## Metis - How it works

## Revisions

| Rev | Date | Changes | By |
|---|---|---|---|
| 1.0 | 23 Aug 2010 | First draft | VK6APH |
| 1.1 | 13 Sep 2010 | Added Boot loader text | VK6APH |
| 1.2 | 3 Nov 2010 | Changed to current discovery data format | VK6APH |
| 1.3 | 14 Nov 2010 | Added data Start/Stop commands. Replaced 'Ozy' with 'Metis'. | VK6APH |
| 1.4 | 16 Nov 2010 | Added new Discovery data format | VK6APH |
| 1.5 | 18 Nov 2010 | Added DHCP retries | VK6APH |
| 1.6 | 19 Nov 2010 | Added Assign IP data format | VK6APH |
| 1.7 | 22 Nov 2010 | Moved Assign IP to Bootloader. Added read MAC and IP to Bootloader. Updated Bootloader menu. Added static IP indication to LED[6]. | VK6APH |
| 1.8 | 30 Nov 2010 | Added LED[x] to PCB silkscreen layout | VK6APH |
| 1.9 | 6 Dec 2010 | Added Destination Unreachable function. Documented DHCP IP lease renewal process. Added fast/slow flash of LED[0] if not full duplex. | VK6APH |
| 1.10 | 9 Dec 2010 | Added description of inline FPGA code update | VK6APH |
| 1.11 | 12 Dec 2010 | Updated description of LED[6] | VK6APH |
| 1.12 | 13 Dec 2010 | Updated description of inline FPGA code update | VK6APH |
| 1.13 | 14 Dec 2010 | More updated to inline FPGA code update | VK6APH |
| 1.14 | 18 Dec 2010 | Added draft description of JTAG loader | VK6APH |
| 1.15 | 2 Jan 2011 | Added description of JTAG load of HPSDR boards | VK6APH |
| 1.16 | 12 Jan 2011 | Replaced *.pof with *.rbf when programming Metis flash or flash on Penelope, PennyLane or Mercury boards. Updated JTAG file format to *.rbf and operations on file needed before file is sent. Changed protocol so that Metis responds to 'from port' of originating PC. | VK6APH |
| 1.17 | 16 Jan 2011 | Metis now always listens on port 1024. Updated protocol. | VK6APH |
| 1.18 | 16 Jan 2011 | Added LED[11] -indicates EPCS16 being erased or programmed. Added warning not to have USB Blaster connected whilst programming. | VK6APH |
| 1.19 | 24 Jan 2011 | Added Bootloader and Metis file merge instructions. | VK6APH |
| 1.20 | 25 Jan 2011 | Corrections to JTAG protocol | VK6APH |
| 1.21 | 1 Feb 2011 | Corrected SVN URL | VK6APH |
| 1.22 | 10 Mar 2011 | Replaced programmer.exe reference with HPSDRProgrammer | VK6APH |
| 1.23 | 26 Mar 2001 | Corrected data format for Bootloader. Clarified that LEDs refer to Mercury board in Bootloader programming example and corrected LED 4 not 7 flashes. | VK6APH |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

*LEDS*

```
LED[0] is located at the top of the board and LED[12] at the bottom

D14 = LED[0]    - Displays state of PHY negotiations - fast flash if no Ethernet
                  connection, slow flash if 100T, on if 1000T and swap between fast and
                  slow flash if not full duplex
D15 = LED[1]    - Lights when the PHY receives Ethernet traffic
D16 = LED[2]    - Lights when the PHY transmits Ethernet traffic
D17 = LED[3]    - Lights when an Ethernet broadcast is detected
D22 = LED[4]    - Lights when traffic to the boards MAC address is detected
D12 = LED[5]    - Lights when an ARP request is received
LED[6]          - Displays state of DHCP negotiations or static IP - on if DHCP ACK, slow
                  flash if DHCP NAK, fast flash if DHCP time out and long then short
                  flash if static IP. If DHCP NAK, or time, out an IPIPA IP address will
                   be used.
D13 = LED[7]    - Lights when a ping request is received
D21 = LED[8]    - Lights when an Metis discovery packet is received
D20 = LED[9]    - Lights when an Metis discovery packet reply is sent
D19 = LED[10]   - Lights when 0x7F7F7F received from PC
D18 = LED[11]   - Lights when a receive sequence error is detected or the  EPCS16 flash
                  memory is being erased or programmed.
D23 = LED[12]   - HEART_BEAT - Flashes once per second
```

*FPGA loaded successfully indicator*

A Heartbeat counter, running off the 125MHz clock from the PHY, continually flashes the HEART_BEAT LED once per second. This indicates that the FPGA has code loaded and that the PHY is providing the necessary master clock.

*Master Reset*

At power on, Tx reset signal (Tx_reset) is set high. This prevents all other code from running other than the Reset and Initialization code. The MAC address of the board is then read from its EEPROM.

The EEPROM is then read to see if a valid IP address is available. If the IP address is 0.0.0.0 or 255.255.255.255 then the Valid_IP flag is not set.

The reset code then reads from PHY Register 31, via the MDIO module, and looks for a 100/1000T, full duplex, connection to a PC or Ethernet Switch. Once found, LED[0] indicates the connect speed as follows:

| | |
|---|---|
| 1000T | Steady |
| 100T | Slow flash |
| Error, half duplex or no connection | Fast flash |

Once a valid network connection has been found a delay of one second is applied to allow the network to stabilize.

*Main Code*

If the Valid_IP flag is set then the board will use this IP address and is ready for further commands. LED[6] will give a long flash followed by a short flash to indicate a static IP address is in use.

If the Valid_IP flag is not set the code then attempts to obtain an IP address via DHCP.

Once an IP address is obtained LED[6] is steady on. If an NAK occurs whilst trying to obtain an IP address then LED[6] flashes slowly.

If a DHCP request is not responded to within 2 second then LED[6] flashes rapidly.

Metis will attempt to obtain an IP address via DHCP for four more times with a two second delay between retries.

If a DHCP provided IP address is successfully received then Metis will use this.

Metis will attempt to renew its DHCP assigned IP address at time equal to the DHCP provided lease time/2. Metis will request a lease renewal directly to the IP address of the DHCP server that issued the original IP address.

If a lease renewal is not received then Metis will try again at lease time * 0.75. The request is sent to the IP address of the original DHCP server. If unsuccessful Metis will broadcast a DHCP Discovery request in an attempt to locate an alternative DHCP server.

At power on, if an IP address is not obtained using DHCP then an APIPA address in the range 169.254.x.x will be assigned. At present the last two bytes are obtained from the last two bytes of the boards MAC address.

ToDo: Change this to a random address in the future and check it's unique via ARP.

Note: the APIPA IP address is not stored in Flash Memory.

Once an IP address has been obtained, either static, via DHCP or an APIPA address assigned, then the board will respond to Discovery, ARP and ping packets etc.

*Metis Discover Packet format*

A Metis Discovery packet is sent from a PC in order to determine if Metis boards are present on the network, and if so, how many.

The Discovery packet is a UDP/IP frame sent to Ethernet address 255.255.255.255 (i.e. a Broadcast) and port 1024 with the following payload:

<0xEFFE><0x02><60 bytes of 0x00>

*Metis Discovery reply format*

Upon receipt of this broadcast a Metis board will reply with a UDP/IP frame sent to the 'from port' on the IP Address and MAC Address of the PC originating the Discovery broadcast.

The payload of the UDP/IP reply frame is as follows:

<0xEFFE><0x02>< Metis MAC Address><51 bytes of 0x00>

where

Metis MAC Address = 6 bytes (MAC Address of the Metis responding to the Discovery Broadcast)

Note1: The IP address of the Metis board(s) responding can be obtained from the IP header.

Note2: A Metis board will NOT respond to UDP/IP data from a PC SDR program until a valid response to a Discovery Broadcast has been sent to the PC by the Metis board AND a Start command has been received by Metis.

Note3: Metis will respond to the 'from port' of the PC that does the initial Discovery request. This port will be used for all other UDP/IP communications. If the 'from port' is required to be changed then a new Discovery request is required.

Metis will use port 1024 as its 'from port'.

Once a Discovery exchange has taken place the PC may communicate with the Metis board(s) using Start and Stop commands as follows.

*Start Command*

Metis will start sending data to the PC once a Start Command has been received. The Start Command is a UDP/IP frame sent to the Ethernet address assigned to the Metis card and port 1024. The command has the following format:

<0xEFFE><0x04><0x01>< 60 bytes of 0x00>

*Stop Command*

Metis will continue sending data following a Start Command until a Stop Command is received. The Stop Command is a UDP/IP frame sent to the Ethernet address assigned to the Metis card and port 1024. The command has the following format:

<0xEFFE><0x04><0x00>< 60 bytes of 0x00>

Note: Metis will stop sending if it receives a 'Destination Unreachable' packet.

*PC <> Metis UDP/IP data format*

Data from the PC to a Metis board is via UDP/IP, sent to the IP address of the Metis board and port 1024, with the following payload:

<0xEFFE><0x01><End Point><Sequence Number>< 2 x HPSDR frames>

where

| | | |
|---|---|---|
| End point | = 1 byte | (0x02 – representing USB EP2) |
| Sequence Number | = 4 bytes | (32 bit unsigned integer starting at zero and incremented each frame – bytes are in network order [big-endian]) |
| HPSDR data | = 1024 bytes (2 x 512 byte USB format frames) | |

Data from the Metis to a PC is via UDP/IP, sent to the IP address of the PC and 'from port' of the originating PC (and from port 1024), with the following payload

<0xEFFE><0x01><End Point><Sequence Number><2 x HPSDR frames>

where

| | | |
|---|---|---|
| End point | = 1 byte | (0x04 or 0x06 – representing USB EP4 or EP6) |
| Sequence Number | = 4 bytes | (32 bit unsigned integer starting at zero and incremented each frame and independently for each end point) |
| HPSDR data | = 1024 bytes (2 x 512 byte USB format frames) | |

Note: The Sequence Number sent by Metis will be reset to zero each time a Start Command is received. Similarly, the Sequence Number sent by the PC should be reset to zero each time a Start Command is sent.

*HPSDR Frame Protocol*

The latest version of this protocol (USB protocol proposal Vx.xx.doc) can be found here:

svn://svn.openhpsdr.org/svn/repos_sdr_hpsdr/trunk/Documentation

*Flash Memory Loader*

IMPORTANT: Do not connect a USB Blaster to the flash programmer socket on Metis (J6) whilst erasing or programming the flash memory. The additional capacitance prevents the erase operation completing correctly.

The EPCS16 flash memory on the Metis board contains two FPGA images. The image starting from address 0x00 is a Boot loader whilst that starting from 0x01000000 (i.e. 1MB) is the normal Metis image.

The Altera ALTREMOTE_UPDATE Megafunction is used in order to select which of these two images the FPGA should load following a Power Reset. The desired image to load is determined by the state of the FPGA input pin MODE1. If set to 0v (by fitting a jumper on JP1) then the Boot loader image is loaded. If set to 3.3v (no jumper fitted) then the Metis image is loaded.

Note: If you wish to use the normal Altera Quartus Byte/USB Blaster to update the EPCS16 then please refer to the Appendix A that explains how to combine the Boot loader and Metis *.sof files into a single *.pof file that can be loaded into the EPCS16.

If you do not wish to use the Boot loader facility then just program the EPCS16 as per normal.

There are two ways to load the EPCS16 flash memory on the Metis board via the Ethernet port.

1. Command Mode
2. Using the Bootloader

Command Mode

This method will normally be used to load a new Metis image into the EPCS16. It requires minimal input from the user and will automatically restart the Metis board, so that the new image is run, once successfully loaded.

A PC program is available, that runs under Windows, MacOS and Linux, that automates the updating of the Metis image. The program is called HPSDRProgrammer and can be downloaded from http://openhpsdr.org/download.php.

See the 'Metis User Manual.pdf' for instructions in using HPSDRProgrammer.

An explanation as to how HPSDRProgrammer works is as follows:

1. Do a Metis Discovery to determine the IP address of the Metis card that is to be updated;

2. Erase the current Metis image in the EPCS16;

3. Load the new image.

The new image data will automatically be loaded into the FPGA, and hence run, once the EPCS16 has been programmed.

The data format is UDP/IP. Data is sent to the IP address of the Metis board and port 1024 as follows:

```
        Program     0xEFFE, 0x03, 0x01, <# of blocks><256 bytes ofdata>

        Erase       0xEFFE, 0x03, 0x02, <60 bytes of 0x00>

where '# of blocks'  means the number of 256 byte blocks that will be sent, 4 bytes,
                     bytes are in network order [big-endian]
```

Metis will respond to the IP address of the PC and 'from port' of the originating PC with a UDP/IP packet as follows:

```
       <0xEFFE><reply>< Metis MAC Address><51 bytes of 0x00>


 Where Metis MAC Address = 6 bytes (MAC Address of the Metis responding to the command)

and reply means:

      0x03 = erase command completed successfully
      0x04 = ready for next 256 bytes of data
```

The original data to send to the Metis board is a compressed *.rbf file, (i.e. use the compress function in Quartus when compiling).

Add bytes of 0xFF to the end of the file such that the length of the file is % 256.

Before the new Metis data can be loaded the upper 1MB of the EPCS16 must be erased. This is done using the command above. Note that this command may take some time ( > 14 seconds) to indicate that it has completed.

Once an erase acknowledgement is received (see above) the converted data can be sent 256 bytes at a time as per the above command. The Metis board will request the next block of 256 bytes when ready.

Once the last block of data has been sent the new code will automatically be loaded into the FGPA and run.

Bootloader

The Bootloader enables new code to be loaded into the FPGA EPCS16 Flash memory as well as reading the MAC address and configuring the configuration EEPROM.

This method of loading code into the FPGA Flash memory is intended to be used if for some reason the usual method can't be used (i.e. a Disaster Recovery Mode). It's deliberately designed to be a simple as possible to maximise the chances of it working successfully.

The Bootloader also enables the flash memory on other HPSDR boards e.g. Mercury, located on the Atlas bus, to be reprogrammed.

In order to use the Bootloader mode a jumper has to be placed on JP1 (MODE1).

Once the jumper is in place the power to Metis should by cycled which will cause the Bootloader code to run. The board will connect to the PC/Switch at 100T, full Duplex.

A PC program is available, that runs under Windows, MacOS and Linux, that automates the use of the Bootloader functions. The program is called HPSDRProgrammer and can be downloaded from http://openhpsdr.org/download.php.

See the 'Metis User Manual.pdf' for instructions in using HPSDRProgrammer.

HPSDRProgrammer performs the following steps when updating the EPC16 flash memory on Metis.

Selecting the Program option enables the EPCS16 on the Metis board to be erased and reprogrammed starting at memory address 0x0100000 (i.e. 1MB).

NOTE: This starting address may change in the future if a larger FPGA image is required. In which case this document will be updated.

The lower 1MB can NOT be erased or re-programmed using the PC code since that would potentially overwrite the Bootloader code. If the entire EPCS16 needs to be erased then use an Altera Byte/USB Blaster. The Bootloader code is stored in the EPCS16 starting from memory address 0x0.

The file to be loaded into the EPCS16 will have the file name Metis_x.x.rbf.

Selecting the Program option will cause the upper 1MB of the EPCS16 to be erased and then loaded with the Metis_x.x.rbf file. Note that erasing can take up to 15 seconds.

The Metis_x.x.rbf file is sent in 256 byte frames and the progress is displayed.

Selecting the Erase option will erase the upper 1MB only.

Selecting the MAC Address option will read the MAC address stored in the EEPROM.

Selecting the IP address option will read the IP address stored in the EEPROM.

Note: An IP address of 0.0.0.0 is ignored and Metis will use a DHCP or APIPA IP address. An EEPROM that has not previously been assigned an IP address will read as 255.255.255.255 which is a blank EEPROM. This will also be ignored.

Selecting the IP Address option will prompt the user to enter an IP address. This should use the format xx.xx.xx.xx . Entering 0.0.0.0 will disable the static IP address facility.

LEDS

During a Read, Write, Eraser or Program cycle the following LEDs on the Metis board are activated:

LED[0] - PHY_RX
LED[1] - MAC address match
LED[2] - Erase Command
LED[3] - Program Command
LED[4] - Connect speed - slow flash for 100T, fast for no network found.
LED[5] - PHY_TX
LED[6] - OFF
LED[7] - OFF
LED[8] - PHY Rx fifo empty
LED[9] - Send more data ACK
LED[10] - Send more data
LED[11] - busy
LED[12] - HEART_BEAT – flashes once per second

Data format.

Note that in Bootloader mode Metis uses a fixed MAC address of 11:22:33:44:55:66

Data from the PC to Metis is a ***Raw*** Ethernet frame as follows:

| | |
|---|---|
| For Program | <11,22,33,44,55,66> <PC MAC Address> 0xEFFE, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00,<256 bytes of data> |
| For Erase | <11,22,33,44,55,66> <PC MAC Address> 0xEFFE, 0x03, 0x02, 46 bytes of 0x00 |
| For Read MAC | <11,22,33,44,55,66> <PC MAC Address> 0xEFFE, 0x03, 0x03, 46 bytes of 0x00 |
| For Read IP | <11,22,33,44,55,66> <PC MAC Address> 0xEFFE, 0x03, 0x04, 46 bytes of 0x00 |
| For Write IP | <11,22,33,44,55,66> <PC MAC Address> 0xEFFE, 0x03, 0x05, < IP Address> 46 bytes of 0x00 |

Where <IP Address> is the IP address to store in the EEPROM, 4 bytes.

Metis will reply to the MAC address of the sending PC with a Raw Ethernet frame as follows:

<PC MAC Address> <11,22,33,44,55,66> 0xEFFE, 03, <reply> <data> <46 bytes of 0x00 >

where reply means:

0x01 = Erase command completed successfully, <data> not sent
0x02 = ready for next 256 bytes of data, <data> not sent
0x03 = MAC address follows, <data> = MAC address, 6 bytes
0x04 = IP address follows, <data> = IP address, 4 bytes

JTAG Programmer

Metis enables other HPSDR boards to have their flash memory updated via the Atlas JTAG bus.

A PC program is available, that runs under Windows, MacOS and Linux, that automates the use of the Bootloader functions. The program is called HPSDRProgrammer and can be downloaded from http://openhpsdr.org/download.php.

See the 'Metis User Manual.pdf' for instructions in using HPSDRProgrammer.

HPSDRProgrammer does the following:

1. Interrogate the JTAG chain to check that there is a Penelope, PennyLane or Mercury card present. The card to be re-programmed needs to be in an Atlas slot ***adjacent*** to the Metis board and the 'Last JTAG' jumper fitted.
2. The Atlas JTAG chain is used to load the necessary FPGA code, which enables the flash memory to be accessed, into the target board e.g. Mercury, Penelope or PennyLane. This exposes the necessary signals to Metis via the Atlas bus;
3. The flash memory on the target board is erased;
4. The flash memory on the target board is updated; and
5. The power is cycled to load the new code from the flash memory

Since we need to re-use the Atlas bus lines for this process it is not possible to have other boards (other than Metis and the target) on the Atlas bus whilst programming.

The JTAG programmer forms part of the Bootloader code.

Step 1. The program firstly checks the ID of the device on the JTAG chain to confirm it is a Cyclone II (Penelope or PennyLane) or Cyclone III (Mercury). Send the following Raw Ethernet frame:

<11:22:33:44:55:66 ><From MAC address> 0xEFFE, 0x03, 0x06 <46 bytes of 0x00 >

Metis will respond with:

< To MAC address ><11:22:33:44:55:66 > 0xEFFE, 0x03, 0x05 <FPGA ID><42 bytes of 0x00>

where FPGA ID is 4 bytes (MSB….LSB) as follows:

for Mercury = 0x020F30XX
for Penelope or PennyLane = 0x020B20XX

HPSDRProgrammer displays a conformation message to the user that a Mercury or Penelope board has been found.

If the JTAG chain is not complete, or the ID can't be read, then 0x00000000 will be returned and a warning message will be displayed. This asks the user to check that the target board is in an adjacent Atlas slot to Metis, Metis is farthest from the power connector, there are no other boards in the Atlas bus and that the 'Last JTAG jumper is in place'.

Step 2. Load the appropriate *.rbf file into the target FPGA. For Mercury this will be 'Mercury_JTAG.rbf' and Penelope or PennyLane 'Penelope_JTAG.rbf'.

VERY IMPORTANT: This file MUST be prepared by Quartus in 'uncompressed' mode

For MERCURY ONLY - Read the Mercury_JTAG.rbf file and add 16 bytes of 0xFF to the end; count the number of bytes it now contains and save this value. Then add sufficient bytes of 0xFF to the end of the file to make it % 256.

For PENELOPE or PENNYLANE ONLY - Read the Penelope_JTAG.rbf file, remove the first 44 bytes, count the number of bytes it now contains and save this value. Then add sufficient bytes of 0xFF to the end of the file to make it % 256.

Use the following commands to program the selected file into the target FPGA, either Mercury, Penelope or PennyLane:

<11:22:33:44:55:66 ><From MAC address> 0xEFFE, 0x03 <board> < # bytes> <256 bytes of data>

where:
<board> = byte, 0x07 for Mercury and 0x08 for Penelope or PennyLane

and # bytes = 4 bytes (MSB….LSB), number of bytes in the file (before applying % 256) that will be sent

Metis will respond with

< To MAC address ><11:22:33:44:55:66 > 0xEFFE, 0x03, 0x02 <42 bytes of 0x00>

which is a 'send more' request, repeat the above until the file has been sent.

Step 3. Erase the flash memory on the target board. Send the following Raw Ethernet Frame:

<11:22:33:44:55:66 ><From MAC address> 0xEFFE, 0x03, 0x09 <46 bytes of 0x00 >

Once erased, which can take some time, Metis will respond with:

< To MAC address ><11:22:33:44:55:66 > 0xEFFE, 0x03, 0x01 <42 bytes of 0x00>

If this is not received after a suitable period then a warning message will be displayed and the user prompted to cycle the power to the HPSDR boards and repeat the process.

Step 4. Program the new code into the flash on the target board. Open the file that is to be programmed into the flash, it will have the extension *.rbf e.g. Mercury_V1.8.rbf or Penelope_V1.3.rbf. This file should preferably be generated by Quartus in *compressed* mode in order to reduce the size of the file.

Read the file and add 0xFF bytes to the end of the file in order to make it % 256.

Send the file to Metis as follows:

<11:22:33:44:55:66 ><From MAC address> 0xEFFE, 0x03 <0x0A> < # blocks> <256 bytes of data>

where # blocks = 4 bytes (MSB…LSB), number of 256 byte blocks that will be sent.

Metis will respond with

< To MAC address ><11:22:33:44:55:66 > 0xEFFE, 0x03, 0x02 <42 bytes of 0x00>

which is a 'send more' request, repeat the above until the file has been sent.

Step 5. Cycle the power to the boards whereby the new code will be loaded into the target FPGA.

**Example:**

The following example indicates what LEDs will be lit on a Mercury board when programming the board's EPCS16 using the Metis JTAG programmer.

Once Mercury_JTAG.rbf has been loaded into the Mercury board LEDs 5 and 6 will be on and LED 4 flashing.

During the erase process LEDs 8, 7, 6 and 5 will be on and LED 4 flashing.

After a successful erase LEDs 9, 6 and 5 will be on and LED 4 flashing.

During the program process LEDs 10, 9, 8, 7, 6, 5 will be flickering and LED 4 flashing.

After successfully programming LEDs 9, 6, 5 will be on and LED 4 flashing.

APPENDIX A - Formatting a Metis file to include the Bootloader.

This process needs to be followed whenever the Bootloader code is changed or the user wishes to load new Metis code directly into the EPCS16, using a USB or Byte Blaster, rather than using the HPSDRProgrammer application.

If this process is not followed then loading code into the EPCS16 starting at address 0x0 will overwrite the Bootloader code.

The Altera Remote Update Function in the Cyclone III is used to select between two images loaded into the EPCS16. The first image is called the Factory Image and the second image is called the Application Image. The Cyclone will always load the Factory Image first, then, by using the Remote Update Mega Function, and depending on the state of an FPGA I/O pin, we can have the Cyclone III load the Application image.

In the case of Metis the Factory Image corresponds to the Bootloader code and the Application Image corresponds to the Metis code. The I/O pin corresponds to the presence or absence of a jumper on JP1. If a jumper is fitted then Metis will run the Bootloader code and if absent will run the Metis code.

The Bootloader code is loaded into the EPCS16 starting at address 0x0. The Metis code is loaded starting at address 0x10000 i.e. at an address equivalent to 1MB.

We need to produce a single *.pof file that contains both the Bootloader and Metis code.

This is done using the Quartus Convert Programming Files utility i.e.

File > Convert Programming Files

The Metis source files in SVN

(svn://svn.openhpsdr.org/svn/repos_sdr_hpsdr/trunk/Metis/Source)

includes a file called Metis.cof which can be used to automate combining the files if desired rather than create a new *.pof file from scratch. To reuse an existing Metis.cof file click on Open Conversion Setup Data and select the file Metis.cof. Then click Generate which will create an output_file.pof (default name) that contains both the Bootloader and Metis code that can be used to program the EPCS16 using a USB Blaster etc.

To produce a new combined *.pof file from scratch proceeded as follows.

To add the Bootloader.sof file, click on 'SOF Data', add File, then select Bootloader.sof. Click Properties and under 'Address mode for selected pages' select 'Start' and set the start address to 0x0 and the Page to 0.

To add the Metis.sof file, click Add SofData, click on SOF Data and then Add File.

Select the Metis.sof file. Then click on Properties, select Page 1 and a start address to match that contained in the Bootloader.v file i.e. 0x00100000.

Right click on each of the *.sof file names then click Properties. Select compression for each file. This is necessary since otherwise the files are too large to fit into the EPCS16.

Finally click Generate which will create an output_file.pof (default name) that can be used to program the EPCS16 using a USB Blaster etc.

Quartus will generate a text file called 'output_file.map'. This can be read to check that the two files are allocated to the correct addresses.

These settings can be saved, and recalled for use again, by clicking on Save Conversion Setup.

This will create a *.cof file (Factory&Application.cof in this example). To reload a *.cof file click on Open Conversion Setup Data and select the file.

NOTES
---

Each time either the Application or Factory file is changed then a new output_file.pof has to be generated using the above steps. The simplest method is to use the Open Conversion Setup Data and select the previously saved*.cof file.

ENDS.