**Installation of RPM**

The DVT rpm contains command line utilities, scripts and libraries needed to perform DVT of MPSi card.

To install rpm, run command as below:

rpm -ivh <rpm-name>

e.g.

[root@localhost bin]# rpm -ivh ife-lls-mps-1.2-1.x86\_64.rpm

Upon successful installation, all the utilities will be installed in /usr/bin on MPS.

Note1: All the commands mentioned in this document assume root privileges.

Note2: DVT utilities need ftdi library installed on MPS target. It can be downloaded and installed from

<http://pkgs.org/centos-7/epel-x86_64/libftdi-1.1-4.el7.x86_64.rpm.html>

Note3: PI 4.4 delivered ife rpm is dependent on several other packages which are included in the delivery. To install, untar the tgz package and run qual\_rpm\_instsall.sh.

**I2C Driver installation and loading**

Microchip driver must be installed and loaded to communicate with I2C devices on MPSi card from X86 processor. The driver binary is included in the DVT rpm.

To install the driver, execute the script ad below.

[root@localhost bin]# driver\_install.sh

To load the driver, execute the script ad below.

[root@localhost bin]# driver\_load.sh

I found the requested VID/PID: 04d8, 00dd

I2C related drivers are loaded

**DVT Utilities**

* ***FTDI Bitbang port-B:***

The FTDI port-B can be configured to drive pins as GPIO out signals. This low level utility configures FTDI port-B in Bitbang mode.

Following command will show the usage

[thales@localhost bin]$ ftdibbb

USAGE:

Bitbang FTDI Port-B

ftdibbb <dir> <val>

Example:

ftdibbb 0x10 0x00

This command will configure bit4 on port-B i.e FTDI\_CONN\_CTRL\_I2C\_SEL as output and will drive logic 0 on the line.

* ***i2c device register read-write:***

i2c devices that support 8-bit address and data can be accessed using low level i2c utility

Following command will show the usage

[root@localhost bin]# i2c

Usage:

i2c <bus> <addr> <reg> <val> --- for i2c register write

i2c <bus> <addr> <reg> --- for i2c register read

Example:

i2c 7 0x73 0x00 0x7f --- writes 0x7f to reg 0x00,chip 0x73, on I2C bus 7

i2c 7 0x4c 0x04 --- reads from reg 0x04, chip 0x4c, on I2C bus 7

Note that following steps must be performed before accessing i2c devices from X86

* I2C drivers must be loaded
* GPIO FTDI\_CONN\_CTRL\_I2C\_SEL must be selected i.e ftdibbb 0x10 0x00
* I2C switch must be enable the devices e.g. to enable all devices, i2c 7 0x73 0x00 0x7f
* ***FPGA (i2c)register read-write:***

FPGA registers can be accessed using fpga utility

Following command will show the usage

[root@localhost bin]# fpga

Usage:

fpga <reg> <val> --- for FPGA register write

fpga <reg> --- for FPGA register read

Example:

fpga 0x01 0x23 --- Writes value 0x23 to reg 0x01

fpga 0x05 --- reads reg 0x5

* ***EEPROM(i2c) read-write:***

i2c eeprom can be accessed using eeprom utility.

Following command will show the usage

[root@localhost bin]# eeprom

Usage:

eeprom <addr> <val> --- for EEPROM register write

eeprom <addr> --- for EEPROM register read

Example:

eeprom 0x01 0x23 --- Writes value 0x23 to addr 0x01

eeprom 0x05 --- reads addr 0x5

* ***IO Expander (i2c) read-write:***

io expander is connected on i2c bus and used to control devices using GPIO. ioexp utility can be used to drive these GPIO signals.

Following command will show the usage and reads io exapander chip registers current values.

[root@localhost bin]# ioexp

USAGE:

ioexp <bit-num> <Mode> <val>

bit-num: Must be between 0-7

Mode:

0=>Output

1=>Input

val: Must be 0 or 1

Example: ioexp 2 0 0 -- configures bit 2 as output and drives value 0 on it

Example: ioexp 3 1 -- configures bit 3 as input

Example: ioexp 0 0 0 -- to disconnect SSD drives (power off) – It may take ~ 15 sec

Example: ioexp 0 0 1 -- to reconnect SSD drives (power on)

Example: Reset Video module – It may take ~ 15 sec

ioexp 1 0 0

ioexp 1 0 1

* ***Switch EEPROM (ftdi-spi) read-write:***

Switch eeprom is connected on ftdi-spi interface and can be accessed using eeprom\_brsw utility

Following command will show the usage.

[root@localhost bin]# eeprom\_brsw

USAGE:

./eeprom\_brsw <adr> <data> -- for eeprom write word

./eeprom\_brsw <adr> -- for eeprom read word

adr : 10 bits address (0x000 - 0x3ff)

data: 16 bits data (0x0000 - 0xffff)

* ***FPGA EEPROM (ftdi-spi) read-write:***

Fpga eeprom is connect on ftdi-spi interface and can be accessed using eeprom\_fpga utility

Following command will show the usage.

[root@localhost bin]# eeprom\_fpga

USAGE:

./eeprom\_fpga <adr> <data> -- for eeprom write byte

./eeprom\_fpga <adr> -- for eeprom read byte

adr : 9 bits address (0x000 - 0x1ff)

data: 8 bits data (0x00 - 0xff)

* ***Temperature sensors (i2c):***

Temperature sensors are connected on i2c bus and can be accessed from X86 processor using tempsensor utility.

Following command will show the usage and read temperature from all the sensors.

[root@localhost bin]# tempsensor

USAGE:

tempsensor [sensor id]

sensor id:

1 --> Internal Temp chip 0x4c

2 --> TR1 Temp Temp chip 0x4c

3 --> TR2 Temp Temp chip 0x4c

4 --> Internal Temp chip 0x4d

5 --> Internal Temp chip 0x4e

6 --> Internal Temp chip 0x4f

* ***Voltage sensors (i2c):***

Voltage sensors are connected on i2c bus and can be accessed from X86 processor using voltsensor utility.

Following command will show the usage and read voltage from all the sensors.

[root@localhost bin]# voltsensor

USAGE:

voltsensor [sensor id]

sensor id:

1 --> VCC[3P3VDC] chip 0x4d

2 --> VCC[3P3VDC] chip 0x4e

3 --> V1[5P0VDC]] chip 0x4e

4 --> V2[5P0VDC\_MPS] chip 0x4e

5 --> V3[1P8VDC]MPS] chip 0x4e

6 --> V4[1P2VDC]MPS] chip 0x4e

7 --> VCC[3P3VDC]PS] chip 0x4f

8 --> V1[3P3VDC]]PS] chip 0x4f

9 --> V2[3P3VDC\_SSD1] chip 0x4f

10 --> V3[3P3VDC\_SSD2] chip 0x4f

11 --> V4[3P3VDC\_SSD3] chip 0x4f

* ***Program K60:***

flashk60 utility can be used to program boot/app on to K60 processor.

Following command will show usage.

[root@localhost bin]# flashk60 h

flashk60 HELP:

flashk60 [ h|H|? ] flashk60 help

flashk60 [ e|E ] erase K60 flash

flashk60 [ p|P ] [ b|B ] bootfile program/verify K60 Boot flash

flashk60 [ p|P ] [ a|A ] appfile program/verify K60 App flash

/dev/ttyUSB2 (video module console) will be lost after running this utility and reboot is required to recover the /dev/ttyUSB2 (video module console).

* ***Program FPGA:***

TBD

* ***VA Area setup:***

This script programs ip address, mac address and port in to fpga registers for specified VA areas

Following command will show usage.

[root@localhost bin]# vaSetup.sh

Usage : vaSetup.sh '<Areas>'

where

Area is : '1 2 3 4 5 6 7 8'

example : vaSetup.sh '1 3 5'

* ***Video Encoder control:***

This script controls video encoder state.

Following command will show usage.

[root@localhost bin]# videoEncoder.sh

Usage:

/usr/bin/videoEncoder.sh [start|stop]

The information below is extracted from usage/help from binaries developed by Pessac team. Current information may be incorrect/incomplete and shall be further reviewed and furnished by Pessac team.

**Note: Commands mentioned below need sidekick (K60) programmed. The K60 image is part of rpm and is located at /Sidekick.afx.S19. To program sidekick, following command can be used:**

***flashk60 /Sidekick.afx.S19***

**Once, sidekick is programmed, assign IP address to network interfaces.**

***ifconfig eno1 192.168.0.64***

***ifconfig eno1:sk 10.1.69.69***

* ***Binary IO:***

Following command will show usage.

[root@localhost bin]# demo\_binaryio

Usage: demo\_binaryio command args

available commands are:

getDiscreteInput Discrete

setDiscreteOutput Discrete 0|1

readPinsProg

readLruId

Where input Discrete is one of:

LLS\_IN\_GP\_KL\_01

LLS\_IN\_GP\_KL\_02

LLS\_IN\_GP\_KL\_03

LLS\_IN\_GP\_KL\_04

Where output Discrete is one of:

LLS\_OUT\_GP\_KL\_01

LLS\_OUT\_GP\_KL\_02

LLS\_OUT\_GP\_KL\_03

* ***BITE monitoring:***

Following command will show usage.

[root@localhost bin]# demo\_bite

Usage: demo\_bite command args

available commands are:

getMpsiTemperatureSensorInfo [sensor id]

getMpsiTemperature [sensor id]

getMpsiVoltageSensorInfo [sensor id]

getMpsiVoltage [sensor id]

getMpsiAllSensorData

Where Mpsi temperature sensor id is one of:

LLS\_TEMPERATURE\_SENSOR\_ID\_U15\_TINT

LLS\_TEMPERATURE\_SENSOR\_ID\_U15\_TR1

LLS\_TEMPERATURE\_SENSOR\_ID\_U15\_TR2

LLS\_TEMPERATURE\_SENSOR\_ID\_U130\_3V3

LLS\_TEMPERATURE\_SENSOR\_ID\_U14\_3V3

LLS\_TEMPERATURE\_SENSOR\_ID\_U123\_3V3

Where Mpsi voltage sensor id is one of:

LLS\_VOLTAGE\_SENSOR\_ID\_U130\_3V3

LLS\_VOLTAGE\_SENSOR\_ID\_U14\_3V3

LLS\_VOLTAGE\_SENSOR\_ID\_U14\_5V

LLS\_VOLTAGE\_SENSOR\_ID\_U14\_5VMPS

LLS\_VOLTAGE\_SENSOR\_ID\_U14\_1V8

LLS\_VOLTAGE\_SENSOR\_ID\_U14\_1V2

LLS\_VOLTAGE\_SENSOR\_ID\_U123\_3V3

LLS\_VOLTAGE\_SENSOR\_ID\_U123\_EXT\_3V3

LLS\_VOLTAGE\_SENSOR\_ID\_U123\_3V3\_SSD1

LLS\_VOLTAGE\_SENSOR\_ID\_U123\_3V3\_SSD2

LLS\_VOLTAGE\_SENSOR\_ID\_U123\_3V3\_SSD3

* ***LRU Configuration:***

Following command will show the usage

[root@localhost bin]# demo\_lruconfig

Usage: demo\_lruconfig command args

available commands are:

readLruConfiguration

* ***RS-485:***

Following command will show usage.

[root@localhost bin]# demo\_serial485

Usage: ./demo\_serial485 command args

available commands are:

initParametersAr485

openAndConfigureAr485 portId baudrate databits stopbits parity

ar485Write portId <HEX bytes to send>

ar485Read portId

ar485LoopbackTest baudrate databits stopbits parity

ar485LoopbackTest2 baudrate databits stopbits parity cycles

Where baudrate is one of : 2400 9600 115200

Where data bits is one of: 7 8

Where stop bits is one of: 1 2

Where parity is one of : N O E

Where portId shall be between 0 and 5 Where string contains up to 100 characters

NOTE:

The option “ar485LoopbackTest2” is for loopback test which allow for master port (0) to communicate to each slave port (1-5) successively and provide result stats. External loopback cable to connect master port and slave ports are required to perform this test.

An example of test which runs for 2 cycles (2 rounds of master sending message to each slave) and output:

[root@localhost bin]# demo\_serial485 ar485LoopbackTest2 9600 8 1 N 2

Req Resp Result

Master-Slave1 2 2 PASS

Master-Slave2 2 2 PASS

Master-Slave3 2 2 PASS

Master-Slave4 2 2 PASS

Master-Slave5 2 0 FAIL

* ***Ethernet Switch:***

Following command will show usage.

[root@localhost bin]# demo\_switch

Usage: demo\_switch command args

available commands are:

receiveMfp

* ***GPIO Test:***

This script alternately asserts and de-asserts all GPIO outputs at around 0.8 Hz and checks all GPIO inputs. If all GPIO inputs reflect the state of connected GPIO outputs then it will print “.” and if the state of any GPIO input does not match with connected GPIO output then it will print “f”

The script can be run as below without any argument.

[root@localhost bin]# ./gpioTest.sh

**Note: The GPIO loopback connector should be plugged-in to P3 connector on x86 board**.



If loopback connector is plugged in and GPIO inputs reflects the values of GPIO outputs then the script will keep printing output as below

“…………………………………..”

If loopback connector is not plugged in and/or GPIO inputs does not reflect the values of GPIO outputs then the script will keep printing output as below

“ffffffffffffffffffffffffffff”