Design Note

Hardware

Rev. 1.01

2009/04/20

Revision History:

Revision	Date	Comment
0.90	2008/05/22	Initial version
0.91	2008/05/30	Add hopelite resource
0.92	2008/06/11	Others in Aero
0.93	2008/06/24	Production Test in Aero
0.94	2008/08/07	HPI interface
0.95	2008/08/27	Add Web menu item description
0.96	2008/09/10	Add multi-port access method get_port_data usage Translate Chinese into English
0.97	2008/09/24	Page layout modification get_eoc_data description set_eoc_data description
0.98	2008/11/05	Modify section 2.3.1.2.3 MIB setting comment
0.99	2009/01/21	Add HPI packet simple description
1.00	2009/02/06	Add EOC using sample
1.01	2009/04/20	Add VDSL chip firmware downloading procedure

Contents

1 Introduction

1.1 Scope

The purpose of the design note is helping HW engineers to design and verification the hardware.

2 Design note

2.1 Aero

2.1.1 Design phrase

2.1.1.1 Get the reference design

Get the reference design. The reference design should include schematic, layout gerber, and database.

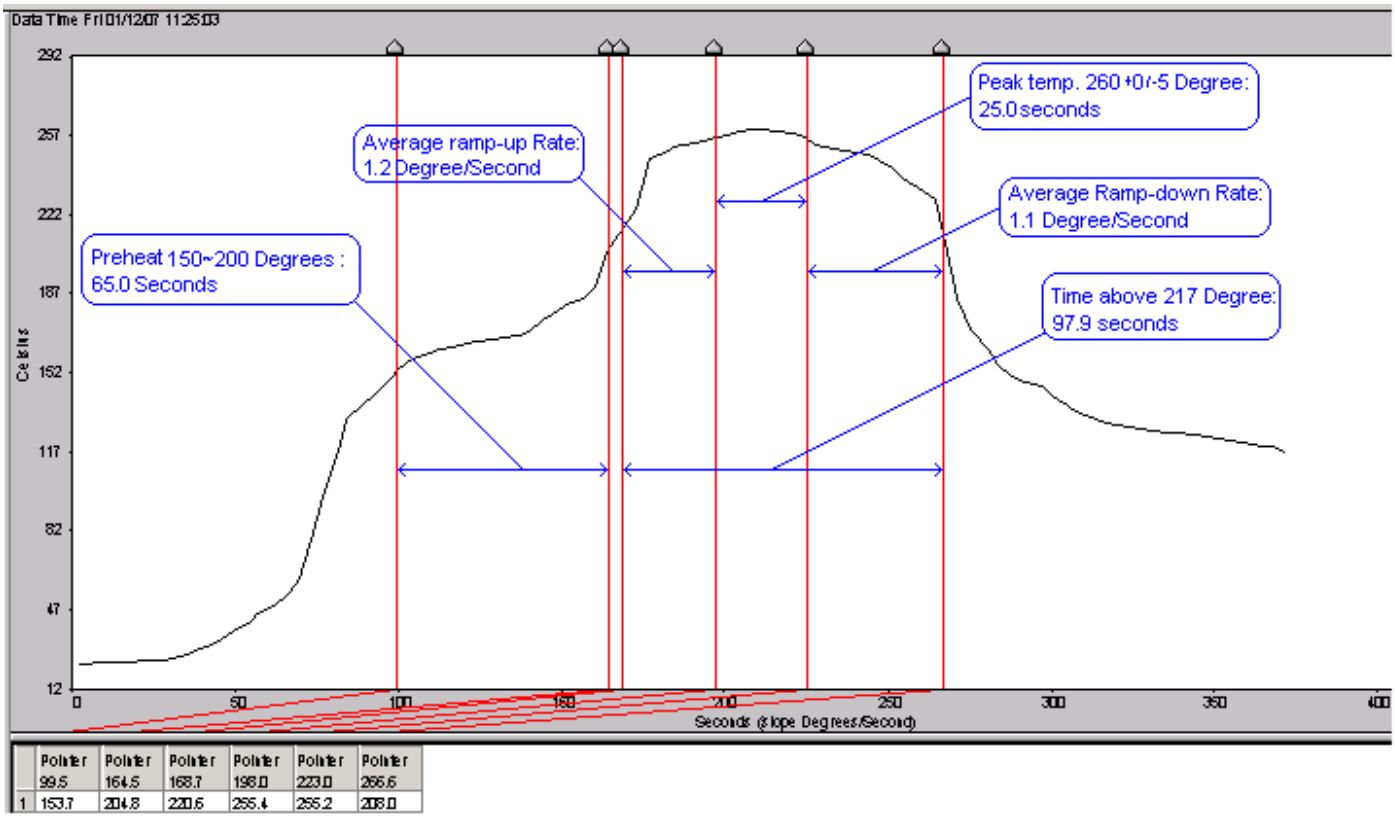
2.1.1.2 Schematic

2.1.1.3 Layout

Please follow metanoia_layout_guide.pdf. Any violation will impact performance, please follow the suggestion or discussion with metanoia HW enginner.

2.1.1.4 SMT

Please follow this reflow chart on SMT procedure

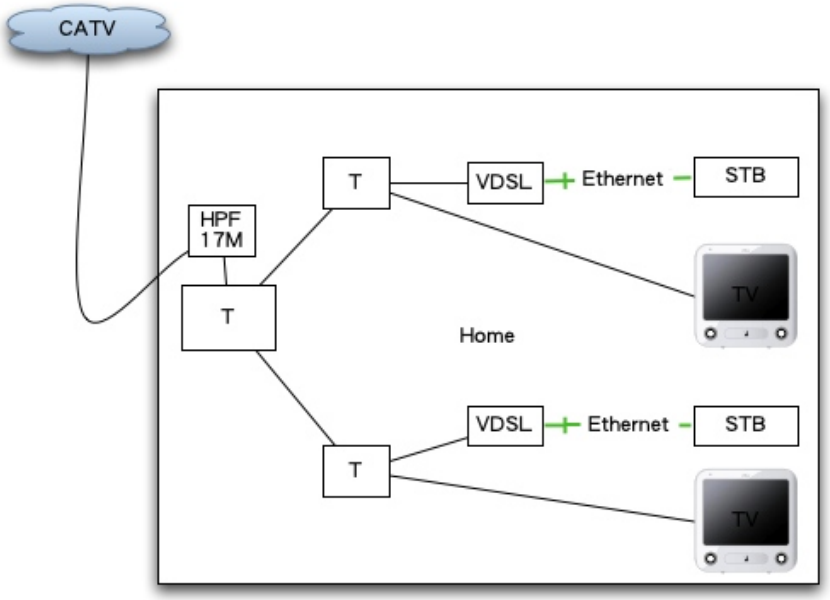


2.1.1.5 Others

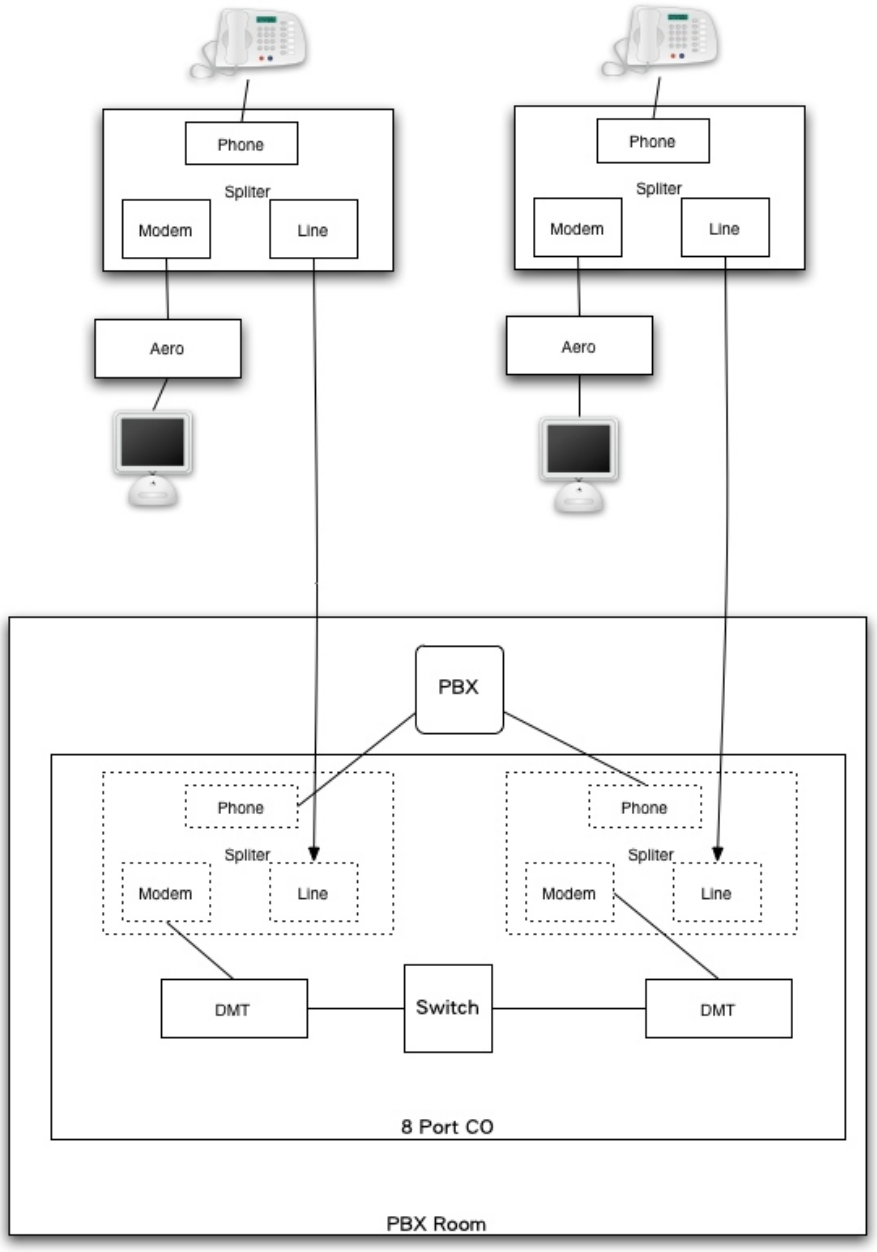
2.1.1.5.1 Data and analog signal in the same cable

With CATV signal

The limitation is P2P and use at home



2.1.1.5.2 Deployment in PBX environment



2.1.2 Verification phrase

2.1.2.1 Function test

Check LED：

Procedure：

1.Connect the Power to DUT。

2.ETH1's and ETH2's LED will be turn off, then turn off→PASS

3.Power LED: always turn on→PASS

4.DIP_sw_1 set OFF, the Slave LED turn ON and the Master LED turn OFF→PASS
5.DIP_sw_1 set On, the Slave LED turn OFF and the Master LED turn ON→PASS
6.VDSL LED blinking slowly→PASS
7.When packets through the DUT, the ETH1's and ETH2's LED blinking fast→PASS

Check DIP SWITCH：

Procedure：

1. Connect the Power to DUT。
2.The DUT's DIP switch setting is 0000，check Dip_Setting=CO_000 on DslMonitor
DIP switch setting is 1111，check Dip_Setting=CPE_111 on DslMontior→PASS

Power consumption test

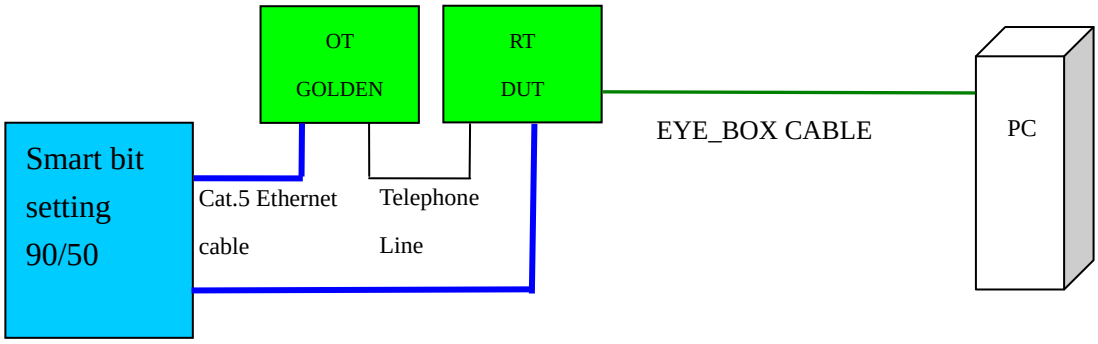
Procedure：

1.When connect 5V to DUT，the current should be under 600mA→PASS
2.The DUT link with OT Golden via VDSL，the current should be under 650mA→PASS
3.The DUT link with OT Golden via VDSL and Ethernet，the current should be under 860mA →PASS

How to fixed the bad DUT：

1.POW LED：This is power LED，if this LED not be lighted, you can check the U7、C1 have 3.3V output or not。
2.ETH1,ETH2 LED：Ethernet link LED，If the LED light off after connect Cat.5 ehternet，you can check R11 have 25MHz clock or not，and check D2 have 1.8V or not, or RJ1/RJ2 have not good solder.
3.VDSL LED：This is VDSL LED，If not blinking, you can measure TP5/TP6 have 35.328MHz clock or not ,C73/C76 have 1.2V or not，or U4 have not FW，or U2 have not good solder.
4.DIP switch：If this function abnormality, you can switch the dip1/2/3/4 check it can be 0V or 3.3V,or U2 have not good solder。
5.power consumption：If the current abnormality, you can measure1.2V/2.5V/3.3V/5V have been short with GND or not.

2.1.2.2 VDSL and Ethernet test



Procedure：

1.Power on check the current is right or not。

2.OT GOLDEN connect with DUT via telephone line，using EYE_BOX TOOL connect to PC and another side connect to DUT。

3.Supply the power to OT GOLDEN and DUT，the VDSL LED will be turn on within 1min.

Link LED turn on →PASS

Link LED still blinking over 1 min→Fail。

4. The massage on DSL Monitor

US_Rate>=55000,DS_Rate>=95000→PASS

VCXO should be 400~A00→PASS

5.Smart bit setting the DS 90M，US 50M, then start transmit 200,000 pkts.

a. Packet loss=0→PASS

b. Packet loss <1 or 2 pkts，resent 200,000 pkts, if Packet loss=0→PASS

c. Packet loss>0→Fail

6.Set DUT as OT and link with RT GOLDEN, then go step 5。

The bad DUT analysis：

1.The VDSL can't Link：check TP5/TP6 have 35.328MHz or not，C48 have 2.5V or not，If it have not clock and power, it need re-solder U3。Or check the T2 have not good solder.

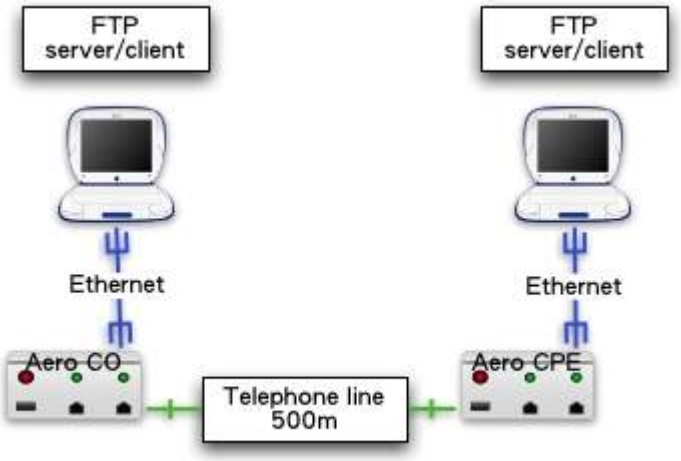
2.Packet loss：check the DSL monitor to see the linerate have DS/US=110/60M or not.

2.1.3 Production phrase

2.1.3.1 Quick test

Please follow 2.1.2.2 to verify hardware component (crystal) , attainable throughput, and data path.

2.1.3.2 SmartBit/FTP performance at short distance

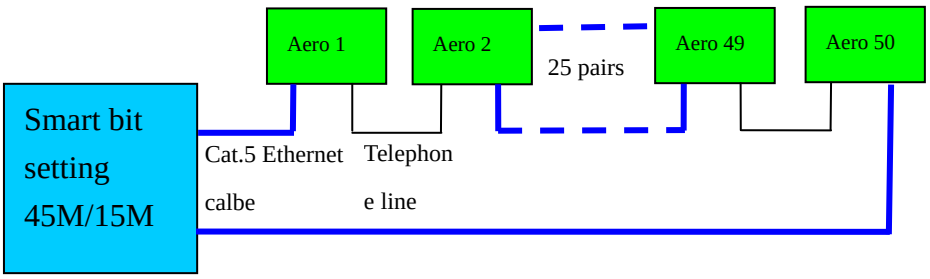


Using packet generator to test is better. Or you can use FTP to test
Using FTP to upload/download file at the same time. Performance base line is different by setup of ftp server and OS and computer performance. You need to test first to get the base performance base line.

2.1.3.3 SmartBit/FTP performance at long distance

The same test with 2.1.3.2. Only the distance is different. The reference distance is 1000m

2.1.3.4 Burn In test



Purpose: Reliability test at Room temp.

Test setup: test condition you can see the above diagram ，there are 25 pairs aero that connect together，then using smart bit to transmit pkts.

Procedure:

1.Smart bit setting the DS=45M，US=15M。

2. set the dip switch to 0001 in even Aero, 1001 in odd Aero.

3.suply power to the 25 pairs Aero simultaneously, check the LK LED in one minute.

A. LK LED always shine → PASS

B. LK LED does not always shine after one minute → FAIL

4.trasmit the packets by SmartBit, check the packet loss after one hour.

C. Packet loss = 0 → PASS

D. Packet loss > 0 → FAIL

Bad item analyze:

1. after long time test, MT3201 (U3) becomes bad contact: resolder the MT3201.
2. solder ball short circuit: check if there is solder ball causes short circuit.

2.2 HopeLite

A multi-port solution use 8051 as CPU

2.2.1 Design phrase

2.2.1.1 Get the reference design

We provide Hope HW schematic, Awardlite HW schematic.CPLD source code. Awardlite reference design guide. After the hardware ready, we provide production grade 8051 source code for customer.After that customer can easily modify it to fit customer requirement

2.2.2 Verification phrase

Hardware interface verification can be done by Metanoia

2.2.3 Production phrase

2.3 HPI Interface

HPI is the only interface that work with CPU/Micro-Controller that can provide download firmware code, setup configuration, get the status. HPI is a management interface. There is no packet data pass through this interface. To get this work in different design, hardware/software need to work correctly. Here is the anything you need when you work on HPI.

2.3.1 Design phrase

2.3.1.1 Hardware

2.3.1.1.1 Get the reference design

We provide Hope HW schematic, CPLD source code. reference design guide. After the hardware ready, we

provide production grade driver source code for customer. After that customer can easily modify it to fit customer requirement

2.3.1.1.2 Host Port overview

The host port interface (HPI) is an eight/sixteen bit parallel interface with a DMA control. The HPI supports both master and slave operation. The HPI can interface with most of the Microprocessor and DSP available on the market, and this with no external components (glue logic).

The Metanoia Host Port implementation is a Master or Slave mode transferring a message length not exceeding 512 bytes per transaction.

Hereunder is an example of connection between the MT2201 and a Host controler.

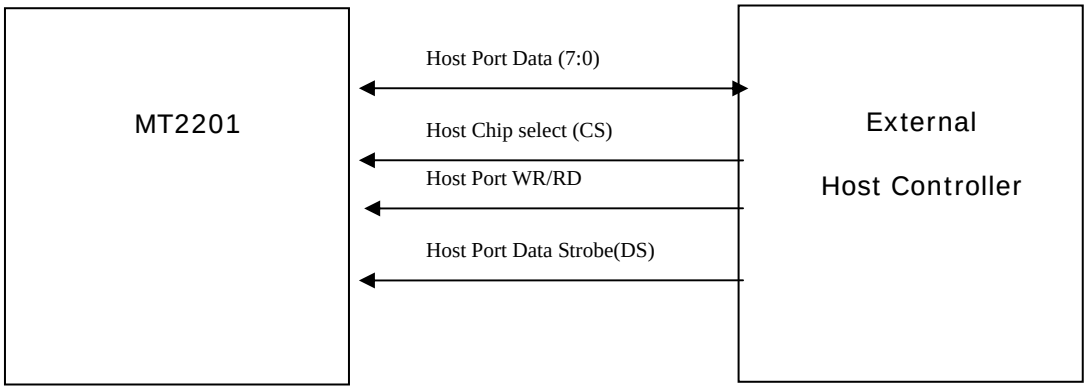


Figure5: Example of MT2201 Host Port Interface to Host Controler

2.3.1.1.3 Host Port pin description

The function of the Host port pins is described in the following table.

Pin Name	Function		Pad Typ
HP_CSn	Host Port Chip Select	Low	I
HP_DSn	Host Port Data Strobe		I
HP_RDWRn	Host Port Read/Write#		I
HP_DY	Host Port Slave Ready		I/O
EX_D8	Host Port Data 0		I/O
EX_D9	Host Port Data 1		I/O
EX_D10	Host Port Data 2		I/O
EX_D11	Host Port Data 3		I/O
EX_D12	Host Port Data 4		I/O
EX_D13	Host Port Data 5		I/O
EX_D14	Host Port Data 6		I/O

EX_D15	Host Port Data 7		I/O

2.3.1.1.4 Host Port description

The HPI interface is asynchronous, and is transfering message of a length not exceeding 512 bytes per transaction.

Each command is followed by an acknowledge command typically within 1ms. If a command is not acknowledged within 10ms the command can be assumed to be not understood or some other error condition preventing the command can be assumed. Some commands will be acknowledged and then will be followed by a status frame sent from the MT2201 to the host at a later time. If the command was a connect command or an inquiry for a distant end EOC message the status can occure several seconds after the command was issued. These special case commands are documented in the following charts.

2.3.1.1.5 HPI timing

Our timing request:

Figure 1 is showing some timing diagram for a read operation using the Host interface, in a 8 bit mode.

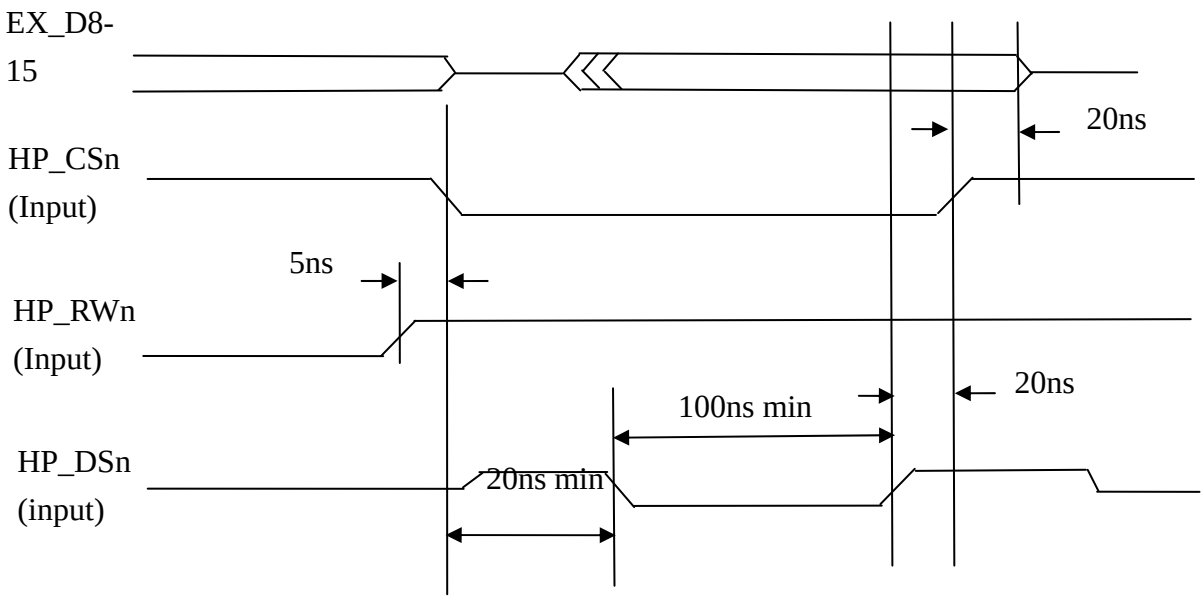
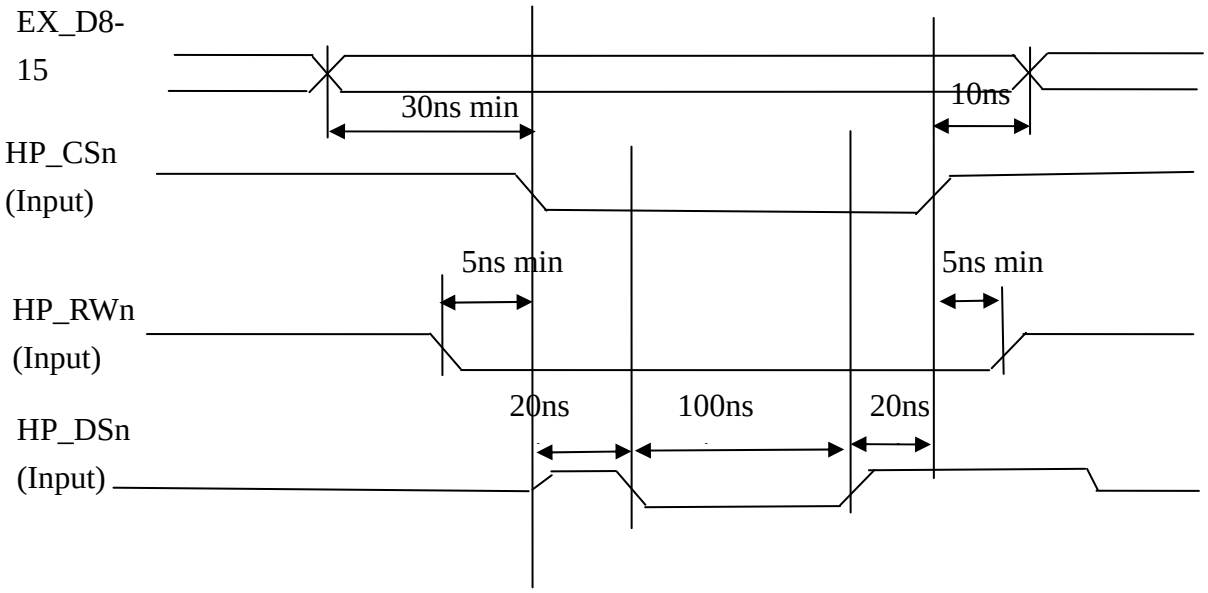


Figure 2 is showing some timing diagram for a write operation using the Host interface, in a 8 bit mode.



STAR 910X timing request:

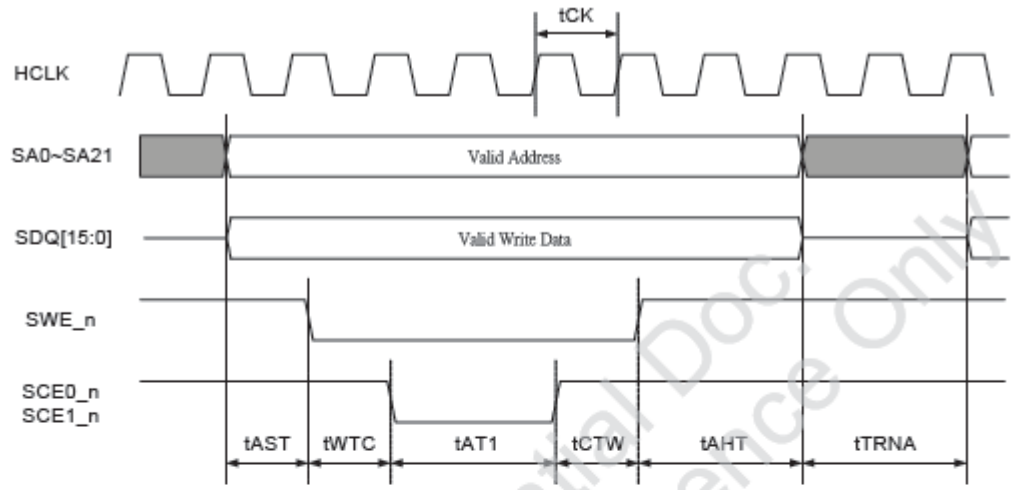


Figure 4-4. Flash/SRAM Interface Write Timing

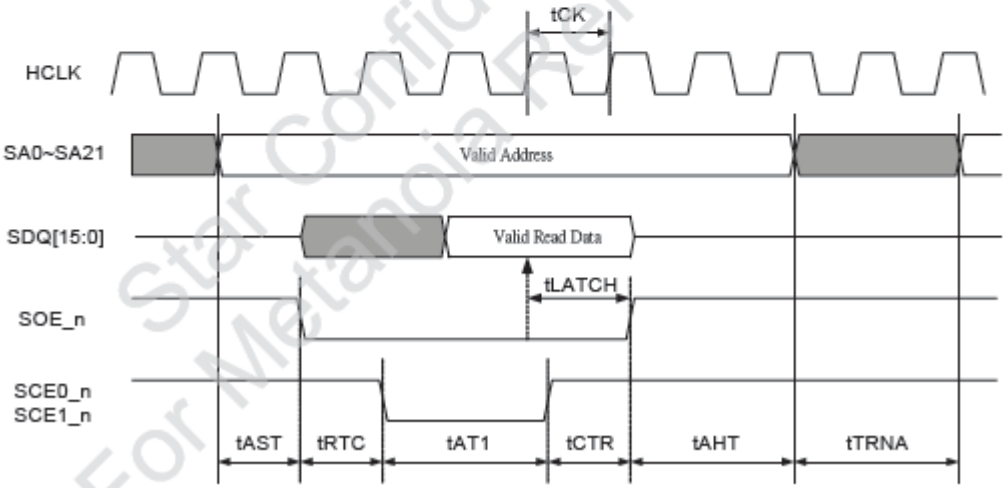


Figure 4-5. Flash/SRAM Interface Read Timing

Flash/SRAM Interface						
Access setup time	*tAST*	-	1~4	-	tCK	*3, 4*
Write-enable to chip-enable delay	*tWTC*	-	0~4	-	tCK	*3, 5*
Read-enable to chip-enable delay	*tRTC*	-	0~4	-	tCK	*3, 5*
Access time	*tAT1*	-	1~256	-	tCK	*3, 6*
Chip-disable to write-disable delay	*tCTW*	-	0~4	-	tCK	*3, 7*
Chip-disable to read-disable delay	*tCTR*	-	-1~3	-	tCK	*3, 7*
Address hold time	*tAHT*	-	1~4	-	tCK	*3, 8*
Turn-around time	*tTRAN*	-	1~256	-	tCK	*3, 9*
Read data latch time before SOE_n de-assert	*tLATCH*	-	1	-	tCK	*3*

2.3.1.1.6 Other hardware signal requirement when accessing HPI

1. Reset signal.

We need to reset the MT2201 before download firmware to it. There should be a reset signal to every MT2201 because sometimes we might download to some port. The time interval should be longer then 1ms between reset and download firmware.

2. Symbol Alignment.

This signal is use for synchronize the VDSL, This is using in bundle line. So we need a synchronous signal from CPLD to pin41(IRQB) of MT2201. This signal is 4Khz, CLK source is 35.328Mhz. This signal only have 1 CLK cycle LOW, others will be the HIGH. The CPLD's code listed below:

```
if    pacing_count ="10001001111111" then ------(8832-1)
        buf_pacing_4k <='0';
        pacing_count<="00000000000000";
      else
        pacing_count <= pacing_count+1;
        buf_pacing_4k <='1';
      end if;

      if    pacing_count1 ="1111111111111" then
        buf_pacing_led <='1';
        pacing_count1<="0000000000000";
      else
        pacing_count1 <= pacing_count1+1;
        buf_pacing_led <='0';
      end if;
```

3. Led Sync.

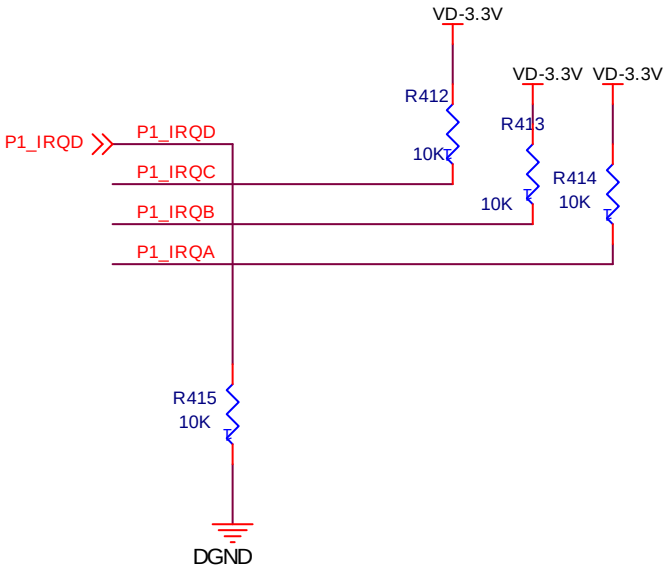
Because we will do re-link the VDSL or reset to MT2201, and cause the VDSL Link LED will be asynchronous, so we need a synchronous signal from CPLD to pin28(EX_a16) of MT2201. This signal is 4.3125Khz, and the CLK source is 35.328Mhz. The CPLD's code listed below:

```
if   pacing_count2 ="1111111111111" then ------------(8192-1)
```

```
            buf_pacing_led1 <='1';
            pacing_count2<="0000000000000";
        else
            pacing_count2 <= pacing_count2+1;
            buf_pacing_led1 <='0';
        end if;
```

4. Boot mode setting.

Boot mode is the hardware setting, IRQA, IRQB, IRQC and IRQD. We should set the IRQA,B,C,D={1,1,1,0} as the figure below. It is the hardware reference schematic.



5. Watch Dog GPIO.

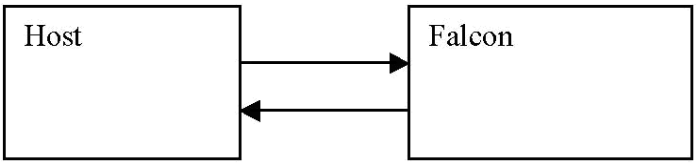
Optional function at DMT pin29(EX_A17/GPIO2). Only use on 8051 with single port VDSL platform.

2.3.1.2 Software

2.3.1.2.1 What we provide

We provide 8051 CLI source code including driver and management functions.
We provide Linux device driver for kernel 2.4, if customer use our reference design. If customer changes any hardware, we provide driver source code, CLI sample code, and test code. Customer in normal business model should do source code integration.

2.3.1.2.2 HPI software access timing and protocol



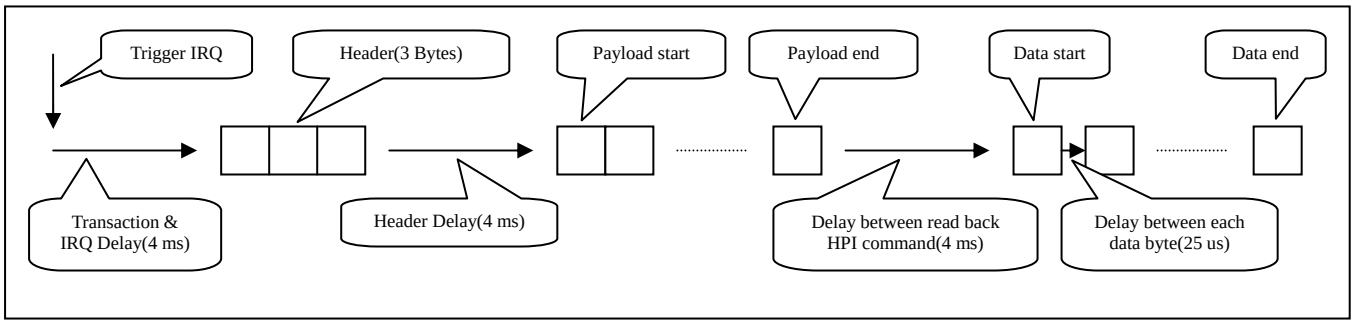
The host-dsp interface for Falcon uses upto 512-byte packets, to exchange control information. The first 3 bytes of each packet are reserved for header, and the rest of the bytes can be used for payload. The nine LSBs of the first 2-bytes of the header are used to indicate the number of bytes in the complete packet (PktLen). The third byte is used to communicate the port number (PortNum) corresponding to the payload data.

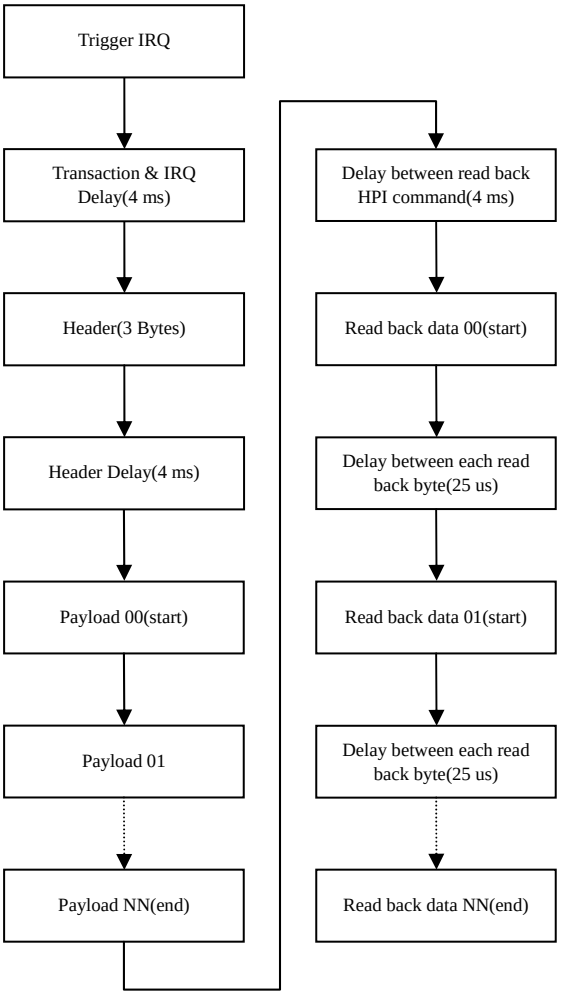
PktLen	PortNum	Payload

The payload uses HDLC-like framing, similar to the one used in G.997.1 as shown in the diagram. The payload can be used to:

1. Send and Receive EOC frames between the near-end and far-end Management Entity.
2. Send Host to Dsp Commands and Receive Dsp to Host Responses / Status information.

Following is the detail description of using HPI:
1. To make DMT response from HPI, we need to download firmware to DMT first. When downloading firmware through HPI, we need some delay (minimum 5us) between each byte.
2. After firmware downloading, we can communicate with DMT through HPI by HPI software protocol. HPI software protocol timing issue is shown by the following two figures. We need to take care of every required delay in different platform. (IRQ trigger delay (including transaction delay between two different HPI commands), 3 bytes header delay, delay between read back HPI command, read back data delay) All these delays are represented in driver level.





3. The entire process to make DMT working properly is shown as followed:
download firmware → wait 500 ms → apply VDSL MIB settings → issue connect command to DMT

4. Changing VDSL MIB settings online should follow the following procedure:
apply new VDSL MIB settings → issue disconnect command to DMT → issue connect command to DMT

2.3.1.2.3 VDSL Firmware download

It depends on the platform we used. We provide two different VDSL firmware, OT and RT firmware. We can only download one firmware into the DMT at a time. If the platform is installed with a file system supported OS, we can put these two raw VDSL firmwares into the file system directly and download one of the VDSL firmwares from first byte to last byte into DMT through HPI. After downloading, the DMT will start working. If we're using a simple platform with a micro-controller but without OS installed, we can flash the raw VDSL

firmware we provided into flash memory in advance. We can program the size of both firmwares in some location of flash memory or hard code the size in the micro-controller program code for later downloading. Also, we choose one of the OT and RT firmware, acquire the size of the firmware we chose and download the firmware data byte by byte into DMT from the firmware starting address in flash memory to the starting address plus firmware size.

2.3.1.2.4 MIB settings

After downloading firmware, we can apply user settings by using VDSL MIBs. Setup user settings including bandplan settings, target SNR, rate limit, interleave delay and many others. How many features need to be supported should be discuss with metanoia in order to meet the requirement of customer. We provide some basic features here.

All VDSL MIBs settings can be applied with “set_port_data()” API we provided and all the settings we would like to change must be setup before issue connect or disconnect command.

Band plan related – we have 3 different plan, 997/998/symmetric, different bandplan have different bandwidth behavior. Here is the detail:

BandPlan 997 with ISDN support
#vdslLineConfBandPlan
set_port_data 1 0 10 0 31 9
#vdslLineMCMConfProfileRxBandStart1
set_port_data 1 1 3 0 2 34
#vdslLineMCMConfProfileRxBandStop1
set_port_data 1 1 3 0 3 65
#vdslLineMCMConfProfileRxBandStart2
set_port_data 1 1 3 1 2 698
#vdslLineMCMConfProfileRxBandStop2
set_port_data 1 1 3 1 3 1179
#vdslLineMCMConfProfileRxBandStart3
set_port_data 1 1 3 2 2 1640
#vdslLineMCMConfProfileRxBandStop3
set_port_data 1 1 3 2 3 2781
#vdslLineMCMConfProfileRxBandStart4
set_port_data 1 1 3 3 2 0
#vdslLineMCMConfProfileRxBandStop4
set_port_data 1 1 3 3 3 0
#vdslLineMCMConfProfileTxBandStart1

set_port_data 1 1 2 0 2 66
#vdslLineMCMConfProfileTxBandStop1
set_port_data 1 1 2 0 3 691
#vdslLineMCMConfProfileTxBandStart2
set_port_data 1 1 2 1 2 1186
#vdslLineMCMConfProfileTxBandStop2
set_port_data 1 1 2 1 3 1631
#vdslLineMCMConfProfileTxBandStart3
set_port_data 1 1 2 2 2 2782
#vdslLineMCMConfProfileTxBandStop3
set_port_data 1 1 2 2 3 3717
#vdslLineMCMConfProfileTxBandStart4
set_port_data 1 1 2 3 2 0
#vdslLineMCMConfProfileTxBandStop4
set_port_data 1 1 2 3 3 0
#vdslMetanoiaRxPgaRefOt
set_port_data 1 2 1 0 47 273
#vdslMetanoiaRxPgaRefRt
set_port_data 1 2 1 0 48 275
#vdslTwConfig
set_port_data 1 2 1 0 49 10826
#vdslTwConfig2
set_port_data 1 2 1 0 50 4

Band plan 998 with ISDN support
vdslLineConfBandPlan
vdsl set mibentry 1 0 10 0 31 8
#vdslLineMCMConfProfileRxBandStart1
vdsl set mibentry 1 1 3 0 2 34
#vdslLineMCMConfProfileRxBandStop1
vdsl set mibentry 1 1 3 0 3 65
#vdslLineMCMConfProfileRxBandStart2
vdsl set mibentry 1 1 3 1 2 872
#vdslLineMCMConfProfileRxBandStop2
vdsl set mibentry 1 1 3 1 3 1207
#vdslLineMCMConfProfileRxBandStart3
vdsl set mibentry 1 1 3 2 2 1972
#vdslLineMCMConfProfileRxBandStop3

vdsl set mibentry 1 1 3 2 3 2787
#vdslLineMCMConfProfileRxBandStart4
vdsl set mibentry 1 1 3 3 2 0
#vdslLineMCMConfProfileRxBandStop4
vdsl set mibentry 1 1 3 3 3 0
#vdslLineMCMConfProfileTxBandStart1
vdsl set mibentry 1 1 2 0 2 66
#vdslLineMCMConfProfileTxBandStop1
vdsl set mibentry 1 1 2 0 3 865
#vdslLineMCMConfProfileTxBandStart2
vdsl set mibentry 1 1 2 1 2 1208
#vdslLineMCMConfProfileTxBandStop2
vdsl set mibentry 1 1 2 1 3 1967
#vdslLineMCMConfProfileTxBandStart3
vdsl set mibentry 1 1 2 2 2 2788
#vdslLineMCMConfProfileTxBandStop3
vdsl set mibentry 1 1 2 2 3 3219
#vdslLineMCMConfProfileTxBandStart4
vdsl set mibentry 1 1 2 3 2 0
#vdslLineMCMConfProfileTxBandStop4
vdsl set mibentry 1 1 2 3 3 0
vdslMetanoiaRxPgaRefOt
vdsl set mibentry 1 2 1 0 47 273
vdslMetanoiaRxPgaRefRt
vdsl set mibentry 1 2 1 0 48 275
#vdslTwConfig2
vdsl set mibentry 1 2 1 0 50 4

Bandplan symmetric with ISDN supported
#bandplan ID
vdsl set mibentry 1 0 10 0 31 11
#vdslLineMCMConfProfileRxBandStart1
vdsl set mibentry 1 1 3 0 2 34
#vdslLineMCMConfProfileRxBandStop1
vdsl set mibentry 1 1 3 0 3 65
#vdslLineMCMConfProfileRxBandStart2
vdsl set mibentry 1 1 3 1 2 616
#vdslLineMCMConfProfileRxBandStop2

vdsl set mibentry 1 1 3 1 3 1207
#vdslLineMCMConfProfileRxBandStart3
vdsl set mibentry 1 1 3 2 2 1972
#vdslLineMCMConfProfileRxBandStop3
vdsl set mibentry 1 1 3 2 3 2787
#vdslLineMCMConfProfileRxBandStart4
vdsl set mibentry 1 1 3 3 2 3230
#vdslLineMCMConfProfileRxBandStop4
vdsl set mibentry 1 1 3 3 3 3761
#vdslLineMCMConfProfileTxBandStart1
vdsl set mibentry 1 1 2 0 2 66
#vdslLineMCMConfProfileTxBandStop1
vdsl set mibentry 1 1 2 0 3 609
#vdslLineMCMConfProfileTxBandStart2
vdsl set mibentry 1 1 2 1 2 1208
#vdslLineMCMConfProfileTxBandStop2
vdsl set mibentry 1 1 2 1 3 1967
#vdslLineMCMConfProfileTxBandStart3
vdsl set mibentry 1 1 2 2 2 2788
#vdslLineMCMConfProfileTxBandStop3
vdsl set mibentry 1 1 2 2 3 3219
#vdslLineMCMConfProfileTxBandStart4
vdsl set mibentry 1 1 2 3 2 3804
#vdslLineMCMConfProfileTxBandStop4
vdsl set mibentry 1 1 2 3 3 4059
vdslMetanoiaRxPgaRefOt
vdsl set mibentry 1 2 1 0 47 273
vdslMetanoiaRxPgaRefRt
vdsl set mibentry 1 2 1 0 48 275
#vdslTwConfig2
vdsl set mibentry 1 2 1 0 50 4

Target SNR related
vdslLineConfDownTargetSnrMgn
set_port_data 1 0 10 0 8 36
vdslLineConfUpTargetSnrMgn
set_port_data 1 0 10 0 11 36

Limit data rate related
#vdslLineConfDownSlowMaxDataRate
set_port_data 1 0 10 0 14 131072
#vdslLineConfUpSlowMaxDataRate
set_port_data 1 0 10 0 18 131072

Max delay related
#vdslLineConfDownMaxInterDelay
set_port_data 1 0 10 0 22 0
#vdslLineConfUpMaxInterDelay
set_port_data 1 0 10 0 23 0

2.3.1.2.5 Data Exchange by EOC

We provide two APIs for exchanging data by EOC(Embedded Operations Channel).

Receiving data:

get_eoc_data(unsigned char chipid, unsigned char *buffer, unsigned char *recvdatalen);

Transmitting data:

set_eoc_data(unsigned char chipid, unsigned char *buffer, unsigned char datalen);

There are 12 bytes overhead when exchanging data, we need to be aware of it especially in 8051 MCU based board.

In the 8051 reference design, we use the following data structure for I/O with DMT by HPI.

```
#define MT_MAX_BUFFERSIZE 24
typedef struct port_struct
{
    unsigned char lastcmdid;
    unsigned int value;
    unsigned char buf[MT_MAX_BUFFERSIZE];
}port_t;
```

In this case, while calling the get_eoc_data() or set_eoc_data(), we can only exchange 12 bytes raw data at one time. That is, we can make a function call like this:

Host side transmit the data directly:

unsigned char buffer[12] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c};

set_eoc_data(chipid, buffer, 12);

Remote side receive the data by checking an data arrived indication MIB first:

```
unsigned char value;
unsigned char recvdatalen;     //length of the received data by EOC
unsigned char eocrecv [12];
get_port_data(MT_MIB_VdslMetanoiaGroup, MT_MIB_VdslMetanoiaTable, MT_IDX_NEAR,
                MT_MIB_VdslTerminal, &value);
if(value & 0x00000400) {      //Data arrived
      get_eoc_data(chipid, eocrecv, &recvdatalen);
}
```

After calling get_eoc_data(), The eocrecv will contain the data that Host side sent and recvdatalen will be 12. If we made a modification with “MT_MAX_BUFFERSIZE”, let’s say 32. Then we can declare “buffer” from “unsigned char buffer[12]” to “unsigned char buffer[20]” and exchange 20 bytes data in one time.

2.3.1.2.6 User Interface

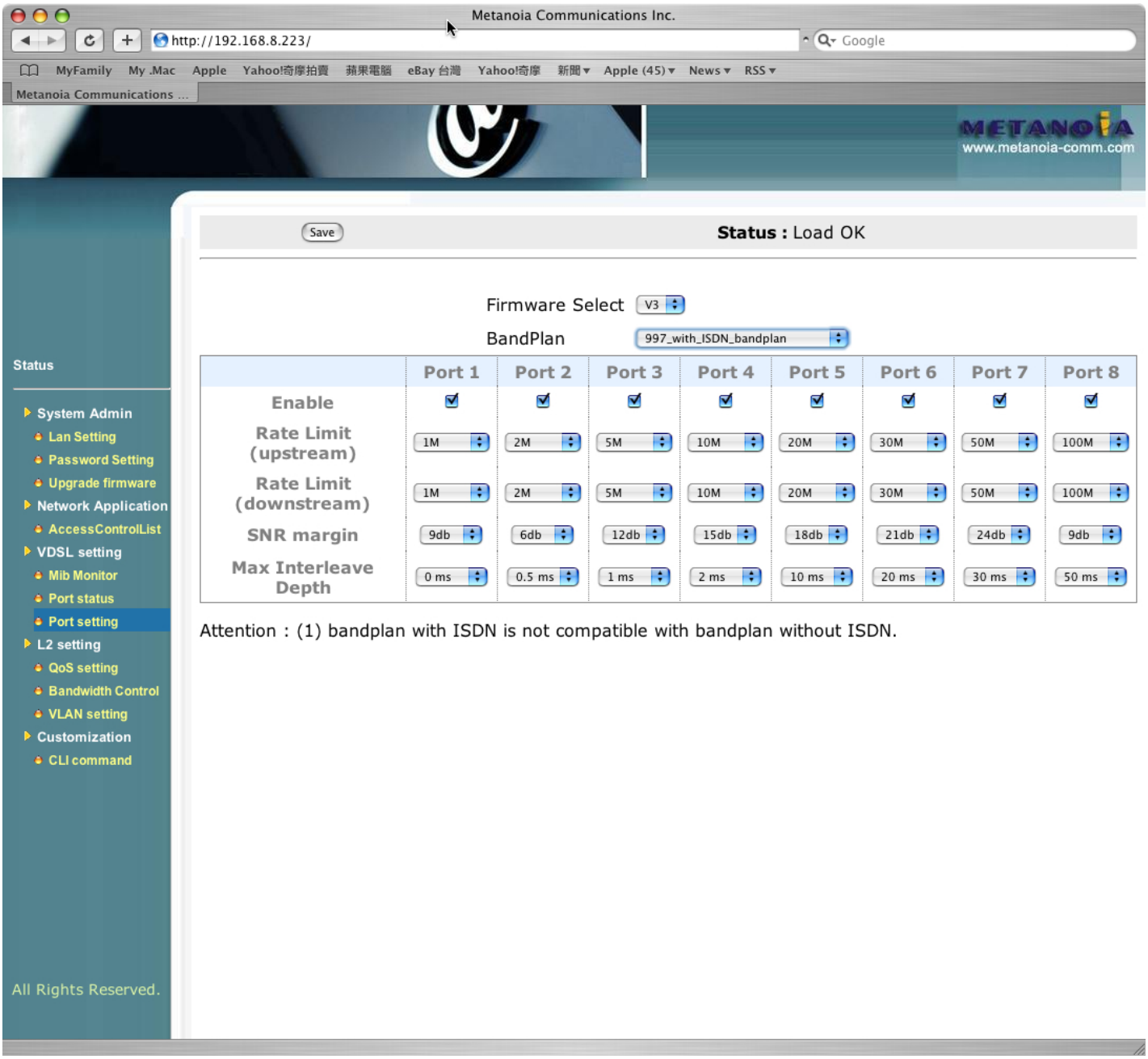
User interface can be CLI or Web, the features need to have VDSL setting, VDSL status, firmware upgrade. Advanced features can be discuss with metanoia

CLI:

```
xDSL>?
help or ?                 - print CLI command help
setting                   - print system settings
exit                      - exit CLI
password                  - change CLI login password
factory                   - reset all settings to factory default
upgrade [fw|conf]         - firmware/settings upgrade
xdsl enable [on|off]      - enable/disable xDSL port
xdsl load [1|2] [co|cpe]  - load xDSL Firmware
xdsl reset                - reset xDSL chip
xdsl retrain              - force xDSL retrain
xdsl param                - print xDSL parameters information
xdsl show                 - print xDSL settings
xdsl profile (profile)    - change/print xDSL profile
xdsl upspeed [i]          - change/print xDSL uprate(2000 - 131072 Kbps)
xdsl downspeed [j]        - change/print xDSL downrate(2000 - 131072 Kbps)
xdsl snrm [n]             - change/print xDSL SNR margin(0 - 24 dB)
xdsl delay [n]            - change/print xDSL delay(0 - 100 0.5ms)
net auto [on|off]         - change/print ethernet auto negotiation mode
net speed (speed)         - change/print ethernet speed(10 or 100 Mbps)
net duplex [full|half]    - change/print ethernet duplex mode
net flowctrl [on|off]     - change/print ethernet flow control
net show                  - print ethernet counters
net clean                 - clean ethernet counters
xDSL>
```

Web:

VDSL setting



How to set variables:

Bandplan

Three types of bandplan, 997, 998 and symmetric, detail bandplan setting please refer to section 2.3.1.2.3

Enable

Check to enable selected port, uncheck to disable selected port

Rate Limit(upstream)

Selectable 1M, 2M, 3M, …, 19M, 20M, 25M, 30M, 35M, …, 95M, 100M, no limit

1M means we have to set UpSlowMaxDataRate as followed

set_port_data 1 0 10 0 18 1000

100M means we have to set UpSlowMaxDataRate as followed

set_port_data 1 0 10 0 18 100000

no limit means we have to set UpSlowMaxDataRate as followed

set_port_data 1 0 10 0 18 131072

Rate Limit(downstream)

Selectable 1M, 2M, 3M, …, 19M, 20M, 25M, 30M, 35M, …, 95M, 100M, no limit

1M means we have to set DownSlowMaxDataRate as followed

set_port_data 1 0 10 0 14 1000

100M means we have to set DownSlowMaxDataRate as followed

set_port_data 1 0 10 0 14 100000

no limit means we have to set DownSlowMaxDataRate as followed

set_port_data 1 0 10 0 14 131072

SNR Margin

Selectable 6db, 9db, 12db, 15db, 18db, 21db, 24db, maximum is 24db

6db means we have to set UpTargetSnrMgn and DownTargetSnrMgn at the same time as followed

set_port_data 1 0 10 0 11 24 (our unit is 0.25db, 24*0.25db = 6db)

set_port_data 1 0 10 0 8 24

24db means we have to set UpTargetSnrMgn and DownTargetSnrMgn at the same time as followed

set_port_data 1 0 10 0 11 96 (our unit is 0.25db, 96*0.25db = 24db)

set_port_data 1 0 10 0 8 96

Max Interleave Depth

Selectable 0ms, 0.5ms, 1ms, 1.5ms, …, 4.5ms, 5ms, 6ms, 7ms, …, 19ms, 20ms, 25ms, 30ms, 35ms, 40ms, 45ms, 50ms, maximum is 50ms

0.5ms means we have to set UpMaxInterDelay and DownMaxInterDelay at the same time as followed

set_port_data 1 0 10 0 23 1 (our unit is 0.5ms, 1*0.5ms = 0.5ms)

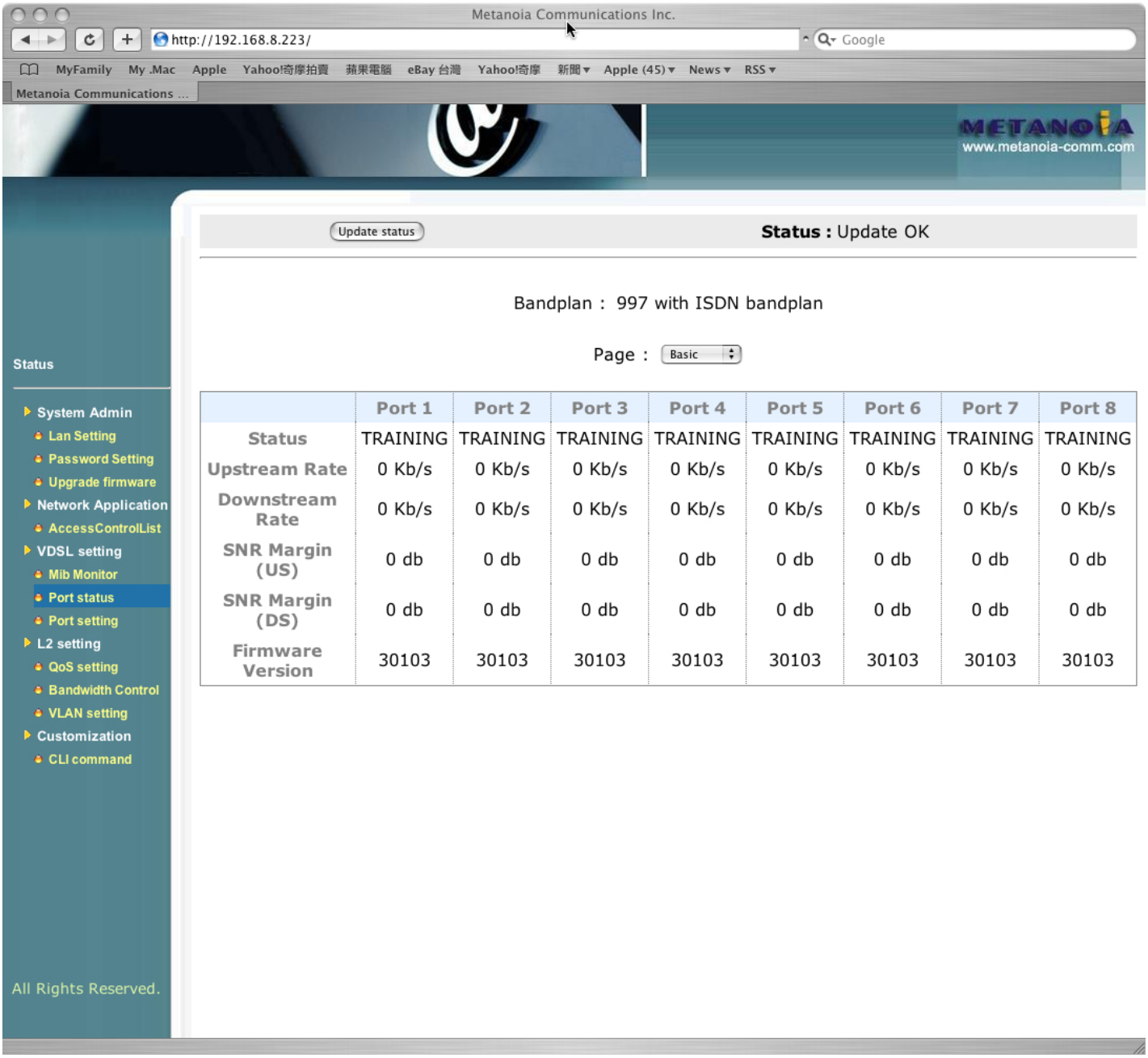
set_port_data 1 0 10 0 22 1

50ms means we have to set UpMaxInterDelay and DownMaxInterDelay at the same time as followed

set_port_data 1 0 10 0 23 100 (our unit is 0.5ms, 100*0.5ms = 50ms)

set_port_data 1 0 10 0 22 100

VDSL status basic



	Port 1	Port 2	Port 3	Port 4	Port 5	Port 6	Port 7	Port 8
Status	TRAINING	TRAINING	TRAINING	TRAINING	TRAINING	TRAINING	TRAINING	TRAINING
Upstream Rate	0 Kb/s	0 Kb/s	0 Kb/s	0 Kb/s	0 Kb/s	0 Kb/s	0 Kb/s	0 Kb/s
Downstream Rate	0 Kb/s	0 Kb/s	0 Kb/s	0 Kb/s	0 Kb/s	0 Kb/s	0 Kb/s	0 Kb/s
SNR Margin (US)	0 db	0 db	0 db	0 db	0 db	0 db	0 db	0 db
SNR Margin (DS)	0 db	0 db	0 db	0 db	0 db	0 db	0 db	0 db
Firmware Version	30103	30103	30103	30103	30103	30103	30103	30103

How to get variables:

US/DS SNR margin and Max Interleave Depth should be the same, so we just need to let user choose a value and set US/DS with the same one.

Slow Burst Protection (US)

get_port_data(chip, MT_MIB_VdslGroup, MT_MIB_VdslChanTable, MT_IDX_OTINTL, MT_MIB_VdslChanCurrTxSlowBurstProtect, &value);

Slow Burst Protection (DS)

get_port_data(chip, MT_MIB_VdslGroup, MT_MIB_VdslChanTable, MT_IDX_RTINTL, MT_MIB_VdslChanCurrTxSlowBurstProtect, &value);

CRC (US)

get_port_data(chip, MT_MIB_VdslGroup, MT_MIB_VdslChanPerfDataTable, MT_IDX_OTINTL, MT_MIB_VdslChanBadBlks, &value);

CRC (DS)

get_port_data(chip, MT_MIB_VdslGroup, MT_MIB_VdslChanPerfDataTable, MT_IDX_RTINTL, MT_MIB_VdslChanBadBlks, &value);

Error Correction (US)

get_port_data(chip, MT_MIB_VdslGroup, MT_MIB_VdslChanPerfDataTable, MT_IDX_OTINTL, MT_MIB_VdslChanFixedOctets, &value);

Error Correction (DS)

get_port_data(chip, MT_MIB_VdslGroup, MT_MIB_VdslChanPerfDataTable, MT_IDX_RTINTL, MT_MIB_VdslChanFixedOctets, &value);

Error Second (US)

get_port_data(chip, MT_MIB_VdslGroup, MT_MIB_VdslPerfDataTable, MT_IDX_OT, MT_MIB_VdslPerfDataESs, &value);

Error Second (DS)

get_port_data(chip, MT_MIB_VdslGroup, MT_MIB_VdslPerfDataTable, MT_IDX_RT, MT_MIB_VdslPerfDataESs, &value);

Throughput (US) High Word

get_port_data(chip, MT_MIB_VdslMetanoiaGroup, MT_MIB_VdslMetanoiaTable, MT_IDX_NEAR, MT_MIB_vdslMetanoiaRxByteCntrHi, &value);

Throughput (US) Low Word

get_port_data(chip, MT_MIB_VdslMetanoiaGroup, MT_MIB_VdslMetanoiaTable, MT_IDX_NEAR, MT_MIB_vdslMetanoiaRxByteCntrLo, &value);

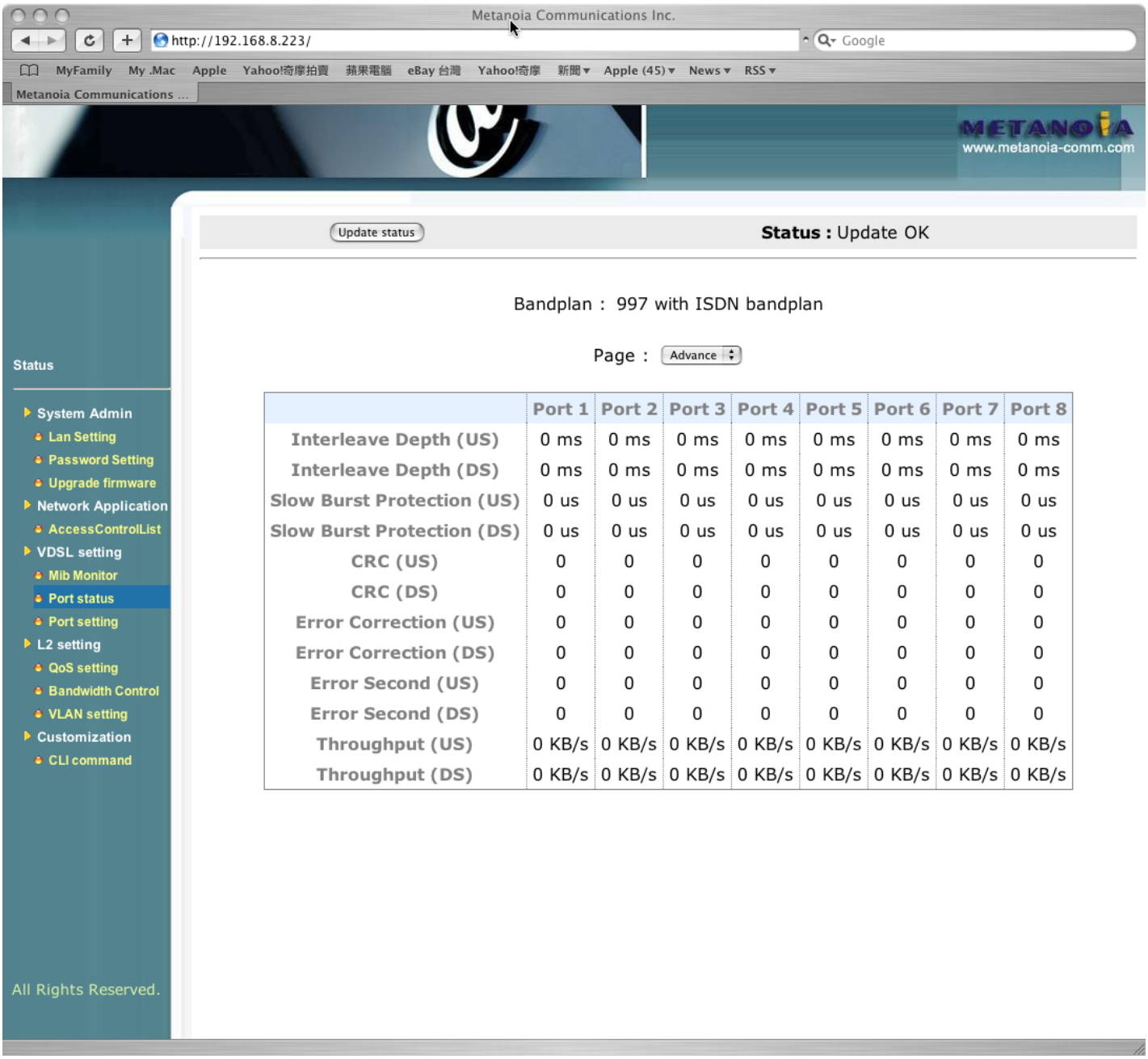
Throughput (DS) High Word

get_port_data(chip, MT_MIB_VdslMetanoiaGroup, MT_MIB_VdslMetanoiaTable, MT_IDX_NEAR, MT_MIB_vdslMetanoiaTxByteCntrHi, &value);

Throughput (DS) Low Word

get_port_data(chip, MT_MIB_VdslMetanoiaGroup, MT_MIB_VdslMetanoiaTable, MT_IDX_NEAR, MT_MIB_vdslMetanoiaTxByteCntrLo, &value);

VDSL status advance



	Port 1	Port 2	Port 3	Port 4	Port 5	Port 6	Port 7	Port 8
Interleave Depth (US)	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms
Interleave Depth (DS)	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms
Slow Burst Protection (US)	0 us	0 us	0 us	0 us	0 us	0 us	0 us	0 us
Slow Burst Protection (DS)	0 us	0 us	0 us	0 us	0 us	0 us	0 us	0 us
CRC (US)	0	0	0	0	0	0	0	0
CRC (DS)	0	0	0	0	0	0	0	0
Error Correction (US)	0	0	0	0	0	0	0	0
Error Correction (DS)	0	0	0	0	0	0	0	0
Error Second (US)	0	0	0	0	0	0	0	0
Error Second (DS)	0	0	0	0	0	0	0	0
Throughput (US)	0 KB/s	0 KB/s	0 KB/s	0 KB/s	0 KB/s	0 KB/s	0 KB/s	0 KB/s
Throughput (DS)	0 KB/s	0 KB/s	0 KB/s	0 KB/s	0 KB/s	0 KB/s	0 KB/s	0 KB/s

Firmware Upgrade

Modify original upgrade features to upgrade VDSL firmware and setting.

To make sure whether the values we've set did applied into the VDSL chip, we can simply replace the "set_port_data" with "get_port_data", then you'll get the current value of the variable we've set by "set_port_data".

For example:

We can set target down stream SNR by

set_port_data(1, MT_MIB_VdslGroup, MT_MIB_VdslLineConfProfileTable, MT_IDX_OT, MT_MIB_VdslLineConfDownTargetSnrMgn, value);

and get current target down stream SNR by

get_port_data(1, MT_MIB_VdslGroup, MT_MIB_VdslLineConfProfileTable, MT_IDX_OT, MT_MIB_VdslLineConfDownTargetSnrMgn, &value);

Please notice the difference in red color.

Get Firmware Version

The following shows how to get VDSL firmware version by get_port_data

```
get_port_data(0, MT_MIB_VdslMetanoiaGroup, MT_MIB_VdslMetanoiaTable, MT_IDX_NEAR, MT_MIB_VdslFirmwareVer, &value1);
printf("FimrwareVersion = %lx", value1);

get_port_data(0, MT_MIB_VdslMetanoiaGroup, MT_MIB_VdslMetanoiaTable, MT_IDX_NEAR, MT_MIB_VdslFirmwareDate, &value1);
printf("FimrwareDate = %lx", value1);

get_port_data(0, MT_MIB_VdslMetanoiaGroup, MT_MIB_VdslMetanoiaTable, MT_IDX_NEAR, MT_MIB_VdslFirmwareTime, &value1);
printf("FimrwareTime = %lx", value1);

get_port_data(0, MT_MIB_VdslMetanoiaGroup, MT_MIB_VdslMetanoiaTable, MT_IDX_NEAR, MT_MIB_VdslTerminal, &value1);
printf("Terminal = %lx\nBit0(1-RT, 0-OT)", value1);
```

2.3.2 Verification phrase

Verify Step by Step

2.3.2.1 HPI One byte write/read timing :

32 bit CPU:
In loader, modify memory map register to make bus write work.
Reference HPI timing to modify parameter of bus signal and timing
If you got a timing that into the HPI timing spec, give the image of scope to metanoia to confirm.

Micro-Controller:
Porting driver to write your own code to produce HPI timing and give the image of scope to metanoia

To trigger corresponding chip select and reset signal in multi-port board with address accessing mode, we can do as followed:

Reset:

```
*(falcon_port+((0x10+chipid)<<16)) = 0x0;
for(i=0;i<100000;i++);
*(falcon_port+((0x20+chipid)<<16)) = 0x0;
```

Read:

```
read_data = *(falcon_port+(chipid<<16));
```

Write:

```
*(falcon_port+(chipid<<16)) = write_data;
```

falcon_port is the mapped CPLD I/O address
chipid start from 0, increase by 1
read_data and write_data are unsigned char variables.

No matter what platform you have, after you can produce HPI one byte write/read, give the board to metanoia to debug interface issue.

2.3.2.2 Porting driver

Porting driver to have download features, write MIB, read MIB, disconnect/connect features

2.3.2.3 Debug driver

Work with metanoia to debug driver

2.3.2.4 Develop user interface

Customer can develop user interface on CLI or web by their own.