ENT-AN1199 Application Note
Ocelot Unmanaged Software

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

VPPD-04413. 1.1 10/19

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.1

Revision 1.1 was published in October 2019. The following is a summary of changes in revision 1.1 of this document.

- Reference to CA51 was removed from the Purpose and Scope section. For more information, see Purpose and Scope, page 2.
- The Reference section was updated. For more information, see References, page 2.
- The Basic Commands section was updated. For more information, see Basic Commands, page 4.
- The introductory paragraph of the Software Customization chapter was updated. For more information, see Software Customization, page 14.
- The PHY Chip Selection section was updated. For more information, see PHY Chip Selection, page 17.
- The Firmware Update section was updated. For more information, see Firmware Update, page 20.
- The Compiler Kits section was updated. For more information, see Compiler Kits, page 24.
- The Understand the Port Status section was updated. For more information, see Understand the Port Status, page 24.
- The SPI NOR Flash Support List section was added. For more information, see SPI NOR Flash Support List, page 25.

1.2 Revision 1.0

Revision 1.0 was the first publication of this document.

2 Overview

The Ocelot Unmanaged Reference Software (or unmanaged software) manages a complete standalone unmanaged Industrial IoT Ethernet switch device. The unmanaged software runs in an 8051-derivative microcontroller and features layer 2 (L2) unmanaged switch functions, including Flow Control, MAC address learning, aging, and other layer 2 functions.

This document assumes the reader has basic understanding of layer 2 switch functions.

2.1 General Features

- 5 or 11 ports triple-speed Ethernet switch with 4 integrated copper PHYs
- Frame size up to 9600 bytes
- Non-blocking, wire-speed performance
- 4096 MAC addresses, automatic learning and aging
- VCore-Ie™ CPU system with integrated 250 MHz 8051 CPU with 64 KB internal memory
- A serial nor flash with at least 64K-byte capacity for embedded software storage
- A Serial GPIO controller for SFP module's control signals and acting as an LED controller
- LED indications for speeds (10M/100M or 1000M/2500M) and link activity for every front port
- Operates over standard Category 5 cabling at 10/100/1000 Mbps
- HP Auto MDI-/MDI-X
- PCIe for external CPU connectivity and can operate as a standard Ethernet port
- NPI for external CPU connectivity and can operate as a standard Ethernet port
- IEEE 1588 End-to-End Transparent Clock
- Microsemi ActiPHY™ Power Management
- Loop guard
- IEEE 802.3az Energy-Efficient Ethernet (software-configurable option)
- Link aggregation (software-configurable option)

2.2 Purpose and Scope

This document describes the implementation of the embedded software for an unmanaged switch based on the following Ocelot switch chips.

- VSC7511
- VSC7512

The software runs in the internal 8051 of the switch chip. Please refer the Ocelot Unmanaged Reference Design document for further details about the hardware.

This document describes the following:

- The features implemented in order to serve as an unmanaged switch
- The command line interface for device debugging and basic configurations
- The software architecture
- An overview of all source files
- The software compiler options
- How to build an executable image based on the source files and a Keil PK51 tool

2.3 References

- VSC7511 Datasheet
- VSC7512 Datasheet
- Ocelot Unmanaged Reference Design (VSC5634EV)
- ENT-AN1187 Using the Serial GPIO/LED Controller
- ENT-AN1186 System MAC Address Setup Guide for the Ocelot Unmanaged Switch

3 Functional Descriptions

This section provides some information about the functional aspects of the VSC7511/VSC7512 Industrial IoT Ethernet switch device.

3.1 Port Status Monitoring

It is very important for the software to continuously monitor the PHYs in order to set up the switch ports based on whether the link is down or up. If the link is up, the software needs to monitor the current speed, duplex mode, pause capabilities, and update the settings of the switch ports accordingly.

PHYs are being polled every 10 ms.

Note: In this document, “PHY”, “PHY monitor”, or “PHY task” may include both copper PHYs and/or optical SFP modules.

3.2 Flow Control

Flow control is enabled by default (can be enabled by CLI command).

3.3 Aging

To prevent an automatically learned MAC address from permanently remaining in the MAC address table, the aging function in the switch is initiated automatically. This is performed by the device with 300–600 seconds scan period

3.4 VLAN

The switch is VLAN unaware.

4 Command Line Interface (CLI)

CLI is used for issuing basic configuration commands and is used for device debugging.

4.1 Serial Port Set-up

To use the CLI, connect a PC COM port to the board through a RS-232 converter PCB or USB to RS232 interface and activate a terminal program. The COM port must be set up to run 8 data bits, 1 stop bit, no parity, 115200 baud, and without flow control.

4.2 Commands Input

The following are some characteristics of command inputs.

- Commands are not case sensitive.
- Use the backspace key to delete the last character.
- Use the up arrow key to show the previous commands.

4.3 Basic Commands

The runtime CLI command sets are dependent on the software configurations. The following sections introduce the commonly used commands for switch configurations or device debug.

Note: The runtime settings is not stored in the flash permanently. After power cycling, the settings is lost except for the `config save` command.

To display the CLI commands set of your system, type ?.

Figure 1 • **CLI Command Sets**

```
?
V : Show version
R <target> <offset> <addr>: Read from chip register
  --> Example: Read Chip ID register DEVCPU_GCB:CHIP_REGS:CHIP_ID
  -->                                0x71070000       0x0     0x0
  --> Command: R 0x71070000 0x0 0x0
W <target> <offset> <addr> <value>: Write switch register
I <uport> <addr> [<page>]: Read PHY register
  --> Example: Read user port 1 PHY ID from PHY register 2 page 0
  --> Command: I 1 0x2 0
O <uport> <addr> <value> [<page>]: Write PHY register
P : Show Port information
? : Show commands
K <0|1> <uport>: Flow control mode (0=Enable, 1=Disable)
M [<uport> [c]: Show/Clear MAC address entries (uport=0 for CPU port)
X : Reboot device
Z <0|1>: E-Col-Drop mode, linkup ports only (0=Disable, 1=Enable)
H <uport> [c]: Show/Clear port statistics (uport=0 for CPU port)
S <0|1>: Suspend/Resume applications (0=Resume, 1=Suspend)
D : Dump bytes from SPI flash
CONFIG                      : Show all configurations
CONFIG MAC xx:xx:xx:xx:xx:xx : Update MAC addresses in RAM
CONFIG SAVE                 : Program configurations at RAM to flash
```

4.3.1 Read Switch Register

This command reads the switch register address.

Syntax:

`R <target> <offset> <addr>`

Where,

- `<target>`—targets base address

- <offset>—groups offset within target
- <addr>—registers address within group

4.3.2 Write Switch Register

This command writes the switch register address.

Syntax:

```
W <target> <offset> <addr> <value>
```

Where,

- <target>—targets base address
- <offset>—groups offset within target
- <addr>—registers address within group
- <value>—registers value

4.3.3 PHY Read Register

This command reads the PHY register for a port.

Syntax:

```
I <uport> <addr> [<page>]
```

Where,

- <uport>—reads the port number starting from 1
- <addr>—reads the PHY address
- <page>—reads the PHY register page

4.3.4 PHY Write Register

This command writes the PHY register for a port.

Syntax:

```
O <uport> <addr> <value> [<page>]
```

Where,

- <uport>—registers the port number starting from 1
- <addr>—registers the PHY address
- <page>—registers the PHY register page
- <value>—registers value to be written

4.3.5 Version

This command displays the version of software, chip, and hardware.

Syntax:

```
v
```

4.3.6 Show Port Information

This command displays port mappings, PHY ID, and link status.

Syntax:

```
p
```

See Appendix, page 24 for port status details.

4.3.7 Flow Control Enabled/Disabled

This command enables or disables the advertisement of flow control capability.

Syntax:

```
K <0|1> <uport>: Flow control mode (0=Enable, 1=Disable)
```

Where,

- <0|1>—enables or disables flow control
- <uport>—registers the port number starting from 1

4.3.8 MAC Address Table Operations

This command displays or clears the MAC table entries of a port.

Syntax:

```
M [<uport> [c]: Show/Clear MAC address entries (uport=0 for CPU port)
```

Where,

- <uport>—registers the port number starting from 1
- <c>—optional. If it is specified, then the MAC table of the specified port is cleared.

4.3.9 Port Statistics

This command displays or clears the port statistics, including the error statistics.

Syntax:

```
H <uport> [c]: Show/Clear port statistics (uport=0 for CPU port)
```

Where,

- <uport>—registers the port number starting from 1
- <c>—optional. If it is specified, then the specified port statistics is cleared.

4.3.10 Suspend or Resume the Applications

This command suspends or resumes the application to start or stop the periodic polling tasks (for example, port monitoring).

Syntax:

```
S <0|1>: Suspend/Resume applications (0=Resume, 1=Suspend)
```

Where,

- <0/1>—resumes or suspends the application to start or stop the periodically polling tasks

4.3.11 Link Aggregation Group (LAG) Management

This command adds, deletes, or shows the runtime static LAG(s).

Syntax:

```
E <0|1|2> <iport_mask>: Trunk configuration (0=Add, 1=Del, 2=Show)
```

Where,

- <0|1|2>—0=Add, 1=Delete, 2=Show status
- <iport_mask>–internal port bit mask for adding or deleting port members to a new group. That is, user port x maps to mask bit (x-1), the port mapping can be shown by using command p.

4.3.12 Link Aggregation Control Protocol (LACP) Management

This command enables, disables, or shows the runtime aggregation group(s).

Syntax:

```
F <0|1|2> <uport> [<key>]: LACP configuration (0=Enable, 1=Disable, 2=Show)
```

Where,

- <0|1|2>—0=Enable, 1=Disable, 2=Show a status overview for all LACP instances
- <uport>–registers the port number starting from 1.

5 Software Architecture

The following sections discusses the software architecture of the unmanaged software in detail.

5.1 Task Execution

The software does not contain an operating system. The task concept, however, is maintained. Tasks are executed in a round-robin loop and are implemented as functions that return within a proper time—they may have to be implemented as state machines. The body of the round-robin loop is located in the function `main_execute` in `main.c` and is called from an eternal loop in `main.c`. For convenience, `main_execute` is called the main task. Besides awaking other tasks, the main task may also perform miscellaneous in-line jobs itself. By means of timeout flags, set by the timer interrupt function described below, the main task may awake tasks and do in-line jobs at certain time-intervals.

One of the built-in hardware, timers of the 8051 micro-controller is set up to generate interrupt every 1 ms. The associated interrupt function ticks various software timers and sets timeout flags to be monitored by the main task or other tasks.

Apart from the timer interrupts, the only interrupt option utilized is the reception complete interrupt from the UART. The associated interrupt function inserts the received character into a software-controlled buffer in RAM for later processing (by the CLI handler). Transmission through the UART is done through polling.

5.2 Task and Module Diagrams

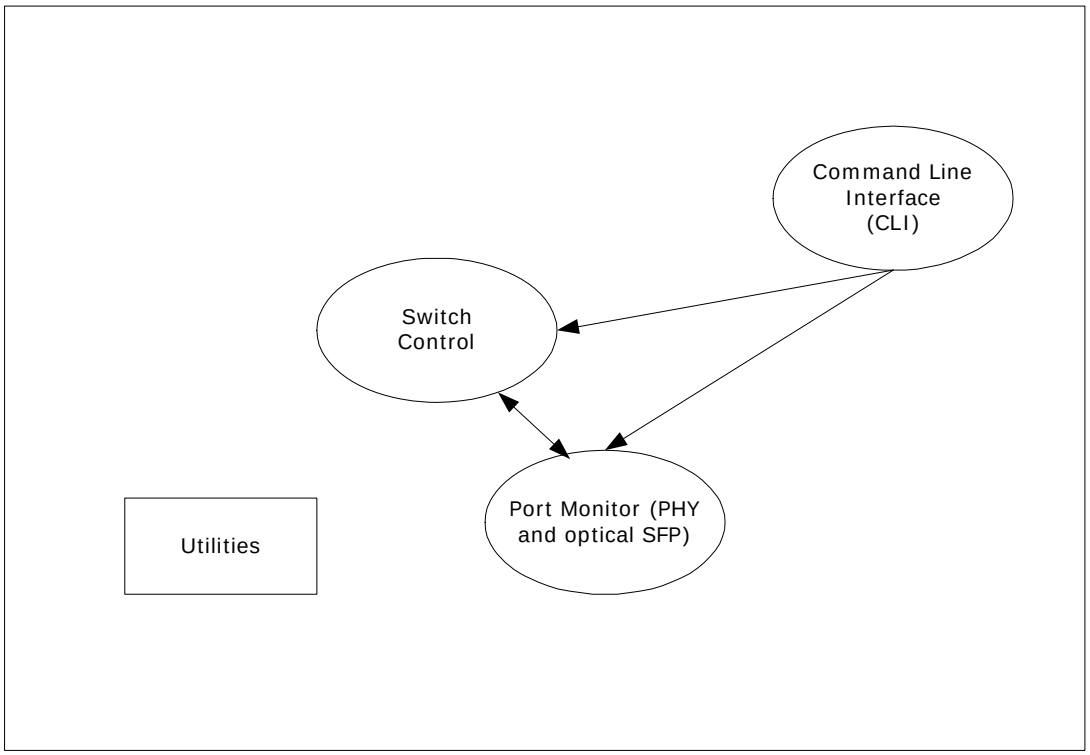
The simplified diagrams in this chapter provide an overview of the tasks, modules, and the connections between them. The following legend applies to the diagrams:

- Boxes indicate modules (a collection of functions and data).
- Ovals indicate tasks (a collection of an autonomous function, functions, and data).
- Clouds indicate subsystems that are not subject to more details within the context.
- Arrows indicate the directions of calls.

5.2.1 Overview of Subsystems

The entire software can be divided into the subsystems as depicted in the following illustration.

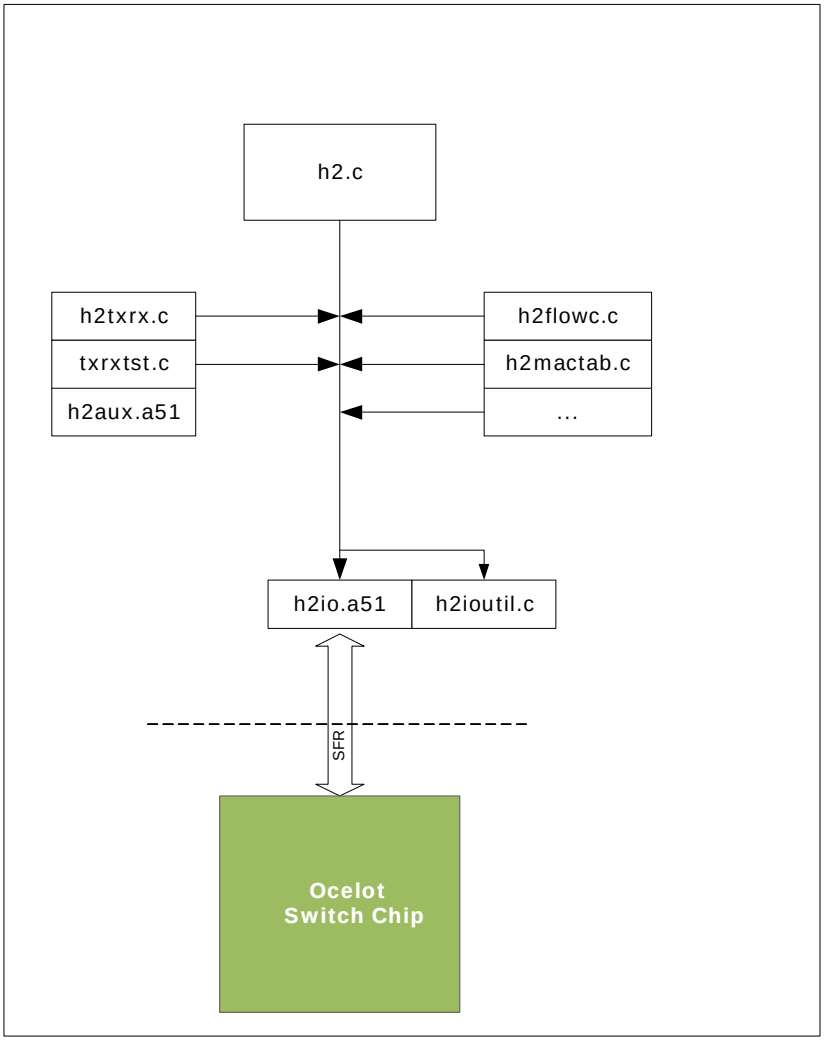
Figure 2 • **Overview of Subsystems**



5.2.2 Switch Control Subsystem

The switch control subsystem controls setup and monitoring of the switch. It does not have one single task, but a number of periodic tasks that are called from within the round-robin loop and a number of setup and status functions that are called at certain events (for example, PHYs link state).

Figure 3 • **Switch Control Subsystem**



The periodic tasks comprises:

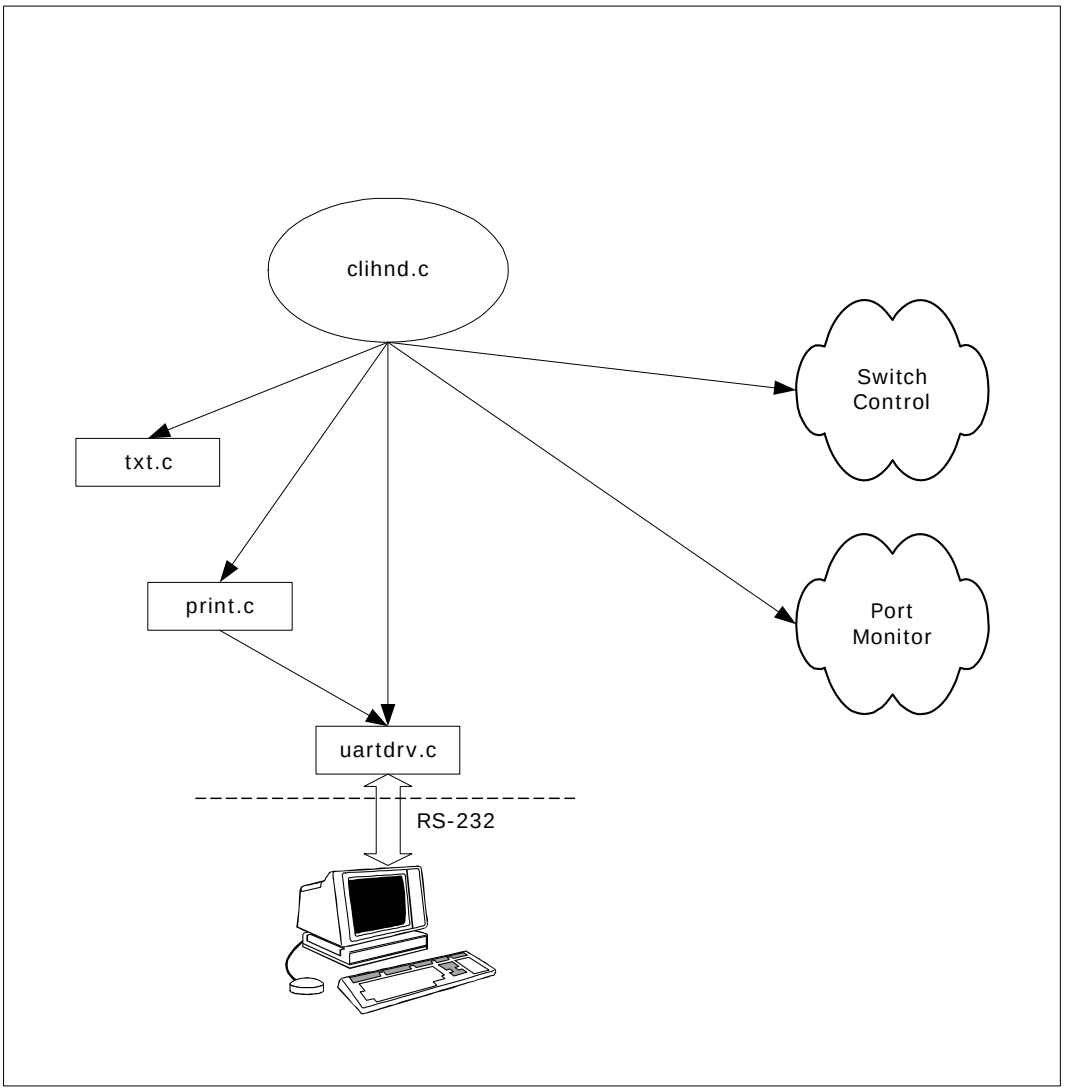
- Aging of MAC address table.
- Setup of port according to PHY link and auto-negotiation status.
- A low-level interface with general functions for reading and writing switch and PHY registers.

5.2.3 CLI Subsystem

The CLI handles commands received from the serial port. The CLI handler (`clihnd.c`) maintains the following overall control of:

- Poll the terminal driver for a ready command.
- Retrieve the command.
- Validate the command.
- Execute the command, typically through function calls into the switch control subsystem or the PHY monitor control subsystem.
- Possibly give response to the command in form of data or error message.

Figure 4 • **Command Line Interface Subsystem**

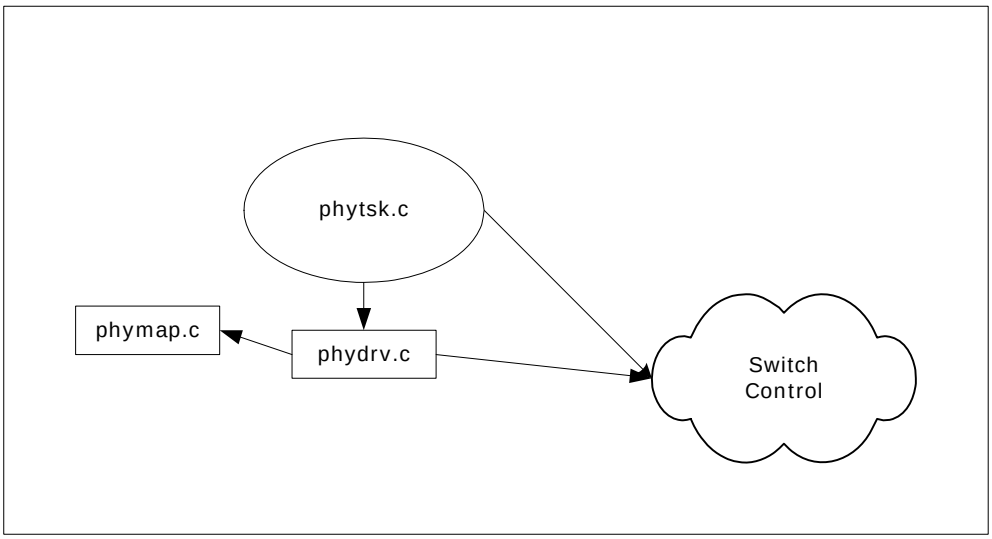


5.2.4 PHY Monitor Subsystem

The PHY monitor subsystem handles the PHYs and optical SFP modules. The PHY task (`phytsk.c`) maintains the following overall control of:

- Map between switch ports and PHYs or SGMII/SerDes/QSGMII/NPI interfaces.
- Set up PHYs or SGMII/SerDes/QSGMII/NPI interfaces according to the hardware configuration.
- Set up PHYs or SGMII/SerDes/QSGMII/NPI interfaces in terms of auto-negotiation, speed, and duplex mode.
- Monitor link state.
- Set up switch ports according to link state and actual speed, duplex mode, and support of pause frames.

Figure 5 • **PHY Monitor Subsystem**

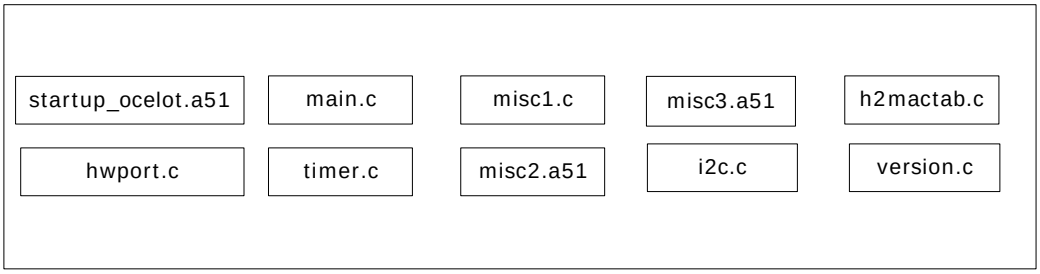


5.2.5 Utilities Subsystem

The utilities subsystem provides the following features:

- Control of start-up sequence.
- Control of round-robin loop (main task).
- Timer interrupt function.
- Miscellaneous general help functions.
- Hardware-dependent functions.
- Software version identification

Figure 6 • **Utilities Subsystem**



5.3 Overview of Source Files

The following table lists the files of switch control subsystem.

Table 1 • **Switch Control Subsystem**

File	Description
`h2.c`	Functions for port set up and miscellaneous operations.
`h2mactab.c`	Functions for operations on the MAC table.
`h2io.a51`	Functions for reading and writing switch registers.
`h2ioutil.c`	Functions for doing masked writings to switch registers.
`txrxtst.c`	Functions for loopback tests.

Table 1 • **Switch Control Subsystem**

File	Description
h2flowc.c	Control of flow control and watermarks settings.
h2txrx.c	Functions for transmitting and receiving frames from and to CPU.
h2aux.a51	Help functions for improvement of performance.
h2.h	Header file for h2.c
h2mactab.h	Header file for h2mactab.c
h2io.h	Header file for h2io.a51 and h2ioutil.c
txrxtst.h	Header file for txrxtst.c
h2flowc.h	Header file for h2flowc.c Contains watermarks values.
h2txrx.h	Header file for h2txrx.c
h2aux.h	Header file for h2aux.a51
h2reg.h	Definitions of the switch registers.

The following table lists the files of switch command line interface subsystems.

Table 2 • **Command Line Interface Subsystem**

File	Description
clihnd.c	Overall control of CLI.
txt.c	Definitions of texts and CLI command strings.
print.c	Holds functions for formatting data.
uartdrv.c	UART driver.
clihnd.h	Header file for clihnd.c
txt.h	Header file for txt.c.
print.h	Header file for print.c
uartdrv.h	Header file for uartdrv.c.
phytsk.c	Monitoring of PHYs.
phydrv.c	Functions for reading and writing PHY registers, resetting PHYs, and so forth.
phymap.c	Tables for mapping between switch ports and PHYs (MIIM and PHY address).
phytsk.h	Header file for phytsk.c.
phydrv.h	Header file for phydrv.c.
phymap.h	Header file for phymap.c.
phyconf.h	PHY configuration parameters.

The following table lists the files of switch utilities subsystem.

Table 3 • **Utilities Subsystem**

File	Description
`ocelot_startup.a51`	Overall control of CLI.
`hwport.c`	Definitions of texts and CLI command strings.
`main.c`	Initialization sequence, round robin loop, and error handling.
`timer.c`	Functions for timer interrupt and time monitoring support.
`misc1.c`	General support functions.
`misc2.a51`	General support functions optimized for speed or size.
`version.c`	Software version string.
`macaddr.c`	MAC address configuration.
`hwport.h`	Header file for hwport.c.
`main.h`	Header file for main.c
`timer.h`	Header file for timer.c
`misc1.h`	Header file for misc1.c.
`misc2.h`	Header file for misc2.a51
`version.h`	Header file for version.c.
`macaddr.h`	Header file for macaddr.c.
`common.h`	Common definitions.
`hwconf.h`	Hardware-dependant configuration.
`hwconf.inc`	Hardware-dependant configuration (for assembler modules).

6 Software Customization

To customize the system, the unmanaged software provides some configurations for reference, which are located in folder, `\src\config\proj_opt\`.

The Ocelot release binary files based on `proj_opt_release.h` are used by selecting the `Ferret_Unmanaged_Release.uvproj` project file.

Before software customization, read `ReleaseNotes.txt` in the doc folder under the software package

6.1 Compile-Time Options

The `#define` preprocessor directive defines various preprocessor symbols that configure optional compile-time behavior of the unmanaged software.

The following sections discuss how to customize the application by changing or commenting out the macro definition(s).

6.1.1 Node Processor Interface

The following syntax selects a chip port as Node Processor Interface (NPI).

Syntax: `#define NPI_CHIP_PORT <Chip Port>`

Where, `<Chip Port>`—specifies the chip port to be configured as NPI. It starts from 0.

Note: If `<Chip Port>` value is specified as `NPI_ACT_NORMAL_PORT` (system default), then the NPI mode is disabled.

6.1.2 System Control

The following syntaxes help perform several system control related configurations.

- To enable and disable the system MAC address at runtime. It is usually used with LACP or LLDP enabled system.
  - Syntax: `#define TRANSIT_UNMANAGED_SYS_MAC_CONF <value>`
  - Where, `<value>` is set to 0 to disable the system MAC address at runtime and to 1 to enable the system MAC address at runtime.
- To enable or disable the loop detection capability.
  - Syntax: `#define TRANSIT_LOOPDETECT <value>`
  - Where, `<value>` is set to 0 to disable the loop detection capability and to 1 to enable the loop detection capability.
- To support Link Aggregation Groups (LAG).
  - Syntax: `#define TRANSIT_LAG <value>`
  - Where, `<value>` is set to 0 to disable the support for LAG and to 1 to enable the support for LAG.
- To support Link Aggregation Control Protocol (LACP).
  - Syntax: `#define TRANSIT_LACP <value>`
  - Where, `<value>` is set to 0 to disable the support for LACP and to 1 to enable the support for LACP.

6.1.3 Port Control

The following syntax enables or disables the advertisement of flow control capability.

Syntax: `#define TRANSIT_FLOW_CTRL_DEFAULT <value>`

Where, `<value>` is set to 0 to disable the advertisement of flow control capability and to 1 to enable the disable the advertisement of flow control capability.

6.1.4 PHY Energy Management

The following syntax enables or disables the advertisement of flow control capability.

- To enable or disable the ACTIPHY™ Power Management.
  - Syntax: `#define TRANSIT_ACTIPHY <value>`
  - Where, `<value>` is set to 0 to disable the ACTIPHY™ power management and to 1 to enable the ACTIPHY™ power management.
- To enable or disable the IEEE 802.3az Energy Efficient Ethernet.
  - Syntax: `#define TRANSIT_EEE <value>`
  - Where, `<value>` is set to 0 to disable the IEEE 802.3az Energy Efficient Ethernet and to 1 to enable the IEEE 802.3az Energy Efficient Ethernet.

6.1.5 IEEE 1588 PTP End-to-End Transparent Clock

The following syntax enables or disables the PTP end-to-end transparent clock.

Syntax: `#define TRANSIT_E2ETC <value_0>`

`#define TRANSIT_TCAM_IS2 <value_1>`

Where, (`<value_0>, <value_1>`)is set to (0, 0) to disable the PTP end-to-end transparent clock and to (1, 1) to enable the PTP end-to-end transparent clock.

6.2 Port Mappings

The following sections describe port mapping.

6.2.1 Port Mapping Configuration

This section uses VSC5634EV reference board as an example to explain the port customization.

VSC5634EV reference board demonstrates an 11-port unmanaged switch system that has a VSC7512 chip (with 4 integrated copper PHYs), a 4-port external copper PHY chip and a single copper PHY chip. Based on the VSC5634EV hardware architecture, the unmanaged software provides four reference configurations to help users customize the port configuration to fit their switch application.

The following illustration shows the user ports to chip ports mapping of VSC5634EV.

Figure 7 • **User Ports to Chip Ports Mapping of VSC5634EV**

U2 (C5)	U4 (C7)	U6 (C1)	U8 (C3)	U10 (C10)	U11/NPI (C9)
U1 (C4)	U3 (C6)	U5 (C0)	U7 (C2)	U9 (C8)	

Where,

- Um: User port m (where m = 2, 4, 6...)
- Cn: Chip port n (where n = 5, 7, 1...)
- NPI: Node processor interface

The following four reference configurations are provided by unmanaged software.

- 8 copper ports + 2 optical SFP ports + 1 NPI port—by selecting target F11 from Keil µVision IDE menu and then FERRET_F11 is defined.
- 8 copper ports + 2 optical SFP ports + 1 PCIe port—by selecting target F10P from Keil µVision IDE menu and then FERRET_F10P is defined.
- 4 Copper ports + 1 NPI port—by selecting target F5 from Keil µVision IDE menu and then FERRET_F5 is defined.
- 4 Copper ports + 1 PCIe port—by selecting target F4P from Keil µVision IDE menu and then FERRET_F4P is defined.

The following table uses F11 (FERRET_F11) to introduce the software data structures related to port customization.

Table 4 • **Software Data Structures for Port Customization**

Syntax	Description
See src/config/common.h, #define NO_OF_BOARD_PORTS 11 /* 11 for FERRET_F11*/	Specifies the total user port count.
See src/config/common.h, #define ALL_PORTS 0x000007FF /*bit0~bit10 are set for FERRET_F11*/	Sets the user port bit mapping accordingly.
See src/config/common.h, #define LED_PORTS (0x000005FF | VTSS_BIT(SYS_LED_SGPIO_PORT)) /* FERRET_F11: SIO port bits 0-7 are enabled for chip port 0-7 SIO port bits 8 and 10 are enabled for chip port 8 and 10 SIO port bit 36 is enabled for system status indicator (VSC5634EV does not implement the LED indication for NPI port). */	Sets the SIO bits according to the board's serial LED deployment.
See src/config/hwconf.h #define UPORT_MAPTO_CPORT { 4, 5, 6, 7, 0, 1, 2, 3, 8, 10, 9 } /* FERRET_F11: user ports 1~4 are mapped to chip port 4~7, user ports 5~8 are mapped to chip 0~3 user ports 9,10,11 are mapped to chip ports 8,10,9 */	Specifies the mapping of the user ports to chip ports.
See src/config/hwconf.h #define CPORT_MAPTO_UPORT { 5, 6, 7, 8, 1, 2, 3, 4, 9, 11, 10 } /* FERRET_F11: device ports 0~3 are mapped to user ports 5~8, device ports 4~7 are mapped to user ports 1~4, device port 8 is mapped to user port 9 device port 9 is mapped to user port 11 device port 10 is mapped to user port 10 */	Specifies the mapping of the device ports to the user ports.

Table 4 • **Software Data Structures for Port Customization**

Syntax	Description
See /src/config/phyconf.h #define CPORT_MAPTO_MIIMBUS { 0, 0, 0, 0, 1, 1, 1, 1, 0xa, 1, 0xa } /* FERRET_F11: chip ports 0~3 connect to MIIM controller 0 chip ports 4~7 connect to MIIM controller 1 chip Port 9 connects to MIIM controller 1 chip port 8 and 10 are optical SFPs so 0xa is specified for auto SFP selection */	Specifies PHY MIIM controller.
See /src/config/phyconf.h #define CPORT_MAPTO_MIIMADDR {0, 1, 2, 3, 4, 5, 6, 7, 20, 0x1c, 21} /* FERRET_F11: chip ports 0~7 are configured as address 0~7 chip port 9 is configured as address 0x1c chip Ports 8 and 10 are SerDes SFP. VSC5634EV uses GPIO20 and GPIO21 pins for TWI SCK(Serial Clock Line) pins. So 20 and 21 must be specified.(See Understand the Port Status, page 24) */	Specifies the PHY address of the each copper port according to the PHY address setting.

6.2.2 PHY Chip Selection

The following setting identifies the PHY chips driver to compile when building the software code.

```
See /src/config/phyconf.h
VSC5634EV uses COBRA (VSC8221) PHY for user port 10:
#define VTSS_COBRA 1 /*define 1 to compile the COBRA PHY driver code*/
VSC5634EV uses ELISE (VSC8514) quad-port PHY for user ports 1~4:
#define VTSS_ELISE 1 /*define 1 to compile the ELISE PHY driver code*/
```

Note: The first port of a VSC PHY chip must be used in a switch board because the VSC PHY chip is initialized through the first port.

6.2.3 NPI Mode Configuration

NPI can be configured either as a standard Ethernet port or as an SGMII/SerDes interface for external CPU register access or packet injection and extraction.

```
#define NPI_CHIP_PORT  NPI_ACT_NORMAL_PORT
/* FERRET_F11 configures NPI as a Ethernet port */
```

Note: Select any chip port as NPI for external CPU access, for example, using chip port 9 as the NPI by setting NPI_CHIP_PORT as 9:

```
#define NPI_CHIP_PORT 9
```

As NPI operates as Ethernet port, the packet forwarding throughput can be up to 2500 Mbps.

6.2.4 PCIe Interface Configuration

Ocelot provides an on-chip PCIe 1.x endpoint controller for external CPU connectivity (includes accessing VCore-Ie CPU internal memory or 8051 SFR). PCIe can also operate as a standard Ethernet port.

The limitation of using PCIe interface is that switch chip registers can not be accessed through PCIe interface. Basically, PCIe can work as data port, but not management interface (that is, cannot access switch registers by PCIe). To access switch registers by external CPU, it is recommended to use the SPI

interface in the slave mode. As PCIe interface operates as Ethernet port, the packet forwarding throughput can be up to 100 Mbps.

The following definition must be declared if you need to use the PCIe interface.

```
#define PCIE_CHIP_PORT      10
/* defined for FERRET_F10P or FERRET_F4P target */
```

7 Build Target

To build the target software, a Keil PK51 tool should be used.

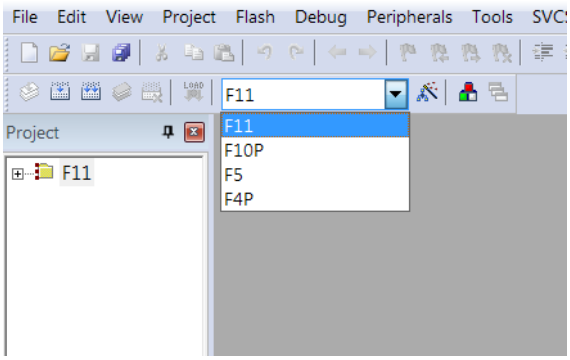
Note: The naming convention for the project files is `xxx.uvproj` (for example, `Ferret_Unmanaged_Release.uvproj`).

Note: Only the Keil µVision-based project manager file is provided; for other compilers, users have to port the code on their own.

Four build targets are provided and can be selected from Keil µVision IDE menu.

- F11: FERRET_F11
- F10P: FERRET_F10P
- F5: FERRET_F5
- F4P: FERRET_4P

Figure 8 • **Build Targets**



The generated hex file can be converted to binary by using a hex-to-bin converter that can be downloaded from Keil's website (`http://www.keil.com/download/docs/7.asp`).

8 Firmware Update

Unmanaged software does not provide a firmware update utility; therefore, a serial NOR flash memory programmer is required. The following example is demonstrated with the FORTE memory programmer from ASIX. Other memory programmers will work as well, but covering their installation methods is out of the scope of this document.

Ensure that the following requirements are fulfilled:

- A board as a target
- A flash memory programmer (FORTE is recommended, PRESTO is slower but works too)
- A PC running Windows 7 or Windows 10 (64-bit)
- Programmer software, UP (downloaded from ASIX.net) installed
- Binary or HEX flash image for the specific board

Once all the hardware is in place and all the drivers and the programmer software is installed, go ahead and start the ASIX UP program. The following pop up window will prompt the user to select and connect to their programmer.

Perform the following steps:

1. Start the ASIX UP software.

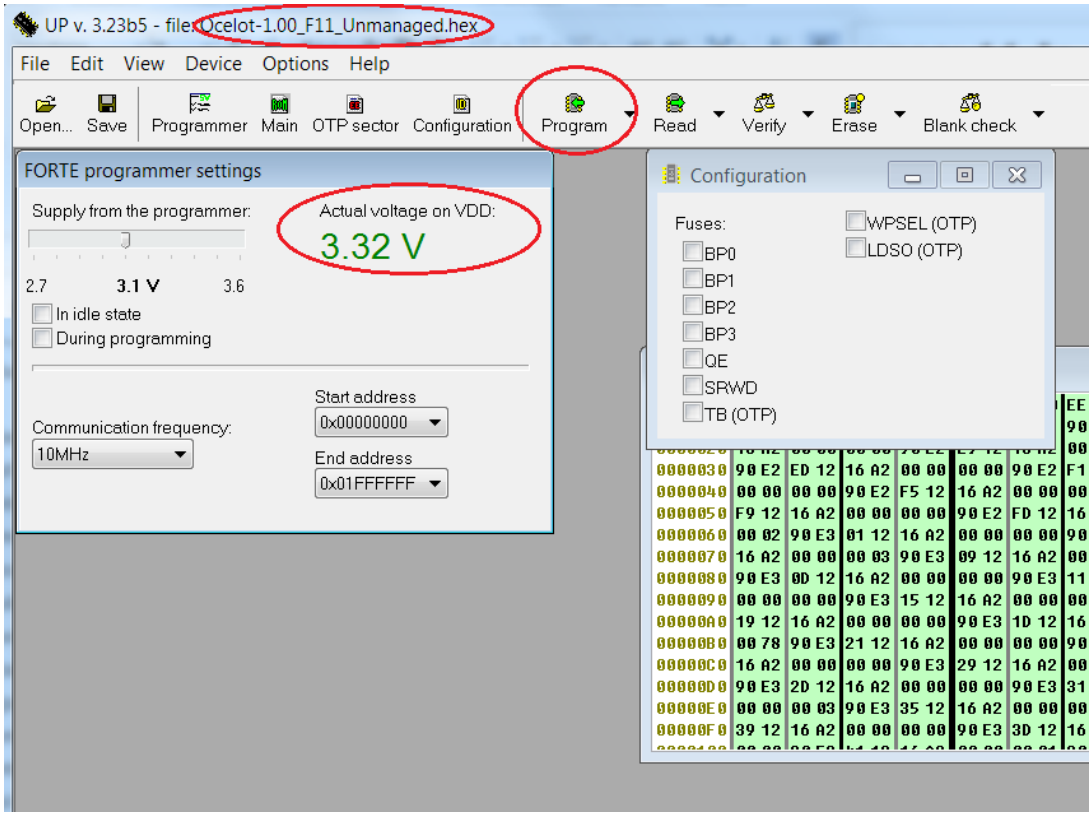
Figure 9 • **Start ASIX UP Software**

2. Select the SPI NOR flash device.

Figure 10 • **Select SPI NOR Flash Device**

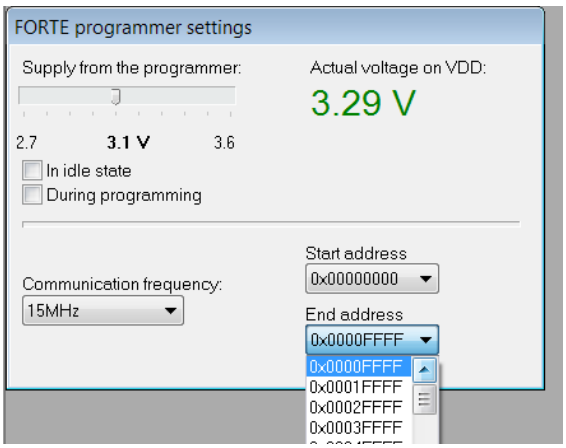
3. Connect the programmer cable to the board and make sure the Actual voltage on VDD is supplied as shown in the following illustration. Select the to-be-programmed HEX or BIN file and then run **Program** from ASIX UP.

Figure 11 • **Run Program**



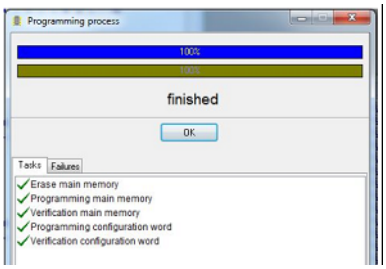
Note: As the unmanaged software image size is within 64K bytes, you can specify smaller end address to save the programming time (this feature needs the programmer's support). For VSC5634EV,a 32 Mbyte sized flash is used, you can specify 0xFFFF(64K address range) through ASIX UP to do the partial programming.

Figure 12 • **FORTE Programmer Settings**



4. Wait until the programming is done.

Figure 13 • **Wait for Programming**



5. Unplug the cable, then the system will boot automatically and display the following message.

Figure 14 • **Message**

```
Chip Family     : 7512 Rev.B
Software Version: Ferret_F11 Ocelot_Unmanaged-1.0.3
Code Revision   : 5b1628b+
Build Date      : Sep 18 2019 10:22:23
HW Revision     : 2

Enter ? to get the CLI help command
```

9 Load Image to VCore-Ie Through External CPU

For some applications, it is desired to have the Ocelot device running as a separate unit, and thereby offloading the external CPU of the system. The external CPU can access the on-chip RAM of the VCore-Ie CPU, which allows the external CPU to load the internal 8051 software to internal RAM, where the internal 8051 can execute from.

For Ocelot, use either SPI or PCIe interface to implement this application.

10 Appendix

10.1 SFP Connector Design Considerations

As Unmanaged Software uses the on-chip two-wire serial interface (TWI) to access the SFP module's EEPROM and then setup the port-related registers, GPIO16 (used as TWI's SDA), GPIO20 and/or GPIO21 (used as TWI's SCL) must be connected to SFP connector(s), otherwise the SerDes interface might be configured improperly.

10.2 Optical SFP Module support List

The following is a list of supported optical SFP modules.

- SPT-SFP+C1, 10G SFP+ DAC 1M Passive Cable
- FTLF8519P3BNL, Finisar 1000Base-SX 850nm MM 550m
- SFP-SX-M2002, Excom 1000Base-SX 850nm MM 550m
- SFP-LH-S1010, Excom 100Base-LX 1310nm SM 10km
- SFP-SX-M1002, Excom 100Base-SX 1310nm MM 2km

Note: The copper SFP module is not supported.

10.3 Copper PHY support List

The following is a list of supported copper PHYs.

- COBRA (VSC8211, VSC8221)
- ELISE (VSC8514)

10.4 Compiler Kits

KEIL PK51 Professional Developer's Kit is used to build the software; internally PK51 version 9.03, 9.56 or 9.57 is used. Other versions with µVision5 V5 IDE provided with the kit should also work.

Note: The KEIL PK51 trial versions will not be able to build the software properly.

10.5 Understand the Port Status

Issuing the CLI command 'p' or 'P' will display the port status, including port mapping configuration and link status.

The following image is an example of the port status of VSC5634EV.

Figure 15 • **Port Status of VSC5634EV**

```
p
uPort iPort cPort MIIM Bus MIIM Addr   PHY/Serdes  CRC uPatch Link Status
-------------------------------------------------------------------------------
    1     0     4         1      0x04 ELISE_8514   No uPatch   Up - 1GFDX FC(D)
    2     1     5         1      0x05 ELISE_8514   No uPatch   Down
    3     2     6         1      0x06 ELISE_8514   No uPatch   Down
    4     3     7         1      0x07 ELISE_8514   No uPatch   Down
    5     4     0         0      0x00 FERRET_7512  No uPatch   Down
    6     5     1         0      0x01 FERRET_7512  No uPatch   Down
    7     6     2         0      0x02 FERRET_7512  No uPatch   Down
    8     7     3         0      0x03 FERRET_7512  No uPatch   Down
    9     8     8         8      0x14 100M SFP     Not PHY     Down
   10     9    10         9      0x15 1G SFP       Not PHY     Down
   11    10     9         1      0x1c COBRA        No uPatch   Up - 1GFDX FC(D)
>
```

Where,

- uPort—the switch front port (user port) index
- iPort—the switch API port index (normally equals to uPort - 1)
- cPort— the internal chip port module, which maps to the uPort

- MIIM Bus: for copper ports, it is the MIIM controller ID which is decided at compile time.
  - Controller 0 is for internal copper PHYs and controller 1 is for external copper PHYs.
  - For SFP ports, it is the connected SFP module's capability read from SFP module's EEPROM. Supported capability are:

    8—indicates a 100M module

    9—indicates a 1000M module

    11—indicates a 2500M module

    10—auto-detected, default value
- MIIM Addr—for copper port, it is the PHY address and for SFP port, it is the GPIO pin used for TWI.
- PHY/Serdes—the copper PHY's family name or SerDes interface mode.
- Link Status—indicates the port's link status, speed, and flow control (FC (E) means enabled, FC(D) means disabled).

10.6 Serial LED Design Consideration and Software Customization

The unmanaged software implements the LED indication control for the front ports and system status using the Ocelot serial GPIO (SIO) controller. See the ENT-AN1187 Using the Serial GPIO/LED Controller application note for more information before designing the schematic because the SIO/LED software customization is hardware dependent.

10.7 SPI NOR Flash Support List

Starting from version 1.0.3, the system MAC address update feature in the flash is supported (see ENT-AN1186).

The following table lists the supported flash chips.

Table 5 • **Supported SPI Flash for Saving MAC Address into Flash Permanently**

Flash Port No.	Size
MX25L1606E	2 MB
MX25L25635F	32 MB
MX25V1635F	2 MB

If the flash chip is not in the support list but its density is 2 MB with 4K Bytes sector size, then the flash driver should still work. In this case, the flash driver assumes that the unknown flash device is a 2 MB density with 4K byte sector size chip and issues the flash erase or program command code based on generic 2 MB flash characteristics.