II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

MATERIALS THAT MATTER™

XMF-H User Guide

D50165-IM

T. 724.352.4455 | F. 724.352.5284 | www.ii-vi.com

II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

Table of Contents

1 Revision History 3
2 Scope 4
3 Network Setting 5
4 Firmware Upgrade for XMF-H using Web UI 7
5 SDK usage to build new application for XMF-H 10
5.1 Setup SDK Prerequisite on host machine 10
5.2 Build and Execute New application on XMF-H 10
6 Supporting Features 13
6.1 Library to include in C/C++ Applications 13
6.1.1 OTDR Library Usage 13
6.1.2 OCM Library Usage 16
6.1.3 SFP Library Usage 18
6.1.4 OCC Library Usage 20
6.1.5 Battery Management Library Usage 21
6.1.6 LED Control Library Usage 22
6.1.7 Common Interface Library Usage 22
6.2 Logging 23
6.3 Serial Application 24
6.3.1 OTDR COMMANDS EXAMPLES 24
6.3.2 OCM COMMANDS EXAMPLES 25
6.4 XMF-H Web UI 27
6.5 Freescale KL02 Flasher Application 29
6.6 Remote Firmware Upgrade 30
7. FCC Interference Statement 32

II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

1 Revision History

Revision	Date	Description
1.0	October 12th, 2020	Initial Release

II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

2 Scope

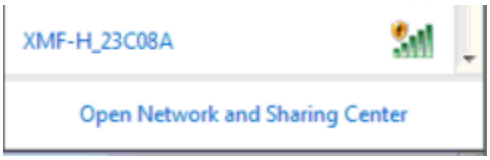
The scope of this XMF-H document is to describe:

- Procedure to configure network setting
- Procedure to upgrade firmware images
- Procedure to use SDK usage to build new application
- Procedure to use supporting features

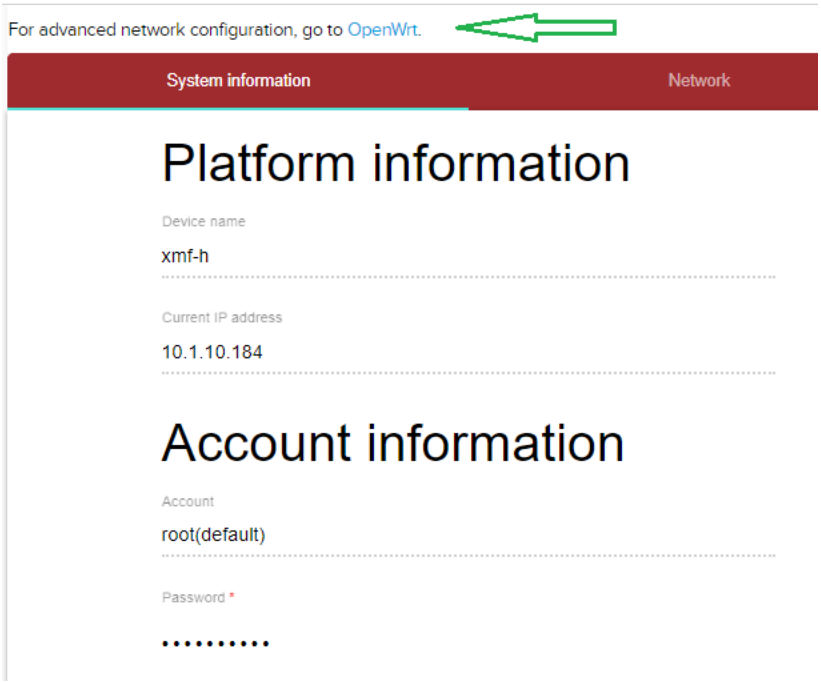
3 Network Setting

This unit is configured in AP mode when it's arrived. User may set it to Station Mode to add it to their network. This section describes the step to change the network setting to Station Mode.

1. Power up the unit and allow the boot to complete.
   Note: It takes approximately 35 seconds to boot up.
2. Click on the wireless access point and choose "XMF-H_{MAC_Address}" to connect to its WIFI.



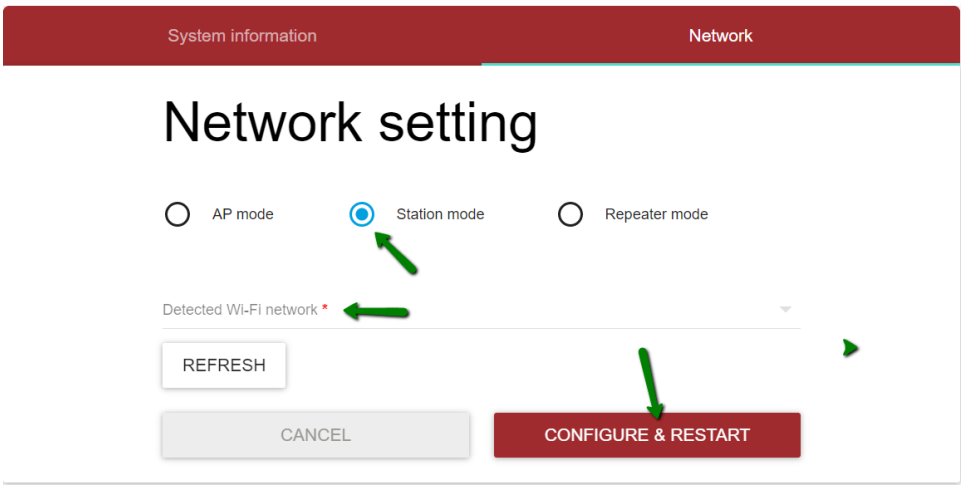
3. Open Chrome browser and enter http://192.168.100.1 to access to the GUI
   Login: root
   Password: password [Default: 8n"iUN5BiL+i]
4. User may choose to change its password by clicking the OpenWrt link.



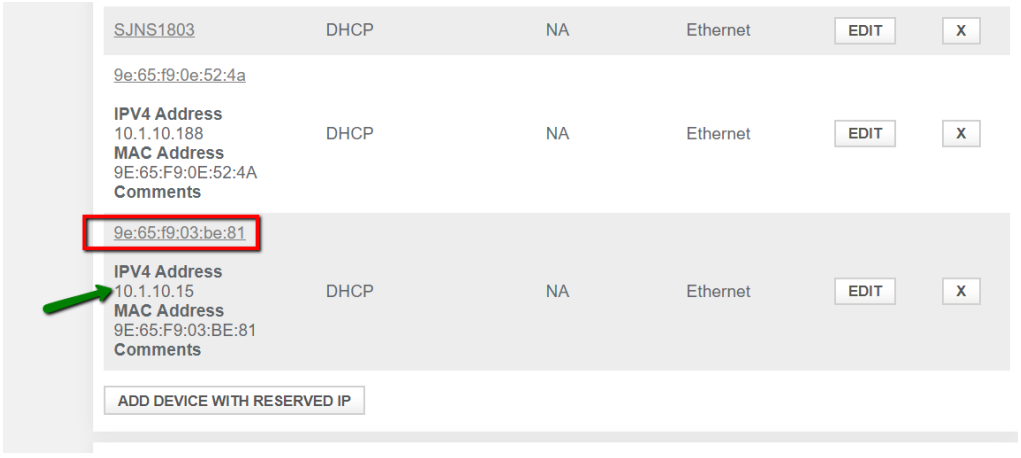
- Log in with credential in Step #3.
- Click ***System*** tab then click ***Administration*** tab.
- Set new password and confirm new password.

- Click “Save and Apply” at the lower right corner to save password.
- Click “xmf-h Web Panel”, upper right corner, to return to the main GUI window.

5. Click NETWORK SETTINGS tab.
6. Click Network tab.
7. Select Station Mode.



8. Choose the “Detected WIFI network” and enter the password (network password).
9. Click CONFIGURE & RESTART.
10. Unit will take approximately 1 minute to go into a configuration and reboot.
11. Click on the wireless access point and choose the network that the unit was added to.
12. Log in to the network router to obtain the IP address by cross referencing the unit’s MAC address. It’s should be under **Connected Device.**



13. The unit is now joined in the network.

4 Firmware Upgrade for XMF-H using Web UI

This section describes how to update the XMF-H firmware through Web UI.

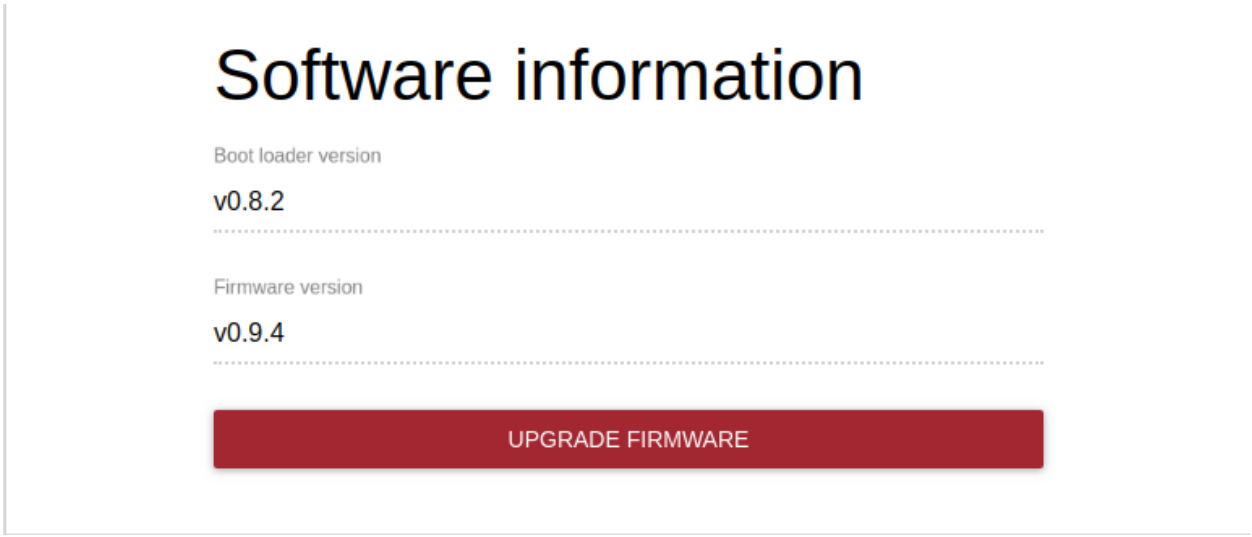
Prerequisites:

- Extract Pre-built-image.zip which contains XMF-H firmware file.
  1. lks7688.img

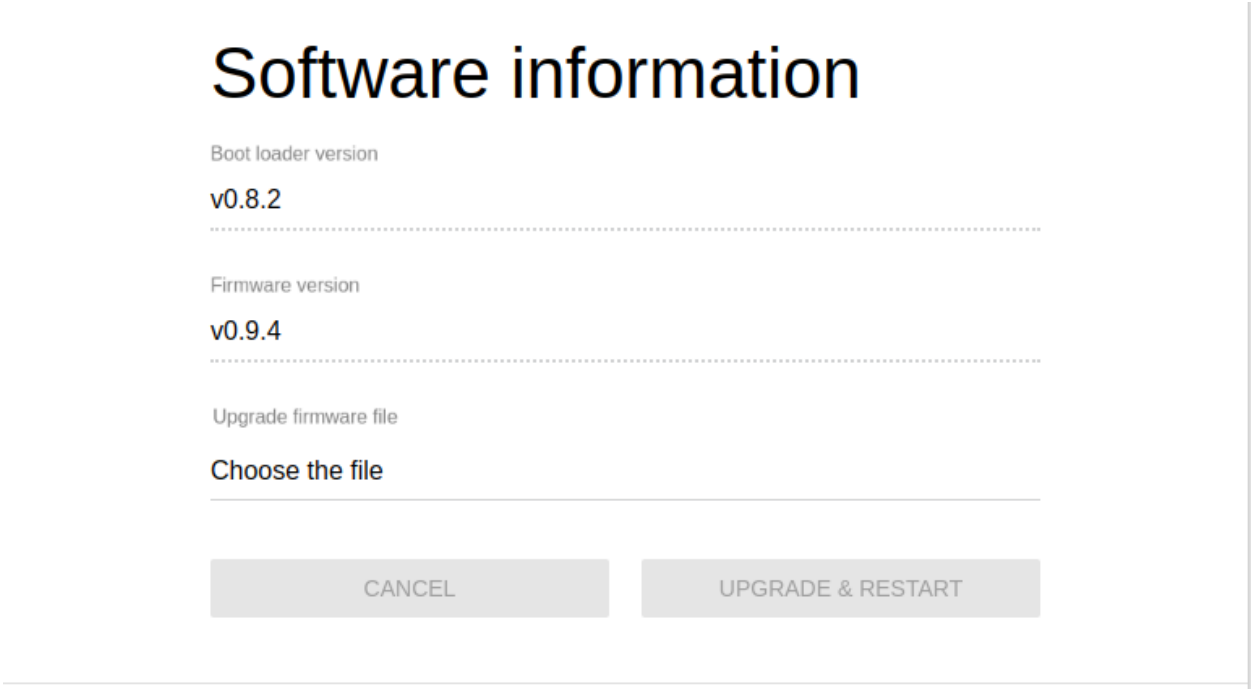
Procedure:

1. Access Web UI of XMF-H by entering http://ipaddress in chrome web browser.
2. Click “Network Settings” option on left side panel on Web page.
3. Enter Username (root) and Password, then press Sign In button.
   Note: If entering Password for the first time, Password will be configured on Sign In.

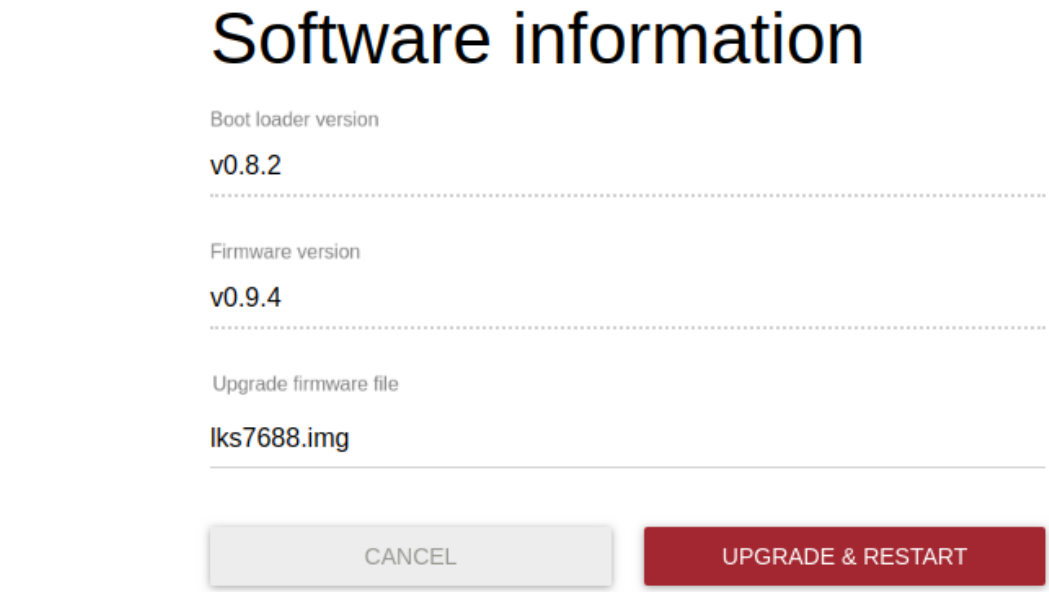
4. In the Web UI page, click Upgrade Firmware button, as shown below.



5. Click on “Choose the file” and select the firmware image file lks7688.img.

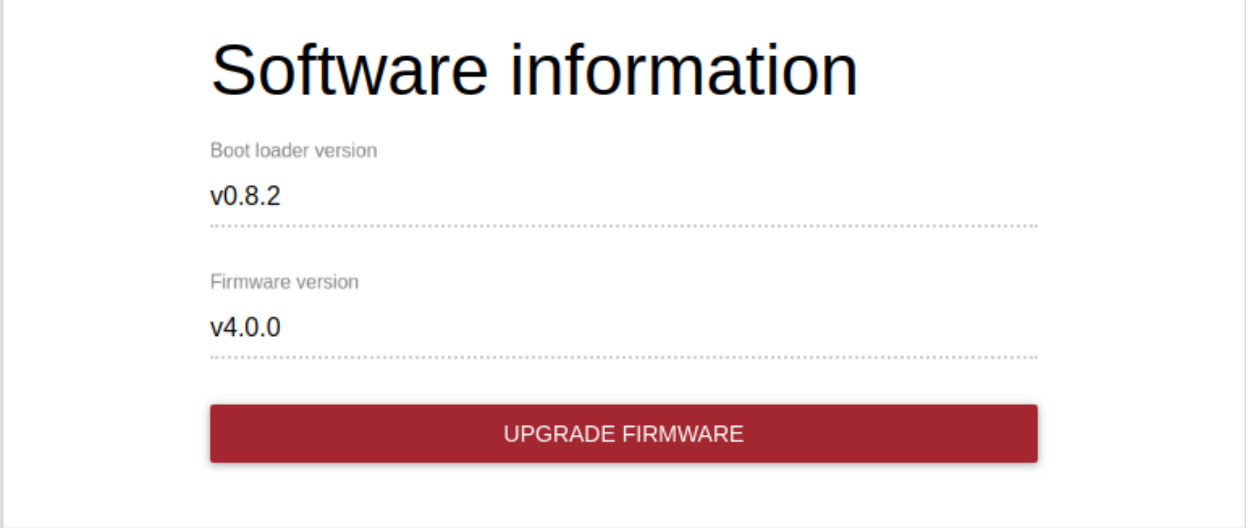


6. Click “Upgrade & Restart” button, as shown below.



7. The firmware uploads to XMF-H. The WiFi LED will start blinking fast. Do not press the PWR button or reset the board. Please make sure the board stays connected to its power source until the firmware update is complete. A message confirms that the firmware is updated, click OK.

8. **After approximately 3 minutes, the WiFi LED will light solid** to indicate that the firmware update is completed. The device will automatically reboot.
9. **After 30 seconds or more, the WiFi LED turns off**. Now scan and connect to the WiFi AP, reload the webpage, set a new password and sign in.
   The new firmware version details will be displayed in the software information, as shown below.



10. You now have the latest firmware on your XMF-H board.
11. The unit is defaulted to AP mode. Follow the steps in Section 3 to change it to Station Mode.

5 SDK usage to build new application for XMF-H

This section provides procedure to build new application for XMF-H using SDK. XMF-H_SDK provides pre-compiled MIPS toolchain designed to cross compile packages.

After building the application, user needs to transfer built application or IPK file using **SCP** to XMF-H. This transferred / installed applications will be retained over reboot.

Tasks you can do with the XMF-H_SDK:

- Compile additional applications, libraries for a specific release while ensuring feature compatibility
- Compile newer versions of certain packages for a specific release
- Recompile existing packages with custom patches or different features

Follow procedure in below section to setup and compile packages using XMF-H_SDK.

5.1 SETUP SDK PREREQUISITE ON HOST MACHINE

Prerequisite:

- Linux machine (Ubuntu 16.04 LTS)

- Install following XMF-H_SDK prerequisite on Host Linux machine.

```
Linux_Prompt$ sudo apt install subversion g++ zlib1g-dev build-essential git python python3
Linux_Prompt$ sudo apt install libncurses5-dev gawk gettext unzip file libssl-dev wget
Linux_Prompt$ sudo apt install libelf-dev ecj fastjar java-propose-classpath
```

5.2 BUILD AND EXECUTE NEW APPLICATION ON XMF-H

These steps explain process to compile application on host machine using XMF-H_SDK.

1. Unzip XMF-H_SDK.zip. It contains APP and SDK directories.
   APP directory contains sample applications.
   SDK directory has XMF-H toolchain and necessary files from provided supports in rootfs.

2. Enter SDK directory and Extract the "SDK.tar.bz2" on Host Linux machine.
   Linux_Prompt$ cd SDK
   Banner file available at this location indicates **Release version associated with this SDK**.

   Linux_Prompt$ tar -xvjf SDK.tar.bz2

3. Navigate to the directory "SDK".
   Linux_Prompt$ cd SDK

4. To compile provided example application, Copy provided sample folders inside **APP/package** to SDK package directory.

   Linux_Prompt [SDK]$ cp -r ../../**APP/package/* package/.**

   Note: SDK/package folder structure will look like:

```
SDK/package
 +helloworld            # Name of the package
   -Makefile            # This Buildroot Makefile describes the package
   +src
     -Makefile          # This GNU Makefile builds the binary
     -helloworld.c      # C/C++ source code
```

   If you create your own package for new application, create **Buildroot makefile** highlighted in above directory structure by:

   - Referring to Makefile for helloworld package here, or
   - Following instructions in the link https://openwrt.org/docs/guide-developer/packages

5. To compile helloworld package, apply following command.
   Linux_Prompt [SDK]$ make package/helloworld/compile

   NOTE:

   Prior to building the application again, clean the package using following command:

   Linux_Prompt [SDK]$ make package/helloworld/clean

   Build all applications or libraries selected using menuconfig using following command.

   Linux_Prompt [SDK]$ make

   Or

   Linux_Prompt [SDK]$ make -j5

   (You can compile faster by writing make -j5 for your build host)

6. Once it's built, navigate to “bin/ramips/packages/base” in SDK directory.
   All the ipk files will be available at this location.
   Linux_Prompt [SDK]$ cd bin/ramips/packages/base

   Note:

   Alternatively, if you DO NOT want to install individual package and just want the executable binary, visit following location:
   Linux_Prompt [SDK]$ cd staging_dir/target-mipsel_24kec+dsp_uClibc-0.9.33.2/root-

II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

ramips/bin

7. Find a package file named helloworld_x.x.x-x_ramips_24kec.ipk

8. Transfer the helloworld_x.x.x-x_ramips_24kec.ipk file from the host Linux machine to the XMF-H board using **SCP.**
   Linux_Prompt [SDK/bin/ramips/packages/base]$ scp helloworld_x.x.x-x_ramips_24kec.ipk root@ipaddress:

   Execute following steps on XMF-H board.

9. In the system console of the XMF-H board, navigate to the location of the .ipk file and apply command
   root@xmf-h:~# opkg install helloworld_x.x.x-x_ramips_24kec.ipk

10. After the installation is complete, type helloworld and you'll see the output of binary on XMF-H
    root@xmf-h:~# helloworld

11. Enter following command to remove any installed application / module.
    root@xmf-h:~# opkg remove helloworld

6 Supporting Features

6.1 LIBRARY TO INCLUDE IN C/C++ APPLICATIONS

Interface libraries are created for OTDR, OCM, SFP, OCC operations with underlying UART / I2C communications. If user needs to create any new C/C++ applications to communicate with these instruments, mentioned libraries can be linked.

6.1.1 OTDR Library Usage

1. User needs to include the header file of the library into the application. For OTDR, It will be ***<otdr_interface.h>*** file.
2. After including the header file user needs to initialize the library by calling the ***OtdrInterfaceInit()*** function. API returns 0 on success and error code on failure.

   [Once Init is successful, multiple APIs can be called before deinit.]
3. Call the relevant API from following list in **OTDR Interface Table** to perform operation on OTDR serial interface. This functions expects different arguments. otdr_interface.h describes all OTDR functions operations, input & output parameters.

   API returns 0 on success, -1 if OtdrInterfaceInit not performed / failed and relevant error code on other failures.
4. De-initialize the library by calling the ***OtdrInterfaceDeInit().*** API returns 0 on success and error code on failure.
5. Link the library while compiling the application by ***"-lotdrinterface"***.

OTDR Fast Scan Feature:

OTDR Emulator is developed in order to obtain fast response from OTDR. Limited commands are supported currently, as mentioned in Table below. OTDR_EMU is initiated at init time.

Emulator can be started and stopped by following commands, in case required:

root@xmf-h:~# /etc/init.d/otdr_emu.sh start

root@xmf-h:~# /etc/init.d/otdr_emu.sh stop

II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

OTDR Interface Table:

Sr No.	Command	API	Remarks	Supported for Fast Scan Feature?
1	*Init - Custom API*	int OtdrInterfaceInit(T_OTDRCOMMMODE mode);	T_OTDRCOMMMODE Structure defined in otdr_interface.h	Yes
2	*DeInit - Custom API*	int OtdrInterfaceDeInit(T_OTDRCOMMMODE mode);	T_OTDRCOMMMODE Structure defined in otdr_interface.h	Yes
3	ver	int OtdrGetVer(T_VER *ver);	T_VER Structure defined in otdr_interface.h	Yes
4	mt	int OtdrGetMt(float *mt);		Yes
5	otdr	int OtdrGetParameters(T_OTDRCONFIG *params);	T_OTDRCONFIG Structure defined in otdr_interface.h	Yes
6	otdr pulse width x	int OtdrSetPulseWidth(float pw);		Yes
7	otdr measure time x	int OtdrSetMeasureTime(float mtime);		Yes
8	otdr window size x	int OtdrSetWindowSize(float ws);		Yes
9	otdr window offset x	int OtdrSetWindowOffset(float wo);		Yes
10	otdr span length x	int OtdrSetSpanLength(float sl);		Yes
11	otdr resolution x	int OtdrSetResolution(float res);		Yes
12	otdr event loss thresold x	int OtdrSetEventLossThreshold(float thr);		Yes
13	pd	int OtdrGetPd(float pd[]);		No
14	alrm	int OtdrGetALRM(T_ALRM *alrm);	T_ALRM Structure defined in otdr_interface.h	No
15	otdr run	int OtdrRun(void)		Yes
16	otdr status	int OtdrStatus(T_OTDRSTATUS *status);	T_OTDRSTATUS Structure defined in otdr_interface.h	Yes
17	otdr dump trace	int OtdrDumpTrace(T_DUMPTRACE *dumpTrace);	T_DUMPTRACE Structure defined in otdr_interface.h	Yes
18	otdr dump event	int OtdrDumpEvent(T_DUMPEVENT dumpEvent, int *event_length)	T_DUMPEVENT Structure defined in otdr_interface.h	Yes
19	otdr stop	int OtdrStop(void)		Yes

II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

20	VER HW	int OtdrGetHwVer(T_HWVER *verHw)	T_HWVER Structure defined in otdr_interface.h	No
21	ALRM x HYS y	int OtdrSetAlrmHysteresis(T_OTDRALARMS alarm, int hysteresis);	T_OTDRALARMS Structure defined in otdr_interface.h	No
22	ALRM x CLR	int OtdrClearAlarmLatch(T_OTDRALARMS alarm);	T_OTDRALARMS Structure defined in otdr_interface.h	No
23	AST	int OtdrGetAst(T_AST *ast)	T_AST Structure defined in otdr_interface.h	No
24	ASTM N	int OtdrSetAstmNormal(void)		No
25	ASTM S	int OtdrSetAstmSticky(void)		No
26	ASTM	int OtdrGetAstm(int *mode)		No
27	RST	int OtdrFactoryReset(void);		No
28	BOOT	int OtdrReboot(void)		No
29	RTC	int OtdrGetRtc(char *time, char *date);		No
30	SUPPLY	int OtdrGetSupply(T_SUPPLY *supply);	T_SUPPLY Structure defined in otdr_interface.h	No
31	OTDR EVENT_REFL_THR x	int OtdrSetEventReflectionThreshold(float thr);		Yes
32	OTDR ANALYZE	int OtdrAnalyze(void)		Yes
33	ALRM x THR y	int OtdrSetAlrmThreshold(T_OTDRALARMS alarm, int threshold);	T_OTDRALARMS Structure defined in otdr_interface.h	No
34	RTC x y	int OtdrSetRtc(char *time, char *date);		No
35	otdr dump ltrace	int OtdrDumpTraceLinear(T_DUMPTRACE *dumpTrace);	T_DUMPTRACE Structure defined in otdr_interface.h	Yes
36	otdr result	int OtdrResult(T_OTDRRESULT *result)	T_OTDRRESULT Structure defined in otdr_interface.h	Yes
37	ver full	int OtdrGetVerFull(T_VER_FULL *ver_full)	T_VER_FULL Structure defined in otdr_interface.h	No
38	OTDR Power Operation	int OTDRPower(E_OTDRPOWERSWITCH i_e_switch)	Structure defined in otdr_interface.h. typedef enum { OTDR_OFF = 0, OTDR_ON }E_OTDRPOWERSWITCH;	Yes
39	Get OTDR Power Status	int getOTDRStatus()	**Possible values:** OTDR_DEVICE_ON,	Yes

			OTDR_DEVICE_OFF, OTDR_POWERING_UP, OTDR_POWERING_OFF If there is issue with Device on XMFH, OFF to ON transition will indicate status sequence as: OTDR_DEVICE_OFF > OTDR_POWERING_UP > Post 6-7 Sec timeout > OTDR_DEVICE_OFF	

6.1.2 OCM Library Usage

1. User needs to include the header file of the library into the application. For OCM, It will be ***<ocm_interface.h>*** file.
2. After including the header file user needs to initialize the library by calling the ***OcmInterfaceInit()*** function. API returns 0 on success and error code on failure.

   [Once Init is successful, multiple APIs can be called before deinit.]
3. Call the relevant API from following list in **OCM Interface Table** to perform operation on OCM serial interface. This functions expects different arguments. ocm_interface.h describes all OCM functions operations, input & output parameters.

   API returns 0 on success, -1 if OcmInterfaceInit not performed / failed and relevant error code on other failures.
4. De-initialize the library by calling the ***OcmInterfaceDeInit().*** API returns 0 on success and error code on failure.
5. Link the library while compiling the application by ***"-locminterface"***.

OCM Interface Table:

Sr No.	Command	API	Remarks
1	*Init - Custom API*	int OcmInterfaceInit(void)	
2	*DeInit - Custom API*	int OcmInterfaceDeInit(void)	
3	DREV	int OcmGetDREV(char *revision)	

II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

4	CSSP	int OcmGetCSSP(uint32_t *channalPlan, float *startingFreq)	T_CSSP Structure defined in ocm_interface.h
5	CSSPx	int OcmSetCSSP(uint32_t *channalPlan, float *startingFreq)	
6	CBDI	int OcmGetCBDI(T_CBDI *cbdi)	T_CBDI Structure defined in ocm_interface.h
7	CGTP1B	int OcmGetCGTP1B(T_INTEGRATE integration, T_CGTP1B *cgtp1b)	T_CGTP1 Structure defined in ocm_interface.h
8	FWVR	int OcmGetFWVR(char *fwvr)	
9	HWVR	int OcmGetHWVR(char *hwvr)	
10	SNUM	int OcmGetSNUM(char *idCode, char *manufactureDate, int *buildNumber)	
11	CENB1	int OcmDoCENB1(T_TOFSCAN_MODE scanMode, uint32_t *scanSetting)	T_TOFSCAN_MODE Structure defined in ocm_interface.h
12	CSTS0	int OcmGetCSTS0(u_int32_t *ctmStatus)	
13	CSTA0	int OcmGetCSTA0(T_CSTA0 *csta1)	T_CSTA0 Structure defined in ocm_interface.h
14	CTPR1	int OcmGetCTPR1(float *totPower)	
15	CGSP1	int OcmGetCGSP1(T_SLICE_WIDTH sliceWidth, T_SPECTRUM_FORMAT spectrumFormat, T_EDGE_BINS edgeBins, T_CGSP1 *cgsp1)	T_SLICE_WIDTH, T_SPECTRUM_FORMAT, T_EDGE_BINS & T_CGSP1 Structures defined in ocm_interface.h
16	CFCR	int OcmGetCFCR(T_CFCR *cfcr)	T_CFCR Structure defined in ocm_interface.h
17	CDDFB	int OcmGetCDDFB(T_CDDFB *cddfb)	T_CDDFB Structure defined in ocm_interface.h
18	CGET1	int OcmGetCGET1(T_INTEGRATE integration, T_CGET1 *cget1)	T_INTEGRATE & T_CGET1 Structure defined in ocm_interface.h
19	CRST	int OcmGetCRST(void)	
20	CDMO	int OcmSetCDMO(bool enableDemoMode)	
21	GNHI1/GNLO1	int OcmSetGain(bool highGain)	

22	CMPP1	int OcmSetCMPP1(float minimumPeakPower)	
23	CSIL1	int OcmSetCSIL1(float inserttionLoss)	
24	CSPT1	int OcmSetCSPT1(float peakDetectThreshold)	
25	OCM Power Operation	int OCMPower(E_POWERSWITCH i_e_switch)	E_POWERSWITCH Structure defined in ocm_interface.h
26	Get OCM Power Status	int getOCMStatus(void)	**Possible values:** OCM_DEVICE_ON, OCM_DEVICE_OFF, OCM_DEVICE_POWERING_UP, OCM_DEVICE_POWERING_OFF

6.1.3 SFP Library Usage

1. User needs to include the header file of the library into the application. For SFP, It will be ***<sfp_interface.h>*** file.
2. After including the header file user needs to initialize the library by calling the ***SFPInterfaceInit()*** function. API returns 0 on success and error code on failure.

   [Once Init is successful, multiple APIs can be called before deinit.]
3. Call the relevant API from following list in **SFP Interface Table** to perform operation on SFP I2C interface. This functions expects different arguments. sfp_interface.h describes all SFP functions operations, input & output parameters.

   API returns 0 on success, -1 if SFPInterfaceInit not performed / failed and relevant error code on other failures.
4. De-initialize the library by calling the ***SFPInterfaceDeInit().*** API returns 0 on success and error code on failure.
5. Link the library while compiling the application by ***"-li2cinterface"***.

SFP Interface Table:

Sr No.	Command	API	Remarks

1	*Init - Custom API*	int SFPInterfaceInit(void)	
2	*DeInit - Custom API*	int SFPInterfaceDeInit(void)	
3	Read Byte	int SFPReadByte(uint8_t i_ui8_deviceAddress, uint8_t i_ui8_registerAddress, uint8_t* o_ui8_data)	
4	Read Word	int SFPReadWord(uint8_t i_ui8_deviceAddress, uint8_t i_ui8_registerAddress, uint16_t* o_ui16_data)	
5	Write Byte	int SFPWriteByte(uint8_t i_ui8_deviceAddress, uint8_t i_ui8_registerAddress, uint8_t i_ui8_data)	
6	Write Word	int SFPWriteWord(uint8_t i_ui8_deviceAddress, uint8_t i_ui8_registerAddress, uint16_t i_ui16_data)	
7	Page Read	int SFPPageRead(uint8_t i_ui8_deviceAddress, uint8_t i_ui8_registerAddress, uint16_t i_ui16_n, uint8_t *o_ui8_data)	SFPPageRead API performs read operation for maximum 63 Bytes chunk due to MRAA library limitation.
8	Page Write	int SFPPageWrite(uint8_t i_ui8_deviceAddress, uint8_t i_ui8_registerAddress, uint16_t i_ui16_n, const uint8_t* i_ui8_data)	SFPPageWrite API performs write operation for maximum 8 Bytes chunk.
9	Read MOD ABS	int SFPReadModABS(bool *o_b_mode)	
10	Read Tx Fault	int SFPReadTxFault(bool *o_b_fault)	
11	Read LOS	int SFPReadLOS(bool *o_b_los)	
12	Read Tx Disable	int SFPReadTxDisable(bool *o_b_tx_disable)	
13	Set Tx Disable value	int SFPSetTxDisable(bool i_b_status)	Pass 1 to set TxDisable thus disable Tx laser; pass 0 to clear TxDisable thus enable Tx laser.
14	SFP Power Operation	int SFPPower(E_SFPPOWERSWITCH i_e_switch)	
15	Get SFP Power Status	int getSFPStatus(void)	**Possible values:** SFP_DEVICE_ON, SFP_DEVICE_OFF, SFP_DEVICE_POWERING_UP, SFP_DEVICE_POWERING_OFF

			If there is issue with Device on XMFH, OFF to ON transition will indicate status sequence as: SFP_DEVICE_OFF > SFP_DEVICE_POWERING_UP > Post 3-4 Sec timeout > SFP_DEVICE_OFF

6.1.4 OCC Library Usage

1. User needs to include the header file of the library into the application. For OCC, It will be ***<occ_interface.h>*** file.
2. After including the header file user needs to initialize the library by calling the ***OCCInterfaceInit()*** function. API returns 0 on success and error code on failure.

   [Once Init is successful, multiple APIs can be called before deinit.]
3. Call the relevant API from following list in **OCC Interface Table** to perform operation on OCC I2C interface. This functions expects different arguments. occ_interface.h describes all OCC functions operations, input & output parameters.

   API returns 0 on success, -1 if OCCInterfaceInit not performed / failed and relevant error code on other failures.
4. De-initialize the library by calling the ***OCCInterfaceDeInit().*** API returns 0 on success and error code on failure.
5. Link the library while compiling the application by ***"-li2cinterface"***.

OCC Interface Table:

Sr No.	Command	API	Remarks
1	*Init - Custom API*	int OCCInterfaceInit(void)	
2	*DeInit - Custom API*	int OCCInterfaceDeInit(void)	

3	Read Power	int OCCReadPWR(int i_i_pdNum, float *o_f_pwrVal)	typedef enum { PD_1577=0, PD_1550, PD_1270, PD_4, PD_NONE } E_PDNUM; XMF-H EEPROM should be flashed with OCC calibration values. If it is not, API will provide uncalibrated Power value and return value will indicate Error.

6.1.5 Battery Management Library Usage

1. User needs to include the header file ***<battery_management.h>*** of the library into the application source code.
2. User needs to initialize the library by calling the ***BatteryManagementInit()*** function. *[Once Init is successful, multiple APIs can be called before deinit.]*
3. Call the relevant API from following list in **Battery management Interface Table** to perform operation on Power management I2C interface.
4. De-initialize the library by calling the ***BatteryManagementDeInit().***
5. Link the library while compiling the application by ***"-lbatterymanagement"***.

Battery management Interface Table:

Sr No.	Command	API	Remarks
1	Init - Custom API	int BatteryAPIInit(void)	
2	DeInit - Custom API	int BatteryAPIDeInit(void)	
3	Get Battery Charger Temperature	int GetBatteryChargerTemp(float *o_f_temp)	Temperature in °C
4	Get Battery Voltage	int GetBatteryVoltage(float *o_f_voltage)	Battery voltage in Volts
5	Get Battery Current	int GetBatteryCurrent(float *o_f_current)	Battery current in A
6	Get Battery Charge Percentage	int GetBatteryChargePercentage(float *o_f_chargePercentage)	Predicted remaining battery capacity as a percentage of FullChargeCapacity
7	Get Battery Charge Voltage	int GetBatteryChargerVoltage(float *o_f_chargeVoltage)	Charging voltage in Volts

6.1.6 LED Control Library Usage

1. User needs to include the header file ***<led_control.h>*** of the library into the application source code.
2. User needs to initialize the library by calling the ***LEDInit()*** function.
   [Once Init is successful, multiple APIs can be called before deinit.]
3. Call the relevant API from following list in **LED Control Interface Table** to perform operation on LED I2C interface.
4. De-initialize the library by calling the ***LEDDeInit().***
5. Link the library while compiling the application by ***"-lledinterface"***.

LED Control Interface Table:

Sr No.	Command	API	Remarks
1	Init - Custom API	int LEDInit(void)	
2	DeInit - Custom API	void LEDDeInit(void)	
3	Enable Controller	int LEDEnableController(bool i_b_state)	
4	LED Brightness	int LEDBrightness(E_LEDID i_e_ledId, uint8_t i_ui8_colorRed, uint8_t i_ui8_colorGreen, uint8_t i_ui8_colorBlue)	typedef enum { LED_ID_ALM=0x01, LED_ID_WIFI=0x02, LED_ID_PWR=0x04, LED_ID_BAT=0x08, LED_ID_ALL=0x0F } E_LEDID; Valid range for brightness: 0-128
5	Reset Power Control Logic	int ResetPowerContolLogic(void)	

6.1.7 Common Interface Library Usage

1. User needs to include the header file of the library into the application. For Common interface, It will be ***<common_i2c_operation.h>*** file.
2. Call the relevant API from following list in **Common Interface Table** to perform operation on I2C interface. This functions expects different arguments. common_i2c_operation.h describes all functions operations, input & output parameters.

3. Link the library while compiling the application by ***"-li2cinterface"***.

Common Interface Table:

Sr No.	Command	API	Remarks

1	Get Diagnostics Data	int GetDiagnosticsData (T_DIAG *diag)	typedef struct _diag
			{ uint8_t conf_reg; // Configuration Register
			uint8_t mfg_id; // Manufacturing ID
			uint8_t rev_id; // Revision ID
			uint8_t conv_rate; // Conversion Rate
			uint8_t ch_disable_reg; // Channel Disable
			uint8_t status_reg; // Status Register
			float temp; // Temperature
			float channel[7]; // channels
			} T_DIAG;
			
			/* Diagnostics Parameters */
			/* Channel array contains values as mentioned below:
			Channel[0]: TIA OUT
			Channel[1]: Ref Voltage
			Channel[2]: Current_PL
			Channel[3]: +3v3
			Channel[4]: VAdaptor
			Channel[5]: Pack_Load
			Channel[6]: +5VUSB
			*/

6.2 LOGGING

Log files are generated at location "/var/log". Instrument specific files are generated ex. OTDR.log, OCM.log.

These log files maintain entries for Interface Library API calls and additional logged error points, useful processed data from Instruments.

Roll over mechanism keeps latest 2 log files for each instrument with maximum 5 MB size for each file.

II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

6.3 SERIAL APPLICATION

Serial Application is intended to use for direct communication over serial interface without much processing. It can help to automate communication using script by calling application along with required arguments.
Serial application can be executed by “serialapp” command on XMF-H terminal prompt.
Sample commands and output are displayed below.

Command Format:- cmd-a == ASCII
Command Format:- cmd-x == HEX

Response Format:- resp-a == ASCII
Response Format:- resp-x == HEX

root@xmf-h:~# serialapp
USAGE : serialapp serial-port-no cmd-a/x resp-a/x command
Example:
serialapp 0 cmd-a resp-a otdr measure_time%0d%0a
serialapp 1 cmd-a resp-x CBDI%0d%0a

6.3.1 OTDR COMMANDS EXAMPLES

root@xmf-h:~# serialapp 0 cmd-a resp-a ver%0d%0a
ver
Configuration: IIVI_OTDR
Firmware Vers: IIVI_SUB_OTDR_0.5.41B
Serial Number:
root@xmf-h:~# serialapp 0 cmd-a resp-a echo%0d%0a
echo
ECHO: ON
root@xmf-h:~# serialapp 0 cmd-a resp-a echo on%0d%0a
echo on
root@xmf-h:~# serialapp 0 cmd-a resp-a otdr measure_time%0d%0a
otdr measure_time
OTDR MEASURE_TIME: 1.0 s
root@xmf-h:~# serialapp 0 cmd-a resp-a otdr run%0d%0a
otdr run
root@xmf-h:~# serialapp 0 cmd-a resp-a otdr status%0d%0a

II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

otdr status
OTDR STATUS: IDLE
root@xmf-h:~# serialapp 0 cmd-a resp-a otdr dump trace%0d%0a
otdr dump trace
km,dB
0.0000,-17.50
0.0040,-17.50
0.0080,-17.33
0.0120,-17.10
0.0161,-17.59
0.0201,-40.00
0.0241,-23.23
…
…
...
34.9839,-40.00
34.9880,-40.00
34.9920,-40.00
34.9960,-40.00
root@xmf-h:~# serialapp 0 cmd-a resp-a otdr dump event%0d%0a
otdr dump event
type,x_km,loss_u_km,refl_u_km,loss_dB,refl_dB,attn_dB/km
3,0.0000,0.1406,0.0618,-3.73,-42.66,0.191
4,24.4855,NaN,0.0618,3.85,-53.99,0.191

6.3.2 OCM COMMANDS EXAMPLES

root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-a DREV%0d%0a
Rev CTM1370 1.5.0, Compiled on Apr 03 2019 16:04:00 Release: 54 IMAGE CRC:36061
root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-a SNUM%0d%0a
LB AGSC 20190716606
root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-a FWVR%0d%0a
01.05
root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-a HWVR%0d%0a
00.00
root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-x CBDI%0d%0a
75 42 ca 01 52 4c d7 83 03 00 00 00 00 05 01 29 eb 67 0c 19 d8 09 36 18 06 00 00 00 34 33 44 43 34 73
3f 43 00 00 80 3f 2f 6e a3 3c 05 24 81 de 5b ed 3d 68 52 84 38 00 00 00 00 00 00 00 00 ff ff ff ff 49 49 2d

56 49 20 50 68 6f 74 6f 6e 69 63 73 20 20 20 20 20 4c 42 20 41 47 53 43 20 32 30 31 39 30 37 31 36 36
30 36 20 20 20 20 20 20 20 20 20 20 20 20 20 2c 0e

root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-x CGTP1%0d%0a
0a 0a 00 00 00 00 10 03 00 00 00 00 00 00 00 00 00 00 00 35 cc 0e 37 ce b7 d5 34 34 33 44 43 a8 8d
d6 34 67 26 44 43 7b 87 d6 34 9a 19 44 43 f0 04 d6 34 cd 0c 44 43 15 e8 d5 34 01 00 44 43 20 50 d6 34
34 f3 43 43 2f 97 d5 34 67 e6 43 43 c9 87 d5 34 9a d9 43 43 12 cb d5 34 cd cc 43 43 28 fd d4 34 01 c0 43
43 a9 a5 d4 34 34 b3 43 43 38 78 d5 34 67 a6 43 43 74 53 d5 34 9a 99 43 43 50 bc d5 34 cd 8c 43 43 87
18 d6 34 01 80 43 43 8e dd d5 34 34 73 43 43 da b2 d5 34 67 66 43 43 6e 18 d6 34 9a 59 43 43 ed 3d 07
37 cd 4c 43 43 a1 d3 d5 34 01 40 43 43 02 30 d6 34 34 33 43 43 1e 49 d6 34 67 26 43 43 16 5d d5 34 9a
19 43 43 32 d3 d4 34 cd 0c 43 43 d4 34 01 00 43 43 ba 01 d4 34 34 f3 42 43 d8 4a d4 34 67 e6 42 43 d3
c8 d4 34 9a d9 42 43 76 86 d4 34 cd cc 42 43 03 5b d4 34 01 c0 42 43 87 d0 d4 34 34 b3 42 43 18 8e d4
34 67 a6 42 43 05 a8 d4 34 9a 99 42 43 60 15 d5 34 cd 8c 42 43 ec f4 d4 34 01 80 42 43 9c 71 d5 34 34
73 42 43 45 85 d6 34 67 66 42 43 da f5 d6 34 9a 59 42 43 a2 00 d7 34 cd 4c 42 43 d1 73 d8 34 01 40 42
43 2f 6f d9 34 34 33 42 43 12 9a d9 34 67 26 42 43 36 da da 34 9a 19 42 43 bf db 34 cd 0c 42 43 5f c8 db
34 01 00 42 43 d0 ed dc 34 34 f3 41 43 73 5f dd 34 67 e6 41 43 e8 af dc 34 9a d9 41 43 1b cf dc 34 cd cc
41 43 7d 56 dd 34 01 c0 41 43 c7 36 dc 34 34 b3 41 43 ef 95 db 34 67 a6 41 43 73 b9 db 34 9a 99 41 43
40 cf da 34 cd 8c 41 43 6a 28 da 34 01 80 41 43 5a 6d da 34 34 73 41 43 78 83 d9 34 67 66 41 43 94 ee
d8 34 9a 59 41 43 42 84 d9 34 cd 4c 41 43 68 cc d8 34 01 40 41 43 27 ec d7 34 34 33 41 43 3f ba d8 34
67 26 41 43 0a 2c d9 34 9a 19 41 43 88 ad d8 34 cd 0c 41 43 d0 90 d9 34 01 00 41 43 0c c2 d9 34 34 f3
40 43 9c 1c d9 34 67 e6 40 43 15 f6 d9 34 9a d9 40 43 38 74 d9 34 cd cc 40 43 e6 90 d8 34 01 c0 40 43 6d
6f d9 34 34 b3 40 43 64 26 d9 34 67 a6 40 43 52 d1 d7 34 9a 99 40 43 1d a8 d7 34 cd 8c 40 43 1b 5d d8
34 01 80 40 43 68 fb d7 34 34 73 40 43 9b 62 d8 34 67 66 40 43 f0 11 d9 34 9a 59 40 43 6a 68 d8 34 cd
4c 40 43 60 58 d9 34 01 40 40 43 e7 df d9 34 34 33 40 43 cd de d8 34 67 26 40 43 8f db d9 34 9a 19 40
43 47 4b da 34 cd 0c 40 43 8a 7e da 34 01 00 40 43 06 6f da 34 34 f3 3f 43 f2 ff d9 34 67 e6 3f 43 a7 7d
d9 34 9a d9 3f 43 97 2d d9 34 cd cc 3f 43 32 8c d9 34 01 c0 3f 43 84 34 d9 34 34 b3 3f 43 41 24 d9 34 67
a6 3f 43 28 02 da 34 9a 99 3f 43 48 9b d9 34 cd 8c 3f 43 c9 7d da 34 01 80 3f 43 f7 7d da 34 34 73 3f 43
ef e8 55 55

root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-x CSSP%0d%0a
0a 0a 00 00 00 00 08 00 00 00 00 00 34 33 44 43 8d 46 55 55
root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-x CSSP0%0d%0a
0a 0a 00 00 00 00 08 00 00 00 00 00 34 33 44 43 8d 46 55 55
root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-x CSSP1%0d%0a
0a 0a 00 00 00 00 08 00 01 00 00 00 34 33 44 43 dc ec 55 55
root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-x CSSP2%0d%0a
0a 0a 00 00 00 00 08 00 02 00 00 00 67 26 44 43 1e 94 55 55
root@xmf-h:/IoT/examples# serialapp 1 cmd-a resp-x CSTA1%0d%0a
0a 0a 00 00 00 00 08 00 00 b5 5a 42 00 00 00 00 b3 8c 55 55
root@xmf-h:/IoT/examples#

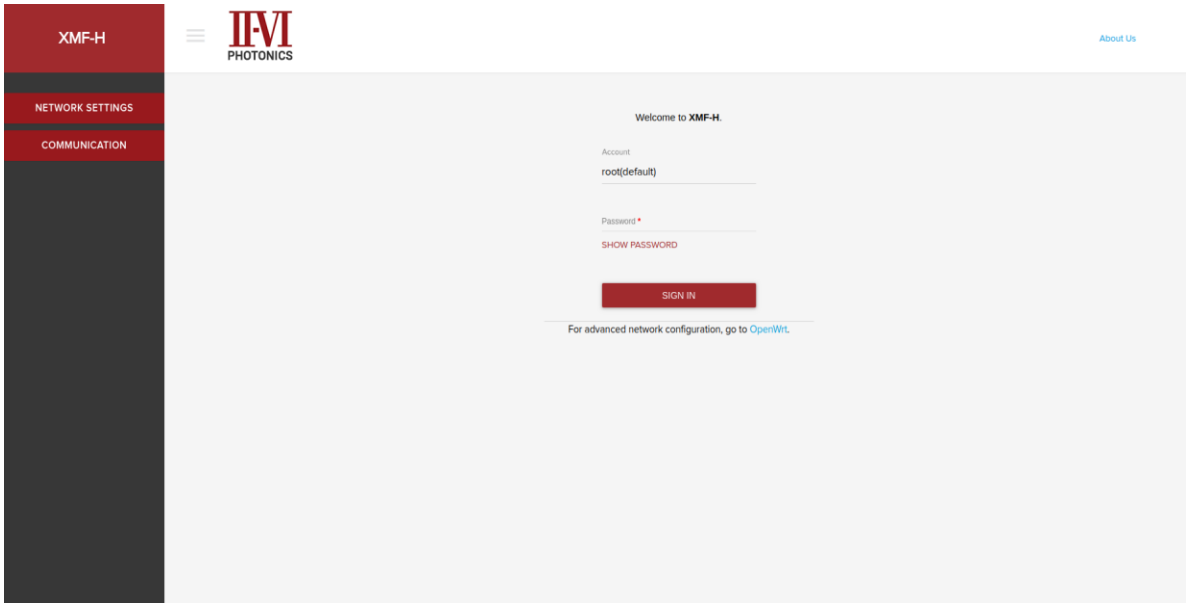
6.4 XMF-H WEB UI

This section provides steps to use Web UI for communication with instruments of XMF-H.

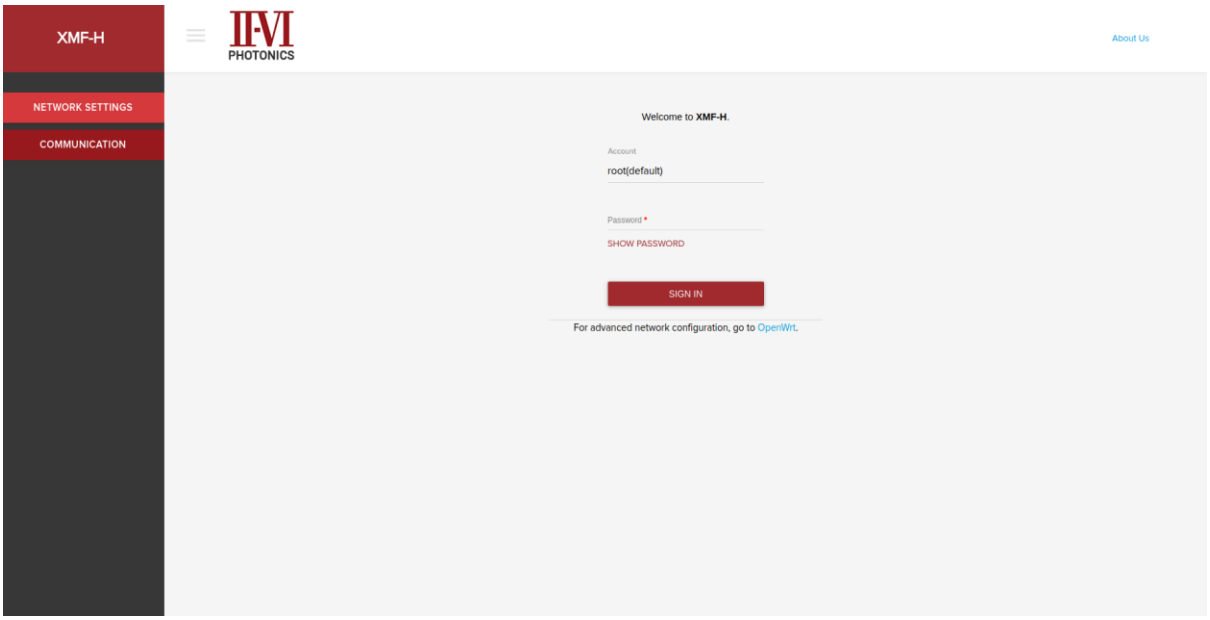
1. To open Web UI, open chrome browser. Then, enter URL **http://IP-ADDRESS**
   - Ex: http://10.1.10.184 (if in Station mode) or
     Ex: http://xmf-h.local or http://192.168.100.1 (if in AP mode)

     Find out IP address from System information tab as shown in **SECTION 3** or by issuing "ifconfig" command on Serial interface console.

   The Web UI appears as shown below:

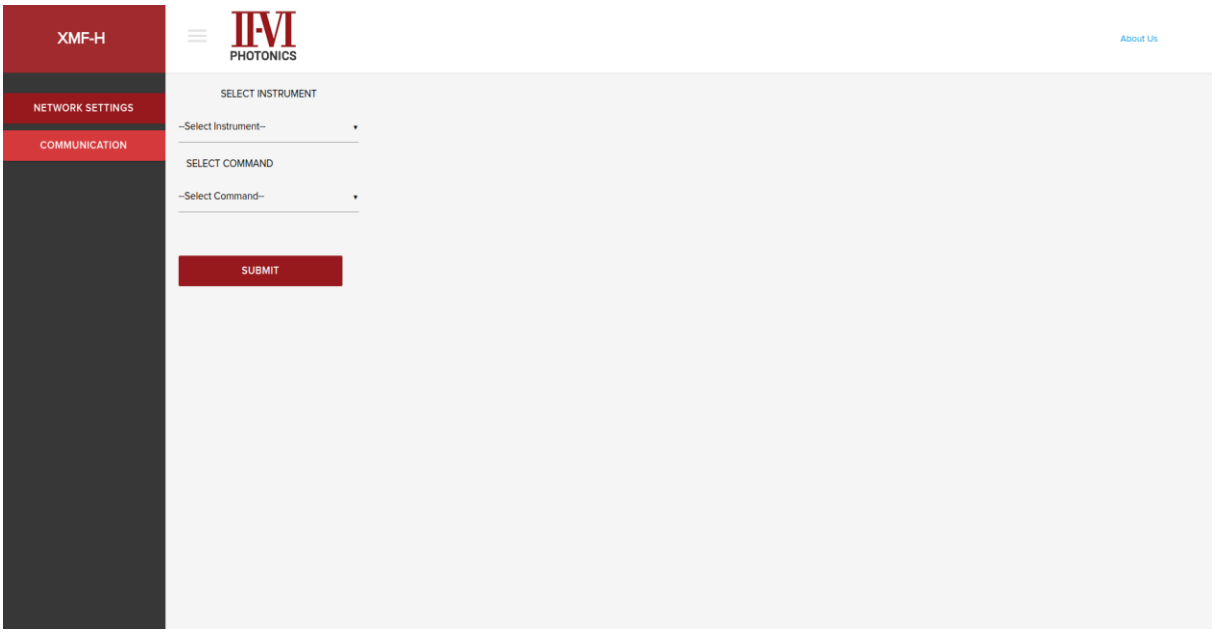


2. Click on "**Network Settings"** from left side navigation menu to configure the network related settings of your board as shown below

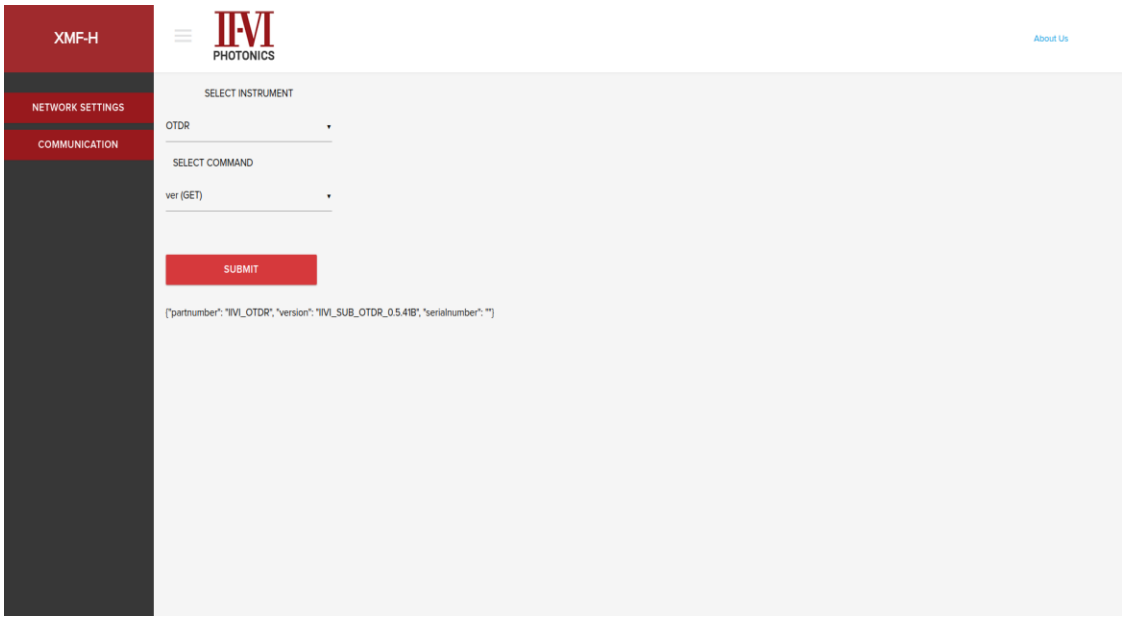


- If you are accessing Web UI at very first time after firmware flash, set a password using at least six alphanumeric characters and click **SIGN IN**.
- Enter the password again and click **SIGN IN**.
- You're now signed into the board's Web UI. You can move on to flash the latest firmware (refer **SECTION 4**) or to configure network settings (refer **SECTION 3**).

3. Click on "**COMMUNICATION"** from left side navigation menu to issue command and get related responses of your XMF-H board as shown below.



- Select the **OTDR** from the instrument dropdown box. Then select (otdr_get_params, ver, OTDR_ON, OTDR_OFF etc.) command from command dropdown box as shown below:



  - Click on **Submit** Button.
  - You will now see response of selected command.
- Select the **OSA** from the instrument dropdown box. Then select any from (PWR_off, CTM_reset, PWR_on etc.) command from command dropdown box.
  - Click on **Submit** Button.
- Select the **OCC** from the instrument dropdown box. Then select (OCC) command from command dropdown box and click on **Submit** Button.
- Select the **SFP** from the instrument dropdown box. Then select (SFP) command from command dropdown box and click on **Submit** Button.
- Select the **OTHER** from the instrument dropdown box. Then select (ADC128D181) command from command dropdown box and click on **Submit** Button.

6.5 FREESCALE KL02 FLASHER APPLICATION

Freescale KL02 is setup by init script on XMFH Boot time. In order to upgrade Freescale firmware, flasher application is included. It halts the execution, erases the chip, writes new firmware and resets Freescale KL02. Issue following command with firmware bin file name to upgrade.

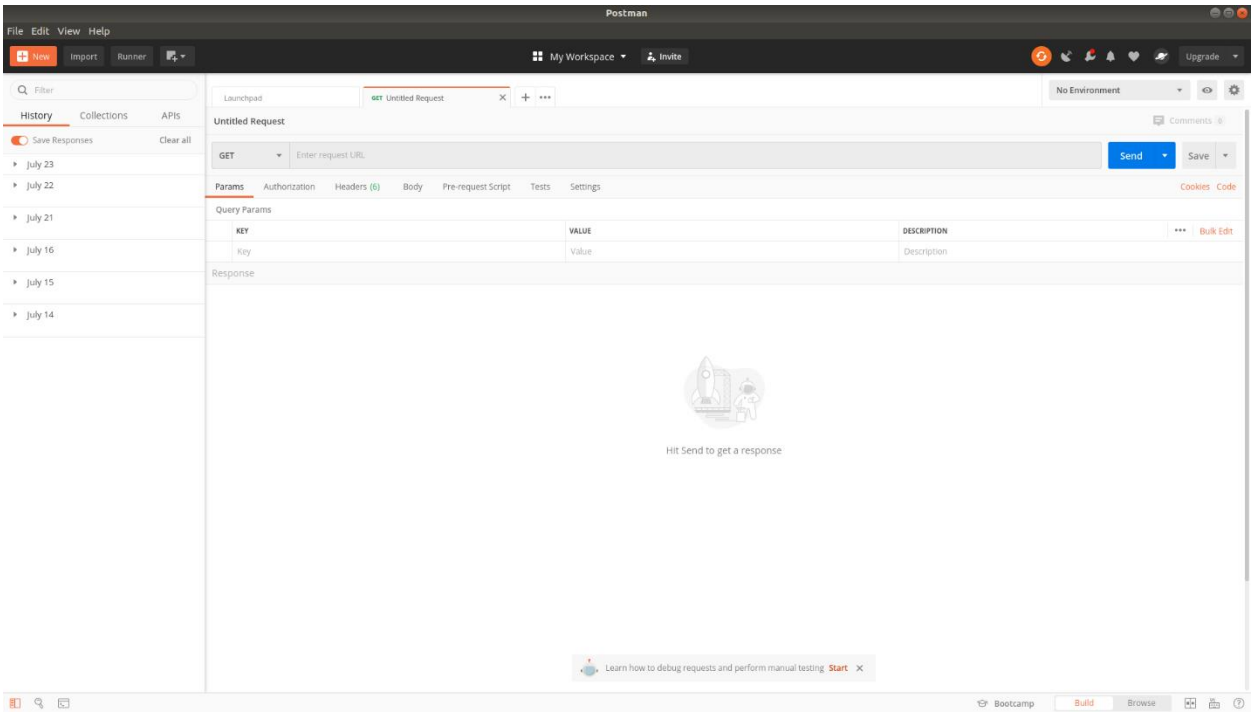
root@xmf-h:~# kl02-flasherapp firmware.bin

6.6 REMOTE FIRMWARE UPGRADE

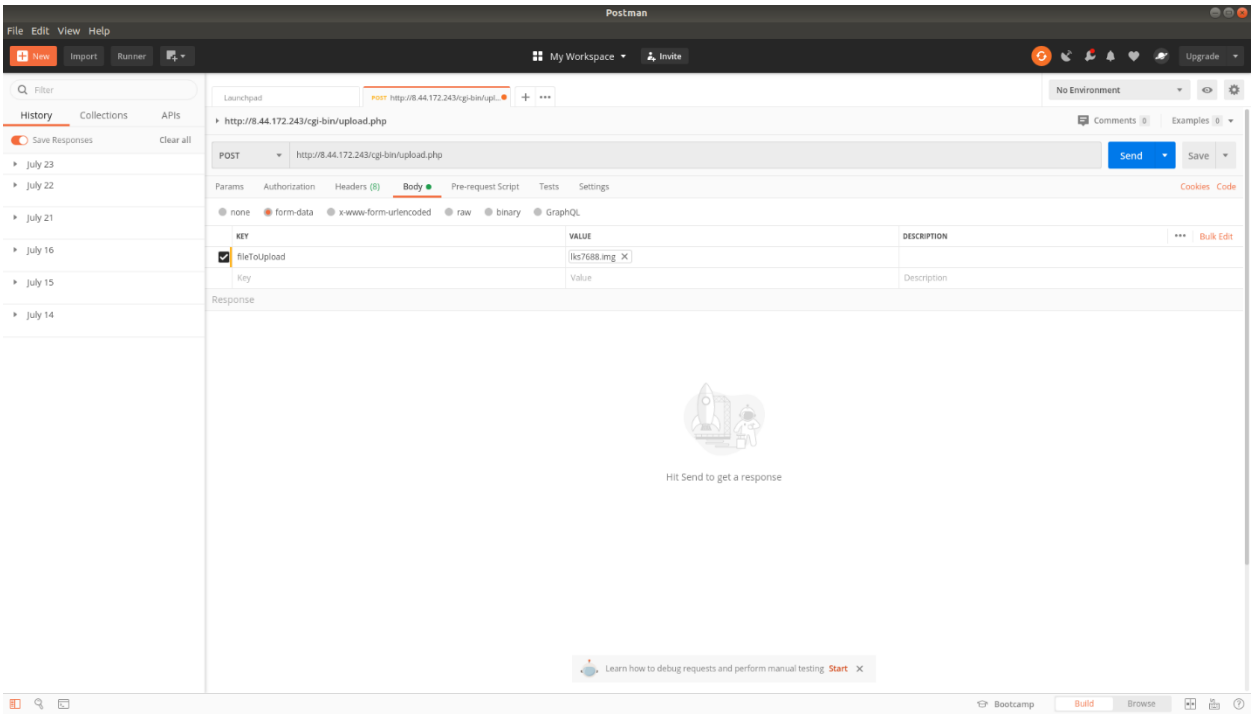
This section describes remote firmware upgrade procedure for XMF-H through Postman Chrome extension.

Procedure:

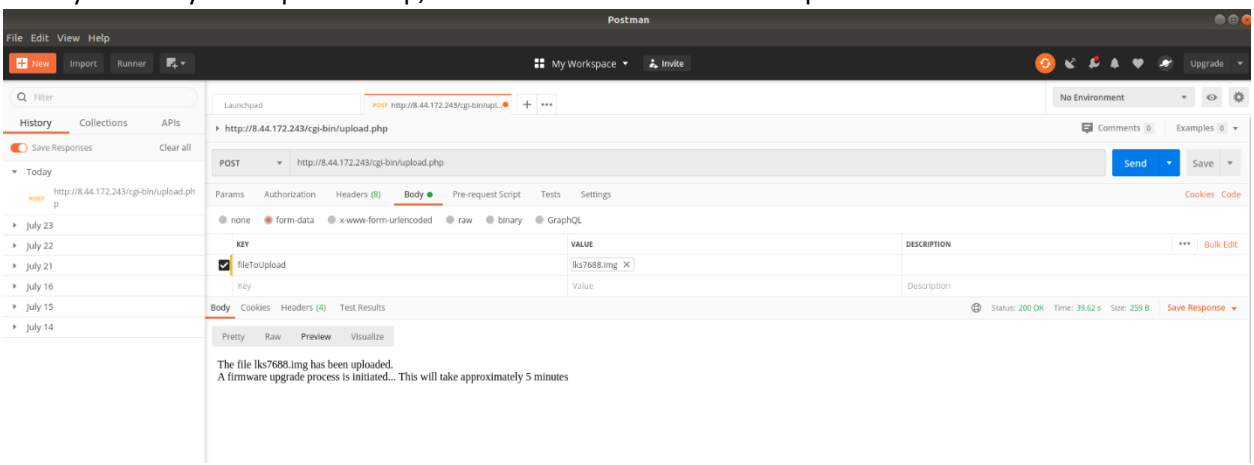
1. Launch Postman Chrome Browser extension.



2. You can create a new request from the Postman launch screen, using New > Request, or by clicking the + button to open a new tab.
3. When using the launch screen or New button, you can first give your request a name and description, and choose / create a collection to save it in. Save to create your request. It will open in a new tab.
4. Once your new tab is open, you can specify the details you need for your request.
5. Use the API http://IP-ADDRESS/cgi-bin/upload.php in the Postman endpoint bar.
6. Set method type to POST.
7. Then select Body -> form-data -> Enter parameter name (fileToUpload) and on the right side of key column, there will be dropdown "text, file", select File. Choose your image file and post it.



8. Once you have your request set up, click Send and examine the Response.



9. A file should be uploaded to /tmp/ to the board.
10. The firmware uploads to XMF-H. The WiFi LED will start blinking fast. Do not press the PWR button or reset the board. Please make sure the board stays connected to its power source until the firmware update is complete. **After approximately 5 minutes,** the firmware upgrade will be completed. The device will automatically reboot.

II-VI Incorporated
375 Saxonburg Boulevard
Saxonburg, PA 16056

7. FCC Interference Statement

Federal Communications Commission (FCC) Interference Statement

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules.

These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generate, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications.

However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

FCC Caution: Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.

The SAR limit adopted by USA and Canada is 1.6 watts/kilogram (W/kg) averaged over one gram of tissue. The highest SAR value reported to the Federal Communications

Commission (FCC) and the Industry Canada (IC) for this device type when it is properly worn on the body is 0.09 W/kg.

RF exposure warning

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment.