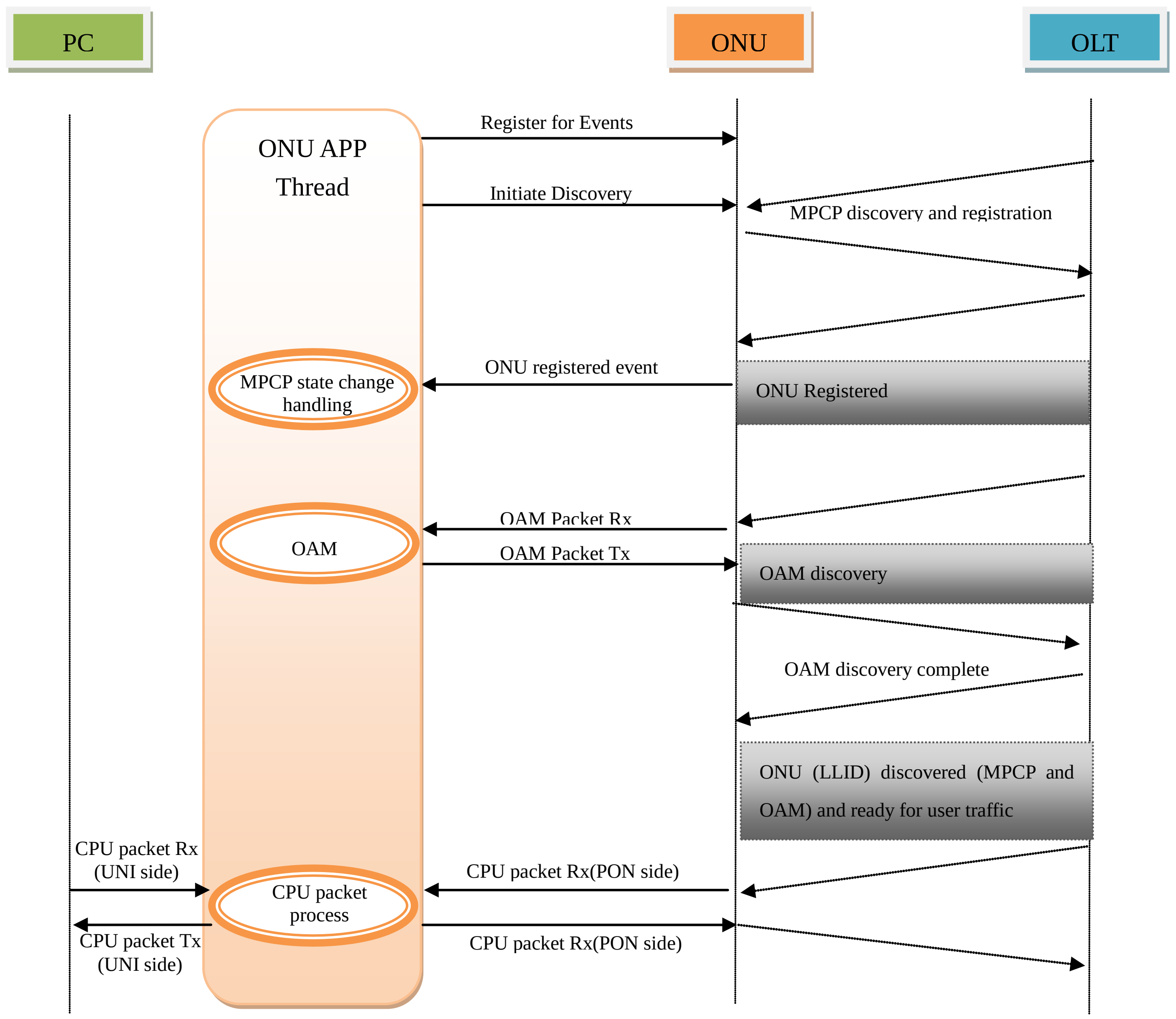


Figure 6 MPCP and OAM Discovery sequences

Cortina Systems, Inc. Confidential

3.9 Diagnostic Console Commands

There are some diagnostics console commands in iROS-ONU 3.0 package. By default, these commands are not available when user login to console. User need to enter the debug mode first. This section describes how customer gets into debug mode and those common commands for debugging purpose.

- Connect console

  Please use following parameters in your terminal client software.

  Baud rate: 38400 bps
  Data bits: 8
  Parity: None
  Stop bits: 1
  Flow control: none

- Enter debug mode

```
ONU> enable
ONU# config terminal
ONU(config)# debug
ONU(config-debug)# legacy
  go to legacy CLI menu...

cmds:
   h                                   - show help
   v                                   - show version and iros information
   v module test_number    - asic verification
   v aal                             - aal unit test
   d m        addr len          - dump memory
   d e        addr len          - dump eeprom
   d p        pool_id [0|1] - dump memory pool info. flag 0 for all blocks. 1 for corrupted blocks only
   d c                               - dump cpuload info.
   b                                   - dump/set by bytes
   w                                   - dump/set by words
   s m        addr val           - set memory to value
   s dis_mode                    - set mpcp discovery mode
   s crc_mode                    - set mpcp crc8 mode
   g addr                           - start execution at addr
   p cpu                             - show cpu port
   p stats                         - show stats
   p thread info               - show thread info
   p thread count              - show thread count
   p mdio device addr       - show mdio
   c stats                         - clear stats
   exit                               - exit from current mode
   r                                    - reset onu
   t arp      port                - transmit arp frame
   t g                                - transmit oam dying gasp
   t oam      port              - transimit oam packet
   k 0|1                             - software watchdog timer enable 1; disable 0
   m [level [mod]]            - set logging level, default 7 for critical
   l m [0|1|2|3]               - set logging output mode, 0-disable, 1-mem, 2(default)-mem/console,
                                       3-mem/console/flash
   l c [f|m]                       - clear current log in flash or memory
   l s [f|m]                       - display current log in flash or memory
   ts help                         - show terminal server configure command
```

- Exit debug mode

```
onu->quit
ONU(config-debug)# exit
```

- Control Laser on / off

```
ONU(config)# laser   set   1      ;Force optics turn on laser
ONU(config)# laser   set   0      ;Control optics back to normal mode
```

- Query CPU Utilization

```
onu->d   c
cpu load (last 0.10 sec): 13% busy        ;Current CPU utilization
cpu load (last 1.00 sec):    8% busy      ;Average CPU utilization in last second
cpu load (last 10.0 sec): 12% busy        ; Average CPU utilization in last 10 seconds
```

- Query Memory pool infomration

```
onu->d   p   0                          ; Query memory pool 0 information
onu->d   p   1                          ;Query memory pool 1 information
malloc info:
  mallinfo.arean 262144
  mallinfo.ordblks 30
  mallinfo.smblks 0
  mallinfo.smblks 0
  mallinfo.hblks 0
  mallinfo.hblkhd 0
  mallinfo.usmblks 0
  mallinfo.fsmblks 0
  mallinfo.uordblks 67824
  mallinfo.keepcost 0
  mallinfo.maxfree 192668

iros memory pool dump
           id: 0
      tracing: 1
   fail abort: 0
  overflow ok: 1
     max size: 32768
     cur size: 140
    peak size: 4720
   malloc cnt: 2315
     free cnt: 2312
   size ranges:
        size      2 to      4: cur      0 peak      0
        size      4 to      8: cur      0 peak      0
        size      8 to     16: cur      1 peak      1
        size     16 to     32: cur      1 peak      1
        size     32 to     64: cur      0 peak      0
        size     64 to    128: cur      1 peak      8
        size    128 to    256: cur      0 peak      0
        size    256 to    512: cur      0 peak      0
        size    512
```

- Query Task status

```
onu->p t i                        ; Query task status

aal_count  :       447508

httpd_count:           95
ID: 0002 | name:                   main | set_pri: 10 | cur_pri: 10 | state:   4
ID: 0003 | name: Network alarm support | set_pri: 14 | cur_pri: 14 | state:   1
ID: 0004 | name:       Network support | set_pri: 15 | cur_pri: 15 | state:   1
ID: 0005 | name:        ONU AAL Thread | set_pri: 11 | cur_pri: 11 | state:   0
ID: 0006 | name:       ONU APPL Thread | set_pri: 12 | cur_pri: 12 | state:   0
ID: 0007 | name:      ONU SHELL Thread | set_pri:  5 | cur_pri:  5 | state:   0
ID: 0008 | name:    ONU TELNETD Thread | set_pri: 19 | cur_pri: 19 | state:   1
ID: 0009 | name:         Logger Thread | set_pri: 18 | cur_pri: 18 | state:   1
ID: 000a | name:              ATHTTPD | set_pri: 16 | cur_pri: 16 | state:   1
ID: 0001 | name:           Idle Thread | set_pri: 31 | cur_pri: 31 | state:   0
```

- Enable debug message

```
onu->m 0 1
```

- Disable debug message

```
onu->m 7 1
```

- Dump OAM message

```
onu->p p      ;Enable
...
onu->p p      ;Disable
```

- Send Dying gasp message

```
onu->t g
```

- Display ONU device information

```
onu->p    s
System Up Time(DD:HH:MM:SS)      : 00:01:30:41
onu is                                     : Registered
pon port link is                          : Up
ge port link is                           : Up
onu_mac_addr                               : 00:13:25:ff:ff:81
olt_mac_addr                              : 00:00:00:00:00:00
llid_port_id                             : 00000000
sync time                                  : 0
arm irq status/raw/mask              : 0x00000000/0x00000108/0x02ec4880
onu glb irq source/enable          : 0x00003011/0x00000106
laser ctrl polarity                    : 0
laser on/laser off                      : 64/64
RX CRC mode                                  : 1
TX CRC mode                                  : 1
FEC mode                                     : 0
ge_l2_ctrl/pon_l2_ctrl               : 0xa8a62200/0xa0a68340
ge rx byte/pkt cnt                       : 1370808/366
ge tx byte/pkt cnt                       : 0/0
pon rx byte/pkt cnt                     : 0/0
pon tx byte cnt                            : 00
MPCP Discover                              : 0
MPCP Register req                        : 0
MPCP Register                              : 0
MPCP Gate for Ack                         : 0
MPCP Register Ack                         : 0
MPCP Gate/Report                           : 0/0
MPCP status                                 : 0x20000000
MPCP register bc_dreg/uc_dreg : 0/0
frm_tx_ok                                   : 8
frm_tx_olt                                 : 0
frm_tx_imstar                              : 0
frm_tx_err                                 : 2299
frm_rx_err                                 : 0
frm_rx_onu                                  : 0
frm_rx_oam                                  : 0
frm_rx_eapol                               : 0
frm_rx_igmp                                 : 0
frm_rx_imstar                              : 0
frm_rx_mpcp                                 : 0
frm_rx_kt_lpbk                             : 106
frm_rx_mgmt                                  : 0
frm_rx_dropped                              : 0
frm_rx_flushed                             : 0
frm_rx_onu_mii                              : 0
frm_rx_eapol_mii                           : 0
frm_rx_imstar_mii                         : 0
frm_rx_dropped_mii                         : 0
frm_rx_ip                                   : 0
frm_rx_arp                                  : 2169
frm_rx_dhcp                                  : 0
frm_rx_stp                                  : 0
onu_aal_count                              : 551518
aal_poll_isr                             : 0
onu_cpu_isr                                 : 2275
onu_mii_isr                                : 0
epon_lnkc_isr                             : 0
authenticated                            : 0
802.1x enabled                            : 0
MC enabled                                  : 0
802.1x tunnel                             : 0
IOP Vendor Id                              : 255
Chip type                                   : 8016.C0
ONU Config Code                             : 254
ONU Ctrl Vid                                : 0
OAM version D2                              : 0
CTC OAM Bypass                               : 0
CTC FEC mode                                 : 0
CTC CRC mode                                 : 0
CTC Reg BackOff                             : 60
Ukn MC Drop                                  : 1
Query Key                                   : 0
Auto reset disable                     : 1
Switch Port Feature (number of ports): 4
CtcUpgrade Mode                             : 0
```

Cortina Systems, Inc. Confidential

- Display terminal server commands

```
onu->ts help
ts enable <uart> <baudrate> <databits> <parity> <proto> <port> <timeout> <min> <max> <resp>
                        --<uart>: 2 for uart2, 3 for uart3...5 for uart5
                        --<baudrate>: 1~312500
                        --<databits>: 5/6/7/8
                        --<parity>: 0-None 1-Odd 2-Even
                        --<proto>: 17 for UDP, 6 for TCP
                        --<port>: 1~65535
                        --<timeout>: 0~65535s for remote timeout, 0-never timeout
                        --<min>: 40~<max> for minimal payload length
                        --<max>: 512~1400 for maximum playload length
                        --<resp>: 25~1000ms for maximum reponse time
                        ---Used to create a terminal server
      ts disable <uart>    --disable terminal server
      ts  dbg                  --enable/disable terminal server debug info
      ts reset <uart>       --reset terminal server wait for new connect
      ts save <uart>         --save configuration into flash
      ts show <uart>         --show terminal server status
      ts rd <uart> <reg> --read UART register
                                  --<uart>: 2 for uart2, 3 for uart3...5 for uart5
                                  --<reg>: 1~15
      ts wt <uart> <reg> <val> --write UART register
                                           --<uart>: 2 for uart2, 3 for uart3...5 for uart5
                                           --<reg>: 1~15
                                           --<val>: 0~0xff
```

- Reboot ONU

```
onu->v   a   1
```

3.10 Telnet

If HAVE_TELNET_CLI is defined in iros_config.h, user can telnet to ONU for remote management. User can telnet to ONU through PON port or UNI/management ports. The default PON port IP address is 192.168.2.1, and the default IP address for UNI or management port is 192.168.1.1. After login with default username and password (admin/admin), user would see the same shell as console. So user can use all the diagnostic console commands which are listed in section 3.9.

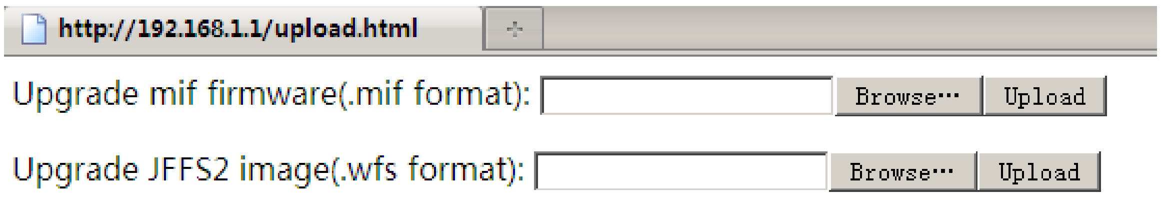
```
>telnet 192.168.1.1
iROS ONU CLI
UserName:admin
PassWord:*****
ONU(config)# help
Commands available:
   help                  Show available commands
   history               Show a list of previously run commands
   quit                  Disconnect
   logout                Disconnect
   exit                  Exit from current mode
   show                  Show running system information
   interface             Select an interface to configure
   onu-mac               ONU PON mac
   debug                 Enter debug mode
   arp                   ARP configuration
   ping                  Ping the specified host
   statistics            Show system statistics information
ONU(config)#
```

3.11 Web

If HAVE_ONU_IP_STACK is defined in iros_config.h, a web server will be launched after system startup. Cortina provide one firmware upgrade page in iROS-ONU package. This demo page is an example for those who want to develop their own web page. Please refer to section 4.4 for the web development guide. User can access the web through PON port or UNI/management ports. The default PON port IP address is 192.168.2.1, and the default IP address for UNI or management port is 192.168.1.1. User can login with default username and password (admin/admin)

Figure 7 illustrate the firmware upload home page. User can upload firmware and file system in this page. Please note that the firmware and file system images are not generated from regular build process. Customer need to use Cortina proprietary utility to make these images. Please refer to section 5.4.2 for steps to create JFFS2 file system and section 5.4.4 for the step to make MIF image.

Figure 7 Firmware upload web page


http://192.168.1.1/upload.html
Upgrade mif firmware(.mif format):
Browse…
Upload
Upgrade JFFS2 image(.wfs format):
Browse…
Upload

4. Development Guide

4.1 Add a C file or Compile options in Makefile

iROS-ONU package contains some binary file (.a) and part of source code. It is possible for customer to add their own source code.If Customer want to add a C file named src/customer.c which contains some private functions, then customer need to edit the librels/Makefile and add the file at the end of CFILES list.
For example:

```
CFILES= \
$(SWHOME)/src/onu/apps/sync_api.c \
…
$(SWHOME)/src/cmn/plat/flash/iros_flash_oper.c \
$(SWHOME)/src/customer.c
```

Then the file src/customer.c will be compiled, lib into librel.a, link to the final binary file.

Also customer may like to modify the compile option, for example, the src/customer.c has use MACRO which name is CUSTEM_DEFINE.
Edit the librels/Makefile's MOD_CFLAGS variable and add the MACRO
For example:

```
MOD_CFLAGS=   -Werror   -mcpu=arm7tdmi      -g   -nostdlib   -c      -fno-inline
-mthumb-interwork   -mthumb  -DAPPS_ONU  -O3  -DCTC_EXT=1  -DONU_8021x=1
-DOOB_MGMT=1 -include $(IROS_CONFIG_H) -DCUSTEM_DEFINE
```

4.2 Support Other External Switch

In 4-Port package, Cortina provide a set of switch operation APIs which is used by upper application, so it is possible to for customer to replace any type of external switch. The switch adaption APIs are defined as below (Refer to $IROS-ONU-ROOT\src\onu\apps\onu_switch_if.h):

```
typedef struct
{
        epon_return_code_t (*reset)();
        /* init */
        epon_return_code_t (*init)();

        /* port management */
        epon_return_code_t (*get_port_oper_status)(epon_port_id_t, epon_port_oper_status_t *);
        epon_return_code_t (*get_port_admin_status)(epon_port_id_t, epon_port_admin_status_t *);
        epon_return_code_t (*set_port_admin_status)(epon_port_id_t, epon_port_admin_status_t);
        epon_return_code_t (*get_port_speed)(epon_port_id_t, epon_ether_speed_t *);
        epon_return_code_t (*set_port_speed)(epon_port_id_t, epon_ether_speed_t);
        epon_return_code_t (*get_port_duplex_mode)(epon_port_id_t, epon_ether_duplex_t *);
        epon_return_code_t (*set_port_duplex_mode)(epon_port_id_t, epon_ether_duplex_t );
        epon_return_code_t (*reset_port_auto_neg)(epon_port_id_t);
        epon_return_code_t (*get_port_auto_neg)(epon_port_id_t, epon_boolean_t *);
        epon_return_code_t (*set_port_auto_neg)(epon_port_id_t, epon_boolean_t);
        epon_return_code_t (*get_port_eth_pauth)(epon_port_id_t, epon_boolean_t *);
        epon_return_code_t (*set_port_eth_pauth)(epon_port_id_t, epon_boolean_t);
        epon_return_code_t (*set_port_ingress_policy)(epon_port_id_t,    epon_sw_port_policy_struct_t *);
        epon_return_code_t (*get_port_ingress_policy)(epon_port_id_t, epon_sw_port_policy_struct_t *);
        epon_return_code_t (*set_port_egress_rate_limit)(epon_port_id_t, epon_sw_port_policy_struct_t *);
        epon_return_code_t (*get_port_egress_rate_limit)(epon_port_id_t, epon_sw_port_policy_struct_t *);
        epon_return_code_t (*set_port_loopback_mode)(epon_port_id_t ,    epon_port_loopback_mode_t );
        epon_return_code_t (*get_port_loopback_mode)(epon_port_id_t ,    epon_port_loopback_mode_t* );
        epon_return_code_t (*set_port_capturepacket_ethertype)(epon_port_id_t , epon_boolean_t , epon_uint16_t    );
        epon_return_code_t (*get_port_capturepacket_ethertype)(epon_port_id_t , epon_boolean_t*, epon_uint16_t* );
        epon_return_code_t (*set_port_egressmode)(epon_port_id_t, epon_sw_port_egressmode_t );
        epon_return_code_t (*get_port_egressmode)(epon_port_id_t, epon_sw_port_egressmode_t *);
        epon_return_code_t (*set_port_unknown_mc_filter)(epon_port_id_t ,epon_boolean_t );
        epon_return_code_t (*get_port_unknown_mc_filter)(epon_port_id_t ,epon_boolean_t *);
        epon_return_code_t (*set_port_unknown_sa_filter)(epon_port_id_t ,epon_boolean_t );
        epon_return_code_t (*get_port_unknown_sa_filter)(epon_port_id_t ,epon_boolean_t *);
        epon_return_code_t (*set_port_igmp_snooping_status)(epon_port_id_t, epon_boolean_t);
        epon_return_code_t (*get_port_igmp_snooping_status)(epon_port_id_t,    epon_boolean_t *);
        epon_return_code_t (*set_port_learning_status)(epon_port_id_t, epon_boolean_t);
        epon_return_code_t (*get_port_learning_status)(epon_port_id_t,    epon_boolean_t *);
        epon_return_code_t (*clear_port_fdb_entry)( epon_port_id_t , epon_fdb_clear_option_t );
        epon_return_code_t (*set_port_max_host)(epon_port_id_t,epon_uint16_t );
        epon_return_code_t (*get_port_max_host)(epon_port_id_t , epon_uint16_t *);
        epon_return_code_t (*get_port_statistics)(epon_uint8_t port, epon_sw_stats_counter_t *stats);
        epon_return_code_t (*clr_port_statistics)(epon_uint8_t port);
        epon_return_code_t (*set_port_isolation)(epon_port_id_t, epon_boolean_t );
        epon_return_code_t (*get_port_isolation)(epon_port_id_t, epon_boolean_t *);
        epon_return_code_t (*set_uplinkport_isolation)(epon_port_id_t, epon_boolean_t);
        epon_return_code_t (*set_port_stormctrl)( epon_port_id_t, epon_sw_port_inratelimit_mode_t,    epon_uint32_t);
        epon_return_code_t (*get_port_stormctrl)(epon_port_id_t, epon_sw_port_inratelimit_mode_t *, epon_uint32_t    *);
        epon_return_code_t    (*set_port_dot1p_pri_map)(epon_port_id_t , epon_uint8_t , epon_uint8_t );
        epon_return_code_t    (*get_port_dot1p_pri_map)(epon_port_id_t , epon_uint8_t , epon_uint8_t *);
        /* fdb */
        epon_return_code_t (*get_fdb_aging_time)(epon_uint32_t *);
        epon_return_code_t (*set_fdb_aging_time)(epon_uint32_t);
        epon_return_code_t (*get_fdb_learn_status)(epon_boolean_t *);
        epon_return_code_t (*set_fdb_learn_status)(epon_boolean_t);
        epon_return_code_t (*get_fdb_learn_mode)(epon_vlan_learning_t *);
        epon_return_code_t (*set_fdb_learn_mode)(epon_vlan_learning_t);
```

```
    epon_return_code_t (*get_fdb_unkown_mc_filter_status)(epon_unknown_mcast_filter_t *);
    epon_return_code_t (*set_fdb_unknown_mc_fileter_status)(epon_unknown_mcast_filter_t);
    epon_return_code_t (*get_fdb_entry)(epon_uint32_t, epon_sw_fdb_entry_t *, epon_boolean_t *);
    epon_return_code_t (*get_next_fdb_entry)(epon_uint32_t, epon_sw_fdb_entry_t *, epon_boolean_t *);
    epon_return_code_t (*search_fdb_entry)(epon_sw_fdb_entry_t *, epon_boolean_t *);
    epon_return_code_t (*add_fdb_entry)(epon_sw_fdb_entry_t *);
    epon_return_code_t (*del_fdb_entry)(epon_sw_fdb_entry_t *);
    epon_return_code_t (*clear_fdb_entry)(epon_fdb_clear_option_t);

    /* vlan */
    epon_return_code_t (*set_port_pvid)(epon_port_id_t, epon_vlan_id_t);
    epon_return_code_t (*get_port_pvid)(epon_port_id_t, epon_vlan_id_t *);
    epon_return_code_t (*set_ingress_tagged_frame_filter)(epon_port_id_t, epon_boolean_t);
    epon_return_code_t (*get_ingress_tagged_frame_filter)(epon_port_id_t, epon_boolean_t *);
    epon_return_code_t (*set_ingress_untagged_frame_filter)(epon_port_id_t, epon_boolean_t);
    epon_return_code_t (*get_ingress_untagged_frame_filter)(epon_port_id_t, epon_boolean_t *);
    epon_return_code_t (*set_vlan_mode)(epon_port_id_t, epon_sw_port_vlanmode_t);
    epon_return_code_t (*get_vlan_mode)(epon_port_id_t, epon_sw_port_vlanmode_t *);
    epon_return_code_t (*get_next_vlan_entry)(epon_uint32_t, epon_sw_vlan_config_t *, epon_boolean_t *);
    epon_return_code_t (*search_vlan_entry)(epon_sw_vlan_config_t *, epon_boolean_t *);
    epon_return_code_t (*add_vlan_entry)(epon_sw_vlan_config_t *);
    epon_return_code_t (*del_vlan_entry)(epon_sw_vlan_config_t *);
    epon_return_code_t (*clear_vlan_entry)();

    /* classification */
    epon_return_code_t (*add_classification_da_mac)(epon_uint8_t, epon_uint8_t, epon_uint8_t, epon_uint8_t *);
    epon_return_code_t (*del_classification_da_mac)(epon_uint8_t, epon_uint8_t *);
    epon_return_code_t (*add_classification_sa_mac)(epon_uint8_t, epon_uint8_t, epon_uint8_t, epon_uint8_t *);
    epon_return_code_t (*del_classification_sa_mac)(epon_uint8_t, epon_uint8_t *);
    epon_return_code_t (*add_classification_vid)(epon_uint8_t, epon_uint8_t, epon_uint8_t, epon_uint8_t *);
    epon_return_code_t (*del_classification_vid)(epon_uint8_t, epon_uint8_t *);
    epon_return_code_t (*add_classification_vlan_pri)(epon_uint8_t, epon_uint8_t, epon_uint8_t, epon_uint8_t *);
    epon_return_code_t (*del_classification_vlan_pri)(epon_uint8_t, epon_uint8_t *);
    epon_return_code_t (*add_classification_ip_pri)(epon_uint8_t, epon_uint8_t, epon_uint8_t, epon_uint8_t *);
    epon_return_code_t (*del_classification_ip_pri)(epon_uint8_t, epon_uint8_t *);
    epon_return_code_t (*set_classification_reg)(epon_uint8_t, epon_uint8_t);
    epon_return_code_t (*del_classification_reg)(epon_uint8_t, epon_uint8_t);

    /* stp */
    epon_return_code_t (*set_stp_status)(epon_boolean_t);
    epon_return_code_t (*get_stp_status)(epon_boolean_t*);
    epon_return_code_t (*set_port_stp_state)(epon_port_id_t, epon_port_stp_state_t);
    epon_return_code_t (*get_port_stp_state)(epon_port_id_t, epon_port_stp_state_t *);

    /* igmp snooping */
    epon_return_code_t (*set_igmp_snooping_status)(epon_boolean_t);
    epon_return_code_t (*get_igmp_snooping_status)(epon_boolean_t *);

    /*mirror*/
    epon_return_code_t (*set_arpmirror_status)( epon_boolean_t);
    epon_return_code_t (*set_port_mirrormode)( epon_port_id_t, epon_sw_port_mirror_mode_t);
    epon_return_code_t (*get_port_mirrormode)(epon_port_id_t, epon_sw_port_mirror_mode_t *);
    epon_return_code_t (*set_monitordst)( epon_port_id_t , epon_port_id_t );
    epon_return_code_t (*get_monitordst)(epon_port_id_t *, epon_port_id_t *);

    /*jumbo frame control*/
    epon_return_code_t (*set_jumboframe_status)( epon_boolean_t );
    epon_return_code_t (*get_jumboframe_status)( epon_boolean_t *);

   /* tx frame */
    epon_return_code_t (*tx_frame)(epon_port_id_t, epon_uint8_t *, epon_uint32_t);

    /* frame handle registration */
    epon_return_code_t (*register_frame_handle)(epon_special_frame_handler_t);
```

```
        /**/
        epon_return_code_t (*set_qos_remark)(epon_port_id_t, epon_uint8_t);
        epon_return_code_t (*get_qos_remark)(epon_port_id_t, epon_uint8_t *);
    } onu_sw_op_set_t;
```

In iROS-ONU 3.0 package, you can find an example driver for Marvell 6045/6097 switch in following directory:

$IROS-ONU-ROOT\src\onu\apps\marvell.

4.3 GPIO Development Guide

iROS-ONU 3.0 program GPIO pins base on Cortina CS8016C evaluation board. Please refer to section 5.3 for the GPIO usage in different boards. It is up to customer to modify the GPIO settings base on different applications. This section describes GPIO programming guideline.

4.3.1 Access GPIO

Generally, programmer would configure GPIO direction (input or output). Following API is used to set GPIO direction:

```
epon_return_code_t epon_onu_GPIO_write_direction(
            EPON_IN epon_uint32_t gpio_num,
            EPON_IN epon_uint8_t status);
Note: Status stands for GPIO direction, 0 is output and 1 is input.
```

After you set GPIO direction, you can invoke read/write function to access the related GPIO interface

```
epon_return_code_t epon_onu_GPIO_read_status(
                  EPON_IN epon_uint32_t gpio_num,
                  EPON_OUT epon_uint8_t *status);
epon_return_code_t epon_onu_GPIO_write_status(
                  EPON_IN epon_uint32_t gpio_num,
                  EPON_IN  epon_uint8_t     status);
```

Here is an example:

```
epon_onu_GPIO_write_direction(5, 0);     /* set GPIO 5 to output */
epon_onu_GPIO_write_status(5, 1);     /* write 1 to GPIO 5 */
```

4.3.2 Polling Mode and Interrupt Mode

Users can configure GPIO interface to input mode so as to detect the device’s status. For this case, GPIO can be configured as polling mode or interrupt mode.

- Polling Mode

iROS-ONU launch a 100 ms timer to poll the GPIO status. GPIO 10 is used to control to enable/disable GPIO polling routine. When system startup, it will check the status of GPIO 10, and if GPIO 10 pulls high, GPIO polling routine will be executed.

A hook function is introduced to handle GPIO events,

```
void onu_gpio_process();
```

This function is located in onu_gpio.c and is an empty function. Users can implement it and customize their own GPIO service routine. This hook function runs in eCOS alarm context, so some time-consuming operations will not be permitted to execute in this function, and also some other operations, like semaphore, message box, which is not fit in alarm context, are not permitted.

There is also a sample implementation for onu_gpio_process() in onu_gpio.c, it use GPIO 5 to control laser always on mode. It will be built into image if you define macro HAVE_ONU_GPIO_SAMPLE.

- Interrupt Mode

IROS-ONU also provides the interrupt mode to check GPIO status. When IROS-ONU receives GPIO interrupt, it will send event to APP thread, and users can install their event handler to handle this event. There are 2 APIs used to enable/disable GPIO interrupt :

```
epon_return_code_t    epon_onu_GPIO_event_enable(
        EPON_IN epon_uint32_t gpio_num,
EPON_IN epon_gpio_event_handler handle);
epon_return_code_t    epon_onu_GPIO_event_disable(
        EPON_IN epon_uint32_t gpio_num) ;
```

Users can invoke epon_onu_GPIO_event_enable() to enable GPIO interrupt process and install their own GPIO event handler. And epon_onu_GPIO_event_disable() is used to disable related GPIO interrupt.

IROS set GPIO interrupt to falling-edge trigger mode.

4.3.3 Reserved GPIO pin

There are some GPIO pins reserved for special usage, and they will not be permitted to use by customers.

GPIO 0 is used for dying gasp. When dying gasp event happens, it will invoke the related ISR. GPIO ISR will capture this change and send out OAM notification message, and SC will receive this event. In startup configuration file there is a field (offset 0x55, bit 5) to configure whether rising edge trigger or falling edge trigger (0 - rising edge trigger, 1 - falling edge trigger)

GPIO 10 is used to determine to enable GPIO polling routine. When it is detected to high at system initialization phase, the GPIO polling routine will be activated.

4.4 Web development guide

iROS-ONU 3.0 use ATHTTP web server which is available in standard ECOS package. This server is specifically aimed at the remote control and monitoring requirements of embedded applications. It provides the following features:

- GET, POST and HEAD Methods
- File system Access
- Callbacks to C functions
- MIME type support
- CGI mechanism through the OBJLOADER package or through a simple tcl interpreter
- Basic and Digest (MD5) Authentication
- Directory Listing
- Extendable Internal Resources

You can also get more information from following URL:

http://www.ecoscentric.com/ecospro/doc.cgi/html/ref/net-athttpd.html

4.4.1 ATHTTP Macros

The release package provides a demo for webpage. The main file for webpage is src/cmn/plat/webpage.c. ATHPPT server provides some useful macros for web developer to handle web request. Here are some important macros:

1. CYG_HTTPD_HANDLER_TABLE_ENTRY (__name, __pattern, __arg)
   - The parameter__name argument is a variable name for the table entry since C does not allow us to define anonymous data structures. This name should be chosen so that it is unique and does not pollute the name space.
   - The __pattern argument is the match pattern. It is a string with the extension url that will be appended to the default directory. A request filename matches an entry in the table if either it exactly matches the pattern string, or if the pattern ends in an asterisk, and it matches everything up to that point.

     For example, the pattern "/index_files/index.htm" will only match that exact filename, but the pattern "/index_files/index.htm*"; will match
     "/index_files/index.htm",
     "/index_files/index.html"; and any other filename starting with"/index_files/index.htm";
     When a pattern is matched, the hander function is called.
   - The __arg argument is the user defined Handler function. The handler is entirely responsible for generating the response to the client, both HTTP header and content. If the handler decides that it does not want to generate a response it can return false, in which case the table scan is resumed for another match. If no match is found, or no handler returns true, then a default response page is generated indicating that the requested page cannot be found. Finally, the server thread closes the connection to the client and loops back to accept a new connection.

   This macro allows the association of particular URLs to C language callback functions. eCos tables are used to define the association between a URL and its corresponding callback.

2. CYG_HTTPD_FVAR_TABLE_ENTRY(__name, __fvar, __val, __len)
   - The parameter __name argument is a variable name for the table entry since C does not allow us to define anonymous data structures. This name should be chosen so that it is unique and does not pollute the name space.
   - The __fvar argument is is the key word from web which used by Handler function .
   - The __val argument is the C var to store the key.
   - The __len argument is the length for the key

The ATHTTP server will automatically try to parse form variables when a form is submitted in the following cases:

- In a GET request, when the URL is followed by a question mark sign
- In a POST request, when the the 'Content-Type' header line is set to 'application/x-www-form-urlencoded'

The variable names to look for during the parsing are held in an eCos table. In order to take advantage of this feature, the user first adds the variable names to the table, which also requires providing a buffer where the parsed value will eventually be stored. The values will then be available in the buffers during the processing of the request, presumably in the body of a c language callback or CGI script.

The ATHTTP server gets the form variable (__fvar) from http GET/Post request, and store the value in __val. User can use “cyg_httpd_find_form_variable(char *p)” in the handler function to get the key from __val and process the request. Finally, server needs to send the response to client. ATHTTP server provides a number of functions for sending the response. These functions will handle the HTTP header, so customer can focus on the http body/content. Here is an example of response function:

```
#define web_send(x,size) \
        if(p != NULL) { \
                cyg_httpd_start_chunked("bin");\     /*This is the type . It can be “TXT” also. */
                        strcpy(p->outbuffer, x);\
                cyg_httpd_write_chunked(p->outbuffer, size); \
                cyg_httpd_end_chunked(); \
                return 0; \
        }
```

3. CYG_HTTPD_IRES_TABLE_ENTRY(__name, __path_name, __pmem, __size)
   - The parameter __name argument is a variable name for the table entry since C does not allow us to define anonymous data structures. This name should be chosen so that it is unique and does not pollute the name space.
   - The __path_name argument is the relative path to visit this webpage.
   - The __pmem argument is the string for web source code.
   - The __size argument is the length of __pmem.

This macro is used by the user who wants to provide web page by hard code. There is an firmware upgrade example in webpage.c :

```
static char upload_html[] =
    "<HTML><HEAD></HEAD>
    "<FORM action=/image.cgi method=post \
    encType=multipart/form-data >\
    <BODY>Upgrade    firmware:    <INPUT    type=file    name=userfile><INPUT    type=submit
value=Upload></BODY></FORM>"
"</HTML>"
CYG HTTPD IRES TABLE ENTRY(cyg ires entry upload, "/upload.html", upload html, sizeof(upload html));
```

Then the page http://xxx.xxx.xxx.xxx/upload.html can be use to upload firmware.(xxx.xxx.xxx.xxx is

ONU’s IP address)

4. CYG_HTTPD_AUTH_TABLE_ENTRY(__name, __path, __domain, __un, __pw, __mode)

- The parameter__name argument is a variable name for the table entry since C does not allow us to define anonymous data structures. This name should be chosen so that it is unique and does not pollute the name space.
- The __path argument is the relative path to visit this webpage.
- The __domain argument is the domain for webpage.
- The __un argument is the string for user name.
- The __pw argument is the string of password.
- The __mode argument is the mode for Authorization type.

The ATHTTP server supports both Basic (base64) and Digest (MD5) authentication. The contents of certain directories of the file system can be protected, such that the user will be required to issue a username/password to access the content of the directory.

For example:

```
CYG_HTTPD_AUTH_TABLE_ENTRY(onu_entry,"/","onu","admin","admin",CYG_HTTPD_AUTH_BASIC);
```

4.4.2 Example of Adding a web page

This section describes the steps to add a web page:

Step 1: Add the map MACRO and corresponding handle.

Add the MACROs in webpage.c

```
CYG_HTTPD_HANDLER_TABLE_ENTRY( cyg_default_index_entry, "/", cyg_default_index);
CYG_HTTPD_HANDLER_TABLE_ENTRY( cyg_index_files_entry,"/index_files/*",cyg_webpage_resource);
Char var_index [VAR_LEN];
CYG_HTTPD_FVAR_TABLE_ENTRY (cyg_fvar_entry_index, "index", var_index, VAR_LEN);
```

Step 2: Add the handler function.

```
Static cyg_default_index (CYG_HTTPD_STATE* p)
{ /*This function form is usual used by Set Request */
        Char index_key[]=”index”
        cyg_httpd_find_form_variable(index_key);
        …..
        web_send(x,size)
}
Static cyg_webpage_resource (CYG_HTTPD_STATE* p)
{/*This function form is usual used by Get Request */
        …..
        sprintf (x,………);
        …..
        web_send(x,size)
}
```

Step 3: Prepare HTML resource files

In general, CGI can generate some simple pages without problem. For some complex web design, it is not easy to generate all HTML resource files (*.html,*.js,*.gif...etc) by CGI. For this case, all these resource files can be stored in the file system. So customer can use general html editor to develop their web pages, and then convert all files into JFFS2 image by using Cortina proprietary utility called mkfs.jffs2. Mkfs.jffs2 is available in iROS-ONU package. Please refer to section 5.4.2 for the detail usage.

4.5 iROS-OOB Porting

Some applications such as MDU/HGU might have a more powerful CPU in backend SoC. For this case, CS8016C can work as slave CPU which only handles MPCP and PON-related OAM messages, other messages are handled by backend Master CPU. To implement this application, customer need to port iROS-OOB to external CPU. iROS-OOB will communicates with the IROS-ONU software components via connection through the management or UNI port of ONU. iROS-OOB is Linux-base which is easy to port to Linux PC or any embedded Linux system. In Most cases, customer just needs to re-write the Makefile and rebuild the iROS-OOB package with cross-compilation toolchain. After that, master CPU can call any iROS-OOB API in their application code. Please also refer to iROS-OOB API manual for the detail API definition.

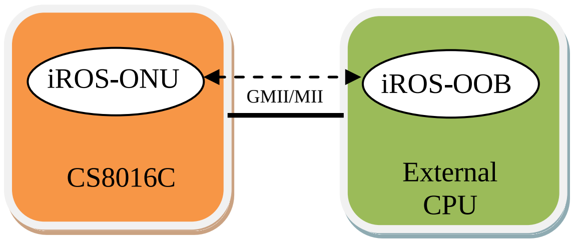


Figure 8 iROS-OOB out-band management

5. Appendix

5.1 Marcos definitions in iros_config.h

ONU Macro Name	Usage	Notes
HAVE_CTC_OAM	For CTC extension OAM	
HAVE_FLASH_FS	For Flash FS	2MB flash EVB don't support this option.
HAVE_LOG_THREAD	For System log	
HAVE_MPU	For memory protect	
HAVE_NOT_IGMP_SNOOPING	For IGMP SNOOP	
HAVE_ONU_8021X	For 802.1x	
HAVE_ONU_IP_STACK	For IP Stack support. (Httpd server will be)	HAVE_TERMINAL_SERVER HAVE_TELNET_CLI HAVE_ONU_SNMP
HAVE_ONU_SNMP	For SNMP	
HAVE_TELNET_CLI	For telnet	
HAVE_TERMINAL_SERVER	For tcp/uart translation	
HAVE_ONU_RSTP	For RSTP	
HAVE_VOIP	Not configurable	
HAVE_ONU_SPECIAL_PKT_RATE_LIMITER	Not configurable	
HAVE_ORGSPEC_OAM_HANDLER	Not configurable	
HAVE_SAL	Not configurable	
HAVE_SAL_DEBUG	Not configurable	
HAVE_EXT_SW_DRIVER	Not configurable	
HAVE_MINI_TARGET	Not configurable	
HAVE_ONU_GPIO	Not configurable	
HAVE_ONU_GPIO_SAMPLE	Not configurable	
HAVE_ONU_SERIAL	Not configurable	
HAVE_ONU_SERIAL_TEST	Not configurable	

Cortina Systems, Inc. Confidential

5.2 Startup Configuration

Table 1 Definition of Startup Configuration File

Offset	Type and Length	Parameter Name	Description	Read /Write	Factory Default	Note
0x0	1 Byte	EEPROM Control Flag	This flag is used to indicate where is the start up config data. 0x55 -- Start up config data is stored in EEPROM. 0xAA-- Start up config data is stored in Flash.	R/W	0x55	
0x1	6 Bytes	MAC Address	It is the MAC address of the device, provided by the ONU system manufacturer.	R/W	Set at the factory	
0x7	1 Byte	EEPROM Version	The current version of EEPROM.	R	1	
0x8 bit0-1	2 Bits	HEC Mode	Vendor-specific mode and 802.3ah compliant mode. The vendor specific mode is used to interwork with non-standard based implementations. 00 - 802.3ah -2004 01 -Vendor-specific 10 - Reserved 11 - Reserved	R/W	802.3ah-2004	
0x8 bit 2	1 Bit	IGMP Snooping Mode	A control bit is used to indicate whether IGMP snooping is enabled or not. 0 - Disabled 1 - Enabled	R/W	Enabled	Obsolete
0x8 bit 3-4	2 Bits	OAM Version	The OAM versions: 802.3ah/Draft2.0 are used to interwork with early versions of implementations of other vendors. 00 - 802.3ah -2004 01 - Reserved 10 - Reserved 11 - 802.3ah Draft 2.0	R/W	802.3ah -2004	
0x9	2 Bytes	MPCP Time Out	This is used to configure the value of MPCP time out. It is used for the detection of no GATE message within the specified time. The value is 50 or 1000.	R/W	1000 (One second)	
0xb	1 Byte	Vendor Code	The ONU system vendor code.	R	Set at the factory	
0xc	2 Bytes	Model Number	The ONU system model ID.	R	Set at the factory	
0xe	2 Bytes	Hardware Version	The ONU system hardware version.	R	Set at the factory	
0x10	2 Bytes	Year	The low two digits of the production year (00 – 99)	R	Set at the factory	
0x12	2 Bytes	Week	The production week (01 – 53)	R	Set at the factory	
0x14	6 Bytes	Serial Number	The ONU system serial number from 0 to 999999.	R	Set at the factory	
0x1a bit 1	1 Bit	Misc Config	CRC Mode 0 - Normal CRC; 1 - Inverse CRC		Normal CRC	Obsolete
0x1b bit 0	1 Bit	Query Key	This bit indicates whether or not ONU install IGMP query mac address. 0 - Disabled; 1 - Enabled	R/W	Disabled	Obsolete
0x1b bit 1	1 Bit	Disable_auto_reset	Disable ONU auto reset after OOB ONU resync beyond 2 times. 0--Enable reset 1--Disable reset	R/W	Enable reset	
0x1b bit2	1 Bit	enable_tk_default_mode		R/W		
0x1b bit 3	1 Bit	Normal_bringup_mode	Normal bringup mode 0- Normal bringup mode with full hardware check. 1-Fast bringup mode skip some hardware check to accelrate the bring up speed.	R/W	Normal bringup mode	
0x1c bit 0	1 Bit	PON Port Configuration: Laser always on	This is a control bit to keep the laser always on. 0 - Disabled; 1 - Enabled	R	Disabled	
0x1c bit 1	1 Bit	PON Port Configurarion: Laser Polarity	This bit indicates when the PON Port's Laser active 0 - Low active;1 - High active	R	Low active	
0x1c bit 7	1 Bit	PON Port Configuration: Admin status	This bit indicates the PON Port's admin status.It is valid only for OOB ONU. 0 - DOWN;1 - UP	R	UP	
0x1d bit 0-1	2 Bits	UNI Port Configuration: MAC Type	It indicates the MAC type of the UNI port. It is valid only when ONU Config code is 2 and UNI Port Configuration: Pinstrap is 0. 00 - MII 01 - GMII 10 - TBI 11 - Reserved	R/W	MII	
0x1d bit2	1 Bit	UNI Port Configuration: Auto negotiation	This control bit is used to turn on or off the auto negotiation on the ONU port. 0 - Disabled; 1 - Enabled	R/W	Enabled	
0x1d bit 3-5	3 Bits	UNI Port Configuration: MII Mode	It indicates the MII mode of the UNI port. It is valid only when ONU Config code is 2 and UNI Port Configuration: Pinstrap is 0. 000 - 10M half duplex 001 - 10M full duplex 010 - 100M half duplex 011 - 100M full duplex 100 - 1000M full duplex	R/W	100M full duplex	
0x1d bit 6	1 Bit	UNI Port Configuration: Pinstrap	This bit indicates which value is valid for UNI port configuration.This bit only valid when ONUConfigCode is 2. 0 - EEPROM settings valid 1 - Hardware Pins valid	R/W	EEPROM settings valid	
0x1d bit 7	1 Bit	UNI Port Configuration: Admin status	This bit indicates the UNI Port's admin status 0 - DOWN;1 - UP	R/W	UP	
0x1e	16 Bytes	User Name	The user name string for 802.1x authentication	R	Set at the factory	
0x2e	16 Bytes	Password	The password string for 802.1x authentication	R	Set at the factory	
0x3e bit 0	1 Bit	802.1x Mode	Enabled/Disabled 802.1x mode 0 - Disabled; 1 - Enabled	R/W	Enabled	
0x3e bit 1	1 Bit	UNI Port Control	When this bit is disabled , it indicates there is no data traffic flow allowed via UNI port. 0 - Disabled; 1 - Enabled	R/W	Enabled	
0x3e bit 2	1 Bit	Multicast/Broadcast Control	When this bit is disabled, it indicates there is no user multicast and broadcast traffic flow allowed in the upstream direction via the UNI port. 0 - Disabled; 1 - Enabled	R/W	Enabled	
0x3e bit 3	1 Bit	Security flag 802.1x tunnel mode	Enabled/Disabled 802.1x tunnel mode 0 - Disabled; 1 - Enabled	R/W	Disabled	
0x3f	2 Bytes	Reserved				
0x41	1 Byte	ONUConfigCode	This byte indicates ONU GE port's external device. 0 or FF - Default with PHY device 1 - Reserved with PHY device 2 - Pinstrap effective with PHY device FE - with Marvell Switch	R	0	
0x42	2 Bytes	ONUCtrlVlan	Vlan Id of OOB MGMT frame	R/W	0	
0x44	3 Bytes	CTC OUI	CTC extension OAM OUI	R/W	0x111111	
0x47	2 Bytes	Laser On time	The laser on time in 1/2 TQs	R/W	0x40	
0x49	1 Byte	Laser Off time	The laser off time in 1/2 TQs	R/W	0x40	
0x4a bit 0	1 Bit	OAM Mode	This bit indicaties whether CTC OAM packet is fowarded to UNI port. 0 - Disabled 1 - Enabled	R/W	Disabled	
0x4a bit 1-2	2 Bits	CRC Mode	This bit indicaties CRC mode. 00 - CTC Mode (Normal CRC) 01 - reserved 10 - PMC Mode (Inversed CRC)	R/W	CTC Mode	
0x4a bit 3	1 Bit	FEC Enabled	This bit indicaties whether FEC Function is enabled or not. 0 - Disabled; 1 - Enabled	R/W	Disabled	

0x4a bit 4	1 Bit	Unknow MC Drop	This bit indicaties whether Unknown Multicast drop is enabled or not. 0 - Disabled; 1 - Enabled	R/W	Disabled	Obsolete
0x4a bit 5	1 Bit	Tx Error Detect	This bit indicaties whether Tx Error detect is enabled or not. 0 - Disabled; 1 - Enabled	R	Disabled	Obsolete
0x4a bit 6-7	2 Bits	Igmp_vlan_learning_mode	Enable/disable strict VLAN checking on IGMP, If enable this mode, IGMP will follow all VLAN behaviors as data path. Otherwise, IGMP will follow the behaviors defined by CTC STD. 0 - Disable ; 1- Enalbe .	R/W	00B	
0x4b	1 Byte	Laser Delay	This bit indicates laser delay time in 1/2 TQs	R	0	
0x4c	4 Bytes	CTC Vendor ID	The Vendor ID string	R/W	IMST	
0x50	4 Bytes	CTC Model ID	The Model ID string	R/W	8015	
0x54	1 Byte	Deregister backofftime	When ONU received REGISTER frame with NACK flag.ONU will come into keep silence state. This byte indicates how long the ONU keep silence in Seconds (0-255) before next REGISTER REQUEST.	R/W	60	
0x55 bit 0-4	5 Bits	Mdio address	Mdio address	R/W	1	
0x55 bit 5	1 Bit	DyingGaspTriggerMode	This bit indicates the DyingGasp trigger Mode 0 - Rise edge trigger 1 - Falling edge trigger	R/W	Rise edge trigger	
0x55 bit 6	1 Bit	Reserved				
0x55 bit 7	1 Bit	MII_enable	This bit indicates the mii is enable or not 0 - Enable MII 1 - Disable MII	R/W	0	
0x56	1Byte	SwitchPortNum	This byte indicates the external switch port number.	R/W	4	
0x57	3Bytes	kt_onu	3 bytes used to specify OUI used by KT OAM messages.	R/W	0xaa 0xaa 0xaa	
0x5a	4Bytes	kt_olt_chip_model	4 bytes used to store current OLT chip model Chip model is retreived from link status exchange OAM (in the remote TLV info section). ONU is initialized differently based on the OLT chip model. Currently the following chip models are supported: { 0x37, 0x21, 0, 0 }, { 0x37, 0x23, 0, 0 }, { 0x37, 0, 0, 0 }, { 0x50, 0x01, 0, 0 }, { 0x50, 0, 0, 0 }, { 0x52, 0x01, 0, 0 }, { 0x52, 0, 0, 0 }	R/W	0,0,0,0	
0X5e	16Bytes	ModelName	IOP Extended ONU info	R/W	'0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'	
0x6e	2Bytes	ClassifRuleNum	The number of classification rule entries.	R/W	8	
0x70	1Byte	BoardType	The number indicates the board hardware schematic difference. 0x55 - T&W, Other - Other Board	R/W	0	
0x71 bit 0	1 Bit	CtcUpgrade Mode	This bit indicates whether CTC image download OAM packet is forwarded to management port. 0 - Enable MII 1 - Disable MII	R/W	0	

5.3 GPIO Definitions of CS8016C EVB

Table 2 Definition of Startup Configuration File

GPIO	**8016B/C- 1GE port**	**8016B/C-4FE port**
GPIO_0	**IN:** Dying Gasp (Falling Trigger)	
GPIO_1	**OUT**: Software Reset (1: Normal, 0: Reset, NOT USED CURRENTLY)	
GPIO_2	**OUT:** PON TX Shutdown (1: Normal, 0: Shutdown)	
GPIO_3	**OUT:** GE PHY Reset (1: Normal, 0: Reset)	**OUT:** Marvell Switch Reset (1: Normal, 0: Reset)
GPIO_4	Not Used	**IN:** Marvell Switch Interrupt (Falling Trigger)
GPIO_5	**IN**: Laser ON Push Button (1: Normal, 0: Always ON)	
GPIO_6	**IN**: CFG Restore and Reset Button (1: Normal, 0: Push [3 Second])	
GPIO_7	**OUT**: ONU LOS LED (1: OFF, 0: ON)	
GPIO_8	**OUT**: PON Link Event Alarm LED (1: OFF, 0: ON)	
GPIO_9	**OUT**: Register Status LED (1: OFF, 0: ON) MPCP+OAM Registered: ON NOT Finish OAM Register: DARK Registering MPCP or OAM: BLINK	
GPIO_10	**IN**: Laser ON Push Button Enable (1: Enable, 0: Disable)	
GPIO_11	Not Used	
GPIO_12	**OUT:** Flash Write Lock (1: Lock, 0: Unlock)	
GPIO_13	Not Used	
GPIO_14	Not Used	
GPIO_15	Not Used	

5.4 Tools

Some useful tools are available in iROS-ONU 3.0 package.

5.4.1 eeprom_onu_gen

Location: $IROS-ONU-ROOT/tools/support/tools/bin/eeprom_onu_gen

Usage: eeprom_onu_gen

This tool can generate startup configuration file for eeprom or flash.

Here is an example:

```
./eeprom_onu_gen
        Select this config for EEPROM/FLASH(0-EEPROM 1-Flash default EEPROM) ? 1
        MAC address (default 00:13:25:ff:ff:81) ?
        EEPROM version (default 1) ?
        HEC mode (0, 1, 2, 3. default 0) ?
        IGMP snooping mode (0 - disable; 1 - enable. default 1) ?
        OAM version (0, 1, 2, 3. default 0) ?
        I2C interface mode (0 - slave; 1 - master. default 1) ?
        MPCP timeout 0 to 68718 ms (default 1000 for one second) ?
        Vendor code (0 to 255. default 0) ?
        Model number (0 to 65535. default 0) ?
        Hardware version (0 to 65535. default 0) ?
        Year (0 to 99. default 0) ?
        Week (1 to 53. default 1) ?
        Serial number (0 to 999999. default 0) ?
        query key    (0 - default; 1 - query key.    default 0) ?
        disable auto reset    (0 - auto reset; 1 - disable auto reset.    default 0) ?
        normal bringup mode    (0 - normal bringup mode; 1 - fast bringup mode.    default 0) ?
        Laser on time (0 to 255. default 64) ?
        Laser off time (0 to 255. default 64) ?
        Laser always on (0 - disable; 1 - enable. default 0) ?
        LaserPolarity (0 or 1. default 0) ?
        PON admin status (0 - down; 1 - up. default 1) ?
        UNI port MAC type (0, 1, 2, 3. default 0) ?
        Auto negotiation (0 - disable; 1 - enable. default 1) ?
        MII type (0 to 4. default 3) ?
        PinStrap (0 - EEPROM settings valid; 1 - Hardware Pins valid. default 0) ?
        UNI admin status (0 - down; 1 - up. default 1) ?
        User name (up to 16 characters. default imst) ?
        Password (up to 16 characters. default imst) ?
        Secrity flag 802.1x mode (0 - disable; 1 - enable. default 1) ?
        Security flag UNI port control (0 - pin control; 1 - register control. default 0) ?
        Security flag multicast/broadcast control (0 - disable; 1 - enable. default 1) ?
        Security flag 802.1x tunnel mode (0 - disable; 1 - enable. default 0) ?
        IOPVendorCode (0 to 255. default 255) ?
        ONUConfigCode (0 to 255. default 0) ?
        ONUCtrlVlan (0 to 255. default 0) ?
        CTC ONU (default 00:00:00) ?
        CtcOamBypassMode (0 to 1. default 0) ?
        unknowMCFlood (0 to 1. default 0) ?
        tx_error_detection (0 to 1. default 0) ?
        Igmp_vlan_learning_mode (0 to 3. default 0) ?
        Laser delay time (0 to 255. default 0) ?
        onu_crc_mode_config (0 to 3. default 0) ?
        onu_fec_enable(0 to 1. default 0) ?
        Vendor Info (up to 8 characters. default 8016) ?
        DeregisterBackoffTime (0 to 255. default 60) ?
        mdio_addr (0 to 31. default 1) ?
        DyingGaspTriggerMode (0 or 1. default 0) ?
        mii_enable (0 or 1. default 0) ?
        SwitchPortNum (0 to 255. default 4) ?
        ModelName(up to 16 characters. default all '0') ?
        ClassifRuleNum (default 8) ?
        BoardType (default 0) ?
        CtcUpgradeMode (0 to 1. default 0) ?
        You have finished the configuration.
        This is your configuration:

        MAC address:    00:13:25:ff:ff:81
        EEPROM version: 1
        HEC mode: 0
        IGMP snooping mode: 1
        OAM version: 0
        I2C interface mode: 1
        MPCP timeout: 1000
        ...
        Save to file (Y/N - default Yes)?
        Save to file (default onu_eeprom.dat )?
        ONU EEPROM configuation saved to file onu_eeprom.dat
        Try again (Y/N - default Yes)? n
        Thank you! See you again!
```

5.4.2 mkfs.jffs2

Location: $IROS-ONU-ROOT/tools/support/tools/bin/mkfs.jffs2
Usage: mkfs.jffs2 [OPTIONS] [file system folder]

Make a JFFS2 file system image from an existing directory tree
Convert the host dir to webpage loadable format image
.wfs format is the jffs2 image to been upload by CS8016 ONU's built-in webpage

Options:

-p, --pad[=SIZE]	Pad output to SIZE bytes with 0xFF. If SIZE is not specified, the output is padded to the end of the final erase block
-r, -d, --root=DIR	Build file system from directory DIR (default: cwd)
-s, --pagesize=SIZE	Use page size (max data node size) SIZE (default: 4KiB)
-e, --eraseblock=SIZE	Use erase block size SIZE (default: 64KiB)
-l, --little-endian	Create a little-endian file system
-b, --big-endian	Create a big-endian file system
-D, --devtable=FILE	Use the named FILE as a device table file
-f, --faketime	Change all file times to '0' for regression testing
-q, --squash	Squash permissions and owners making all files be owned by root
-h, --help	Display this help text
-v, --version	Display version information

Here is an example:

```
mkfs.jffs2 -l -q -d ./testdir
CS8016 ONU jffs2 image for web upload [testdir.wfs] create succeed
```

This utility will generate 2 images below:

- testdir.jffs2
  This is original jffs2 binary image. It can be uploaded by boot loader (Please refer to section 3.6.4). This image is also used for compose the mif image.
- testdir.jffs2.wfs
  This image can be uploaded by the build-in page (upload.html).

5.4.3 flash_onu_gen

Location: $IROS-ONU-ROOT/tools/support/tools/bin/ flash_onu_gen
Usage: flash_onu_gen [Options]

This tool can assemble loader,startup_config_data,ONU app firmware and jffs2 image to a single image.
This single can be used to burn flash for manufacture production.

Options:

-o	output file=FILE	Use the named FILE as the output image
-l	loader_name=FILE	Use the named FILE as a bootloader image
-f	firmware_name=FILE	Use the named FILE as a ONU app firmware image
--fo	firware_offset=OFFSET	Use firware offset OFFSET

-s	startup_configure_data	Use the named FILE as a startup configure data "NULL" for no startup configure data
--so	startup_configure_offset=OFFSET	Use startup_configure_offset OFFSET
-j	jffs2 image=FILE	Use the named FILE as a jffs2 image file "NULL" for no jffs2 image
--jo	jffs2_offset=OFFSET	Use jffs2_offset OFFSET

Here are some examples:

```
1.          ____Default partition table: 8MB_________
         |BOOT                  0x2f00_0000         0x10000 |
         |STARTUP_CONFIG        0x2f01_0000         0x10000 |
         |SUPER_BLOCK           0x2f02_0000         0x10000 |
         |XIP                   0x2f03_0000         0x260000|
         |BLOB0                 0x2f29_0000         0x180000|
         |BLOB1                 0x2f41_0000         0x180000|
         |USER_DATA             0x2f59_0000         0x10000 |
         |CORE_DUMP             0x2f5a_0000         0x10000 |
         |some free space here                                |
         |JFFS2            0x2f70_0000        0x100000|
         ________________________________________
         ./flash_onu_gen -l loader.img      -o     8MB_init_jffs2.img
                   -f onu.zblob    --fo 0x290000
                   -s startup.dat   --so 0x10000
                   -j rootdir.jffs2          --jo 0x700000

2.          _____Default partition table: 4MB _______
         |BOOT                  0x2f00_0000         0x10000 |
         |STARTUP_CONFIG        0x2f01_0000         0x10000 |
         |SUPER_BLOCK           0x2f02_0000         0x10000 |
         |XIP                   0x2f03_0000           0x130000|
         |BLOB0                 0x2f16_0000         0xc0000 |
         |BLOB1                 0x2f22_0000         0xc0000 |
         |USER_DATA             0x2f2e_0000         0x10000 |
         |CORE_DUMP             0x2f2f_0000         0x10000 |
         |JFFS2                 0x2f30_0000         0x100000|
         _________________________________________
          ./flash_onu_gen    -l loader.img            -o 4MB_init_jffs2.img
                             -f onu.zblob             --fo 0x160000
                             -s startup.dat           --so 0x10000
                             -j rootdir.jffs2         --jo 0x300000

3.         _____Default partition table: 2MB ______
         |BOOT                  0x2f00_0000       0x10000|
         |STARTUP_CONFIG        0x2f01_0000       0x10000|
         |SUPER_BLOCK           0x2f02_0000       0x10000|
         |XIP                   0x2f03_0000       0xb0000|
         |BLOB0                 0x2f0e_0000       0x90000|
         |BLOB1                 0x2f17_0000       0x90000|
         _________________________________________
          ./flash_onu_gen    -l loader.img   -o     2MB_init.img
                             -f onu.zblob    --fo 0xe0000
                             -s startup.dat  --so 0x10000
                             -j NULL         --jo 0x0
```

5.4.4 multi_image_gen

Location: $IROS-ONU-ROOT/tools/support/tools/bin/multi_image_gen

Usage: multi_image_gen -[c/d] [mif_file] [-crc32] -[B/J/S] [file_version] [file_name] -[B/J/S]T [file_time] [file_version] [file_name] -UT [type] [decompress] [file_time] [file_version] [file_name]

This tool can assemble multiple images such as ONU app firmware, jffs2 image...etc into a single image which called mif (multi-firmware) image. This image can be used in OAM/Web to upgrade firmware remotely.

Example:

```
./multi_image_gen -c outputfile -B 0 blobfile_name -J 0 JSS2_filename
```

Cortina Systems, Inc. Confidential