The PMT-3000 programming kit includes:

1) PMT-3000 programming cable (40 pin header to 8 pin Hirose).
2) Micro SD card. (Raspbian 10 "buster" 32-bit)
3) Drawings of the connector pinouts.

The micro SD card contains the OS and software needed to program the PMT-3000.

To get started you will also need:

a) Raspberry pi. The software has been tested on both pi-3 model B and pi-4 model B 4GB.
b) Power supply and power cable for the pi.
c) USB Keyboard, USB Mouse and HDMI Monitor.
d) Ethernet connection (Optional but recommended).
d) USB mini b cable to power and connect the PMT-3000 to the raspberry pi.

You should place the SD card in the raspberry pi's micro SD card slot then connect the keyboard, mouse and monitor.

Next connect the 40 pin connector to the GPIO header of the raspberry pi. The included drawings indicate position of pin one. There is a yellow dot on pin one of the 40 pin connector. Now connect the other end to the 8 pin Hirose connector on the PMT-3000. If you are building a programming cable cable you can refer to the include drawings and or Appendix 4 below.

Next connect the USB mini b cable from the raspberry pi to the PMT-3000. This will power the MCA as well as give USB access to the PMT-3000 from the raspberry pi.

You can now supply power to boot the raspberry pi.

Once system is booted you can login as

username: pi
password: bpiadminpass .

The username pi is in the superuser group so you can precede commands with sudo if you want them executed as root. If you need it, the root password it is also bpiadminpass. We normally just run as root.

We are using the open source project OpenOCD's JTAG transport to access the FPGA on the PMT-3000. The micro SD card we sent has version 0.10 built and installed in

/opt/bpi/programmer/openocd .

In addition the micro SD card contains BPI's wxMCA software ported to raspberry pi. This is installed in

/opt/bpi/programmer/wxMCA .

We are using the four GPIO header pins as a bitbanged JTAG interface adapter. This allows us to attach to the FPGA and program it using OpenOCD in Serial Vector Format (SVF) mode.

All,the OpenOCD configuration files, programming scripts and firmware are located in directories rooted off of

/opt/bpi/programmer .

A) Starting BPI wxMCA Software:

Running the BPI wxMCA sofware on the raspberry pi is not technically necessary to program the MCA, however it is convenient for checking ARM and FPGA Settings/Status and determining build versioning information. The software is already installed on the microSD card BPI provides, however if you are setting up a new SD card you should review Appendix 5. for help installing the required modules.

We first need to insure we can talk to the MCA from the raspberry pi over USB. To do so will start and run both the mds (mca_server.py) and the GUI interface. Both will be run in the foreground from separate terminals so they can be stopped and started easily.

Open two terminals. In the first one we run the mds server. You must run the mca_server.py as root or sudo.

```
$ cd /opt/bpi/programmer/wxMCA
$ sudo ./run_mds.sh
```

In the second terminal we will start the GUI interface.

```
$ cd /opt/bpi/programmer/wxMCA
$ ./run_wxMCA.sh
```

You can use ctrl-c to kill the mds server. And you can close the GUI's terminal window to kill it. The BPI software does not interfere with the FPGA programming so you can leave them running.

B) Testing the JTAG Connection:

Before trying to program the FPGA it is a good idea to run a scan_chain test first to insure you have reliable JTAG connection. To do so you need to open a terminal window. In the terminal, change to the programmer directory and enter the openocd command

```
$ cd /opt/bpi/programmer/bin
$ sudo ./run_check_fpga_lx25.sh
```

This script loads the OpenOCD configuration file xc6sl25.cfg which sets up the JTAG connection and runs a scan_chain command. You should now see something like this in the terminal

```
pi@raspberrypi:/opt/bpi/programmer/bin $ sudo ./run_check_fpga_lx25.sh
Open On-Chip Debugger 0.10.0+dev-01379-g6ec2ec4d3-dirty (2021-06-02-12:07)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
   TapName            Enabled  IdCode     Expected   IrLen IrCap IrMask
-- ------------------- -------- ---------- ---------- ----- ----- ------
 0 xc6slx25.tap          Y     0x00000000 0x24004093     8 0x01  0x03

Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : SysfsGPIO JTAG/SWD bitbang driver
Info : This adapter doesn't support configurable speed
Info : JTAG tap: xc6slx25.tap tap/device found: 0x24004093 (mfg: 0x049 (Xilinx), part: 0x4004, ver: 0x2)
Warn : gdb services need one or more targets defined
```

You need to insure there are no Error : labels in the first column. You can stop the process using ctrl-c.

C) Programming FPGA:

Once you are confident you have reliable JTAG connection you can move on to programming the FPGA. Open a terminal change to the programmer's bin directory and issue the cat command

```
pi@raspberrypi:/opt/bpi/programmer/bin $ cat run_svfplayer_lx25.sh
#!/bin/sh
CONFIGS_DIR="/opt/bpi/programmer/bpi_openocd_configs"
FIRM_DIR="/opt/bpi/programmer/firmware"
FIRM_NAME="<file name>.svf"

export CONFIGS_DIR
export FIRM_DIR
export FIRM_NAME
```

```
cd /opt/bpi/programmer
openocd/bin/openocd -f $CONFIGS_DIR/xc6slx25_sysgpio_script.cfg
exit
```

You will need to edit the run_svfplayer_lx25.sh script to load the correct .svf file. The SVF formatted firmware will be made available from BPI. The edit location is highlighted in yellow above.
If instructed, the firmware SVF file can be retrieved and unzipped from BPI servers using wget.

```
# cd /opt/bpi/programmer/firmware/fpga
# wget http://www.bridgeportinstruments.com/sw/<firmware-file>.zip
# unzip <firmware-file>.zip
```

Once you have corrected the firmware filename in run_svfplaver_lx25.sh, you can execute the run_svfplayer_lx25.sh bash script. Upon execution you should notice the terminal window scrolling very fast as it echos' the SVF commands.

```
$cd /opt/bpi/programmer/bin
$ sudo ./run_svfplayer_lx25.sh
Open On-Chip Debugger 0.10.0+dev-01379-g6ec2ec4d3-dirty (2021-06-02-12:07)
Licensed under GNU GPL v2
:
0%  SD1R 16 TDI (66aa) ;
0%  SDR 16 TDI (850c) ;
:
:
75%  00000000000000000000000000000000000000000000090004004106a956a95) ;
75%  RUNTEST 2224 TCK;
75%  RUNTEST 6000 TCK;
75%  SIR 6 TDI (02) ;
:
 ***********Pauses here for upto ~4.5 minutes***************
:
 95%  TDR 0 ;
(Above Padding command skipped, as per -tap argument)
 95%  HDR 0 ;
(Above Padding command skipped, as per -tap argument)
 95%  SIR 6 TDI (3f) ;
 95%  SDR 1 TDI (00) SMASK (01) ;
shutdown command invoked
$
```

It can take between 3 to 7 minutes to complete the programming. The programming time is FPGA code dependent, so you will need to be patient. During programming, the process seems to pause or hang for about 5 minutes at the 75% mark. This is normal, it will eventually continue.

After the device is programmed you can use the BPI wxMCA software mentioned above to explore the new settings.

D) Miscellaneous Linux Commands:

Allowing remote access over Ethernet using secure shell daemon sshd.

```
$ /etc/init.d/ssh restart .
```

Listing devices connected by USB. BPI devices will have usb id 1f4a:*xxxx*

```
$ lsusb .
```

Appendix 1. Setting up Raspbian as OpenOCD programmer.

We run OpenOCD on the raspbery pi and use its gpio header as a bitbanged jtag. If you are using the micro SD card BPI provides the OS should already be set up you can logon as

user: pi
password: bpiadminpass

If you are setting up a new install of raspbian you will need to download the programming software along with firmware from BPI. The programming scripts are hard coded with the expected location of the files. This insures you are using the BPI compiled version of OpenOCD. We have made no changes to the source code, but it is a newer version and is installed in a non-standard location.

```
# cd /opt
# wget http://www.bridgeportinstruments.com/sw/bpi_programmer-XX.tar.gz
```

The bpi_programmer-XX.tar.gz tarball contains OpenOCD along with BPI programming scripts and configuration files. Again, the script paths are hard coded so you should decompress the bpi_programmer-XX.tar.gz archive in the /opt directory.

The bash script run_svfplayer.sh can be used to program the PMT-3000.
You will need to change the name of the SVF file in the xc6slx16_sysgpio_script.cfg. This is outlined in Section C).

```
# cp bpi_programmer-XX.tar.gz /opt
# cd /opt
# tar -vxf bpi_programmer-XX.tar.gz
```

To complete the setup of OpenOCD, you will need to add the following libraries to Raspbian

```
# apt-get update
# apt-get install libftdi1 libgpiod2 libhidapi-hidraw0 telnet
```

Once you have completed you should be able to jump to Section B) and start exploring the JTAG connection.

Appendix 2. Important Files and Scripts in the tarball (deprecated).

/opt/bpi/programmer

armMorpho_std_xxxM_b12.svf	This is the fpga firmware file translated to SVF format.
run_svfplayer.sh	Bash script to call OpenOCD and load SVF file into Spartan6 lx16 fpga
xc6slx16.cfg	Openocd config file allowing simple JTAG connection to the Spartan6 lx16 FPGA.
xc6slx16_sysgpio_script.cfg	Openocd config file and programming script called by run_svfplayer.sh
openocd/	root directory of the openocd software.
openocd-code.tar.gz	openocd source code.

Appendix 3. OpenOCD Links.

http://openocd.org/doc-release/html/index.html

Appendix 4. Raspberry Pi sysgpio adapter JTAG cable pin out.

There are two drawings attached which illustrate the table pin out below.
eMORPHO Program cable.pdf and Raspberry_pi.pdf describe the cable available from BPI.
You should be able to make your own with the following table.

Signal	Color	Rasp pi GPIO Header	8 pin Circular Hirose Connector
TDI	Purple	#19	3
TDO	Brown	#21	2
TCK	Orange	#23	4
TMS	White	#22	1
GND	Green	#14	7
GND	Green	#20	8
Vref	RED	N/C	5
N/C	N/C	N/C	6

Appendix 5. Installing pyzmq and wxPython widgets for wxMCA software.

The latest version of Raspbian (buster) comes with Python 3.7 , but you will need to install pyzmq for the mds server to run. We can use pip for this.

```
$ sudo python3.7 -m pip install pyzmq
```

This should complete the installation for mds server.
To run the GUI interface you will need wxPython. The wxPython modules in the Raspbian repositories are out of date. We have built the necessary wx module wheel and it is include in the bpi_programmer-0.10.tar.gz file. To install it you will need to install matplotlib and upgrade numpy.

```
$ cd /opt/bpi/rpi_wls
$ sudo python3.7 -m pip install wxPython-4.1.1-cp37-cp37m-linux_armv7l.whl
```

We need to install matplotlib

```
$ sudo python3.7 -m pip install matplotlib .
```

Now will need to upgrade numpy

```
$ sudo python3.7 -m pip install -U numpy .
```

And last we need to add the libatlas library

```
$ sudo apt-get install libatlas-base-dev .
```

Once completed you should be able to jump the Section A) and run the BPI software.