Micro-Research Finland Oy
Välitalontie 83 C, FI-00660 Helsinki, Finland

Document: GUNTX-TREF-003
Date: 29 September 2005
Issue: 1

Author: Jukka Pietarinen

E-Gun Transmitter GUN-TX

Technical Reference

Contents

Introduction....2
E-Gun Transmitter Block Diagrams....2
Channel Transmitter....2
RF Clock Source....2
FPGA Implementation....3
Connections....3
Trigger Input Specifications....4
RF Clock Input Specifications....4
Programming Details....5
CR/CSR Support....5
E-Gun Transmitter Function 0 Registers....5
Register Map....6
devMrfGUNTX Driver Functions....9
GunTxSetDelay....9
GunTxGetDelay....9
GunTxSetWidth....9
GunTxGetWidth....9
GunTxFPEnable....10
GunTxFPGetState....10
GunTxSwEnable....10
GunTxSwGetState....10
GunTxSwTrigger....11
GunTxSetFTDelay....11
GunTxShowTemp....11
Network Interface....11
Changing the IP Address of the Module....12
Linux....12
Windows....12
Using Telnet to Configure Module....12
Boot Configuration (command b)....12
Upgrading FPGA Configuration File....13
Linux....13
Windows....13
Linux....13
Windows....13
Temperature Control....14

Introduction

The E-Gun Transmitter (GUNTX) accepts two trigger signals from its front panel, delays the trigger with a resolution of an externally provided reference clock, typically 500 MHz i.e. 2 ns steps and generates modulated optical signals which are sent to the E-Gun Receiver. The trigger may be delayed by up to 2^{32}-1 RF clock cycles. The pulse width is programmable to up to 2^{32}-1 RF clock cycles. In addition to the delay in RF clock steps, the E-Gun Transmitter utilises a fine precision programmable delay which allows adjusting the triggering position with a resolution of approximately 10 ps steps.

The E-Gun Transmitter is provided as a 6U VME64x module.

E-Gun Transmitter Block Diagrams

Channel Transmitter

The E-Gun Transmitter has two identical transmit channels. The channel may be triggered either from the front panel differential LVPECL input or from software register access. Both triggers have dedicated enables which should not be enabled simultaneously. The front panel input, although differential, may be used in single-ended mode, too. A block diagram of one transmit channel is represented in Figure 1.

The front panel input has to be synchronous to the RF clock. A trigger is generated on a rising edge of the input trigger signal. When the channel is triggered two counters start counting down to zero to generate the requested pulse delay and width. Note: due to modulation technique used, the minimum pulse width is limited to four (4) RF clock cycles i.e. 8 ns @ 500 MHz.

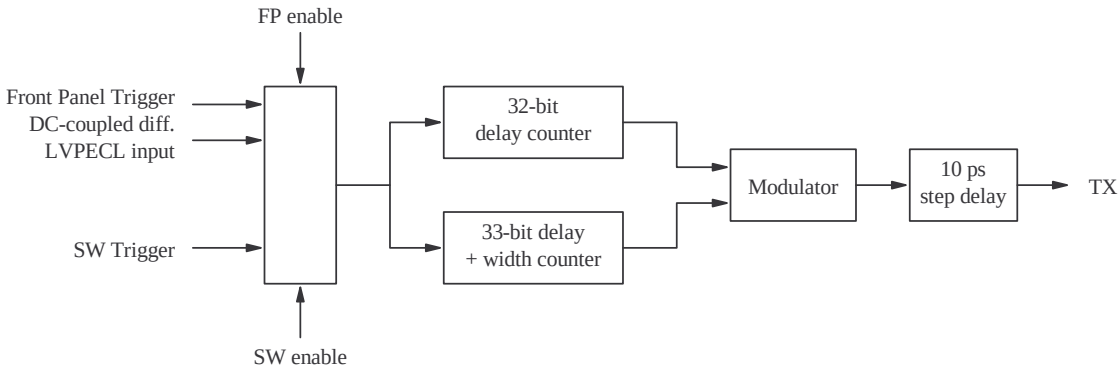

Figure 1: Channel Transmitter (one channel)

The modulator combines the counter output signals with the RF clock to generate the modulated signal. Before transmission to the E-Gun Receiver the signal is delayed by a programmable delay line which is capable of delaying the signal by approximately 0-10 ns in 10 ps steps.

RF Clock Source

By default the E-Gun Transmitter uses an external RF clock source which may be provided either as a single-ended or differential LVPECL signal in the front panel. The clock input is AC-coupled. Optionally there are a places for two 7x5 LVPECL oscillators on the PCB.

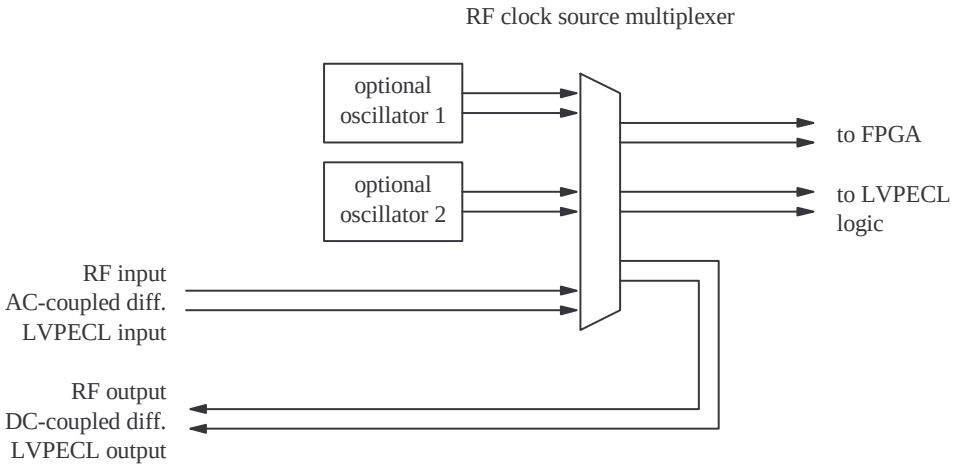

Figure 2: RF Clock Source

FPGA Implementation

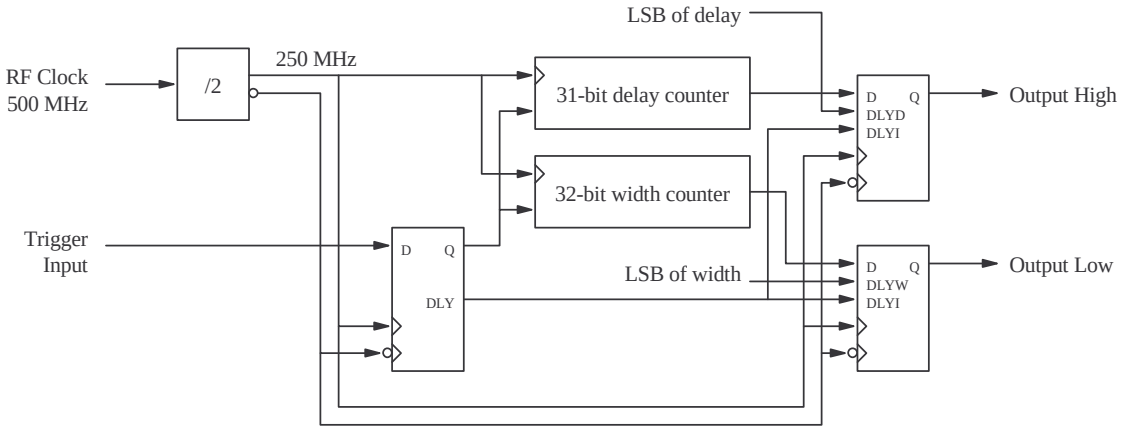

Figure 3: FPGA Implementation of Delay and Width Counters

Connections

The front panel of the E-Gun Transmitter is shown in Figure 4.

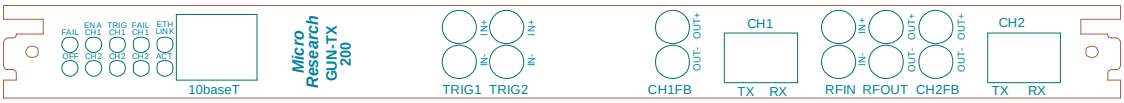

Figure 4: E-Gun Transmitter (GUNTX) Front Panel

The front panel of the E-Gun Transmitter includes the following connections and status leds:

Connector / Led	Style	Level	Description
FAIL	Red Led		Module Failure
OFF	Blue Led		Module Powered Down
ENA CH1	Green Led		CH1 enabled
ENA CH2	Green Led		CH2 enabled

TRIG CH1	Yellow Led		Trigger on CH1
TRIG CH2	Yellow Led		Trigger on CH2
FAIL CH1	Red Led		CH1 Failure (no RF signal)
FAIL CH2	Red Led		CH2 Failure (no RF signal)
ETH LINK	Green Led		10baseT Activity Led
ACT	Yellow Led		Ubicom IP2022 Active (Flashing)
10baseT	RJ45	10baseT	10baseT Ethernet Connection
TRIG1	LEMO EPY	diff. LVPECL	CH1 trigger input (DC-coupled)
TRIG2	LEMO EPY	diff. LVPECL	CH2 trigger input (DC-coupled)
CH1FB	LEMO EPY	diff. LVPECL	CH1 monitor output (decoded from feedback from GUNRX)
CH1TX	LC	optical	CH1 optical output to GUNRX
CH1RX	LC	optical	CH1 optical input feedback from GUNRX
RFIN	LEMO EPY	diff. LVPECL	RF clock input (AC-coupled)
RFOUT	LEMO EPY	diff. LVPECL	RF clock output (AC-coupled)
CH2FB	LEMO EPY	diff. LVPECL	CH2 monitor output (decoded from feedback from GUNRX)
CH2TX	LC	optical	CH2 optical output to GUNRX
CH2RX	LC	optical	CH2 optical input feedback from GUNRX

Trigger Input Specifications

Symbol	Characteristics	Min	Max
V_{IH}	Input High Voltage (Single-Ended)	2075 mV	2420 mV
V_{IL}	Input Low Voltage (Single-Ended)	1355 mV	1675 mV
V_{IHCMR}	Input HIGH Voltage Common Mode Range (Differential)	2.0 V	3.3 V

RF Clock Input Specifications

Symbol	Parameter	Min	Max
V_{DIFF_IN}	Differential Input Voltage Swing (IN-to-/IN)	200 mV	3.3 V

Programming Details

CR/CSR Support

The E-Gun Transmitter module provides CR/CSR Support as specified in the VME64x specification. The CR/CSR Base Address Register is determined after reset by the inverted state of VME64x P1 connector signal pins GA4*-GA0*. In case the parity signal GAP* does not match the GAx* pins the CR/CSR Base Address Register is loaded with the value 0xf8 which corresponds to slot number 31.

After power up or reset the board responds only to CR/CSR accesses with its geographical address. Prior to accessing E-Gun Transmitter functions the board has to be configured by accessing the boards CSR space.

The Configuration ROM (CR) contains information about manufacturer, board ID etc. to identify boards plugged in different VME slots. The following table lists the required field to locate an E-Gun Transmitter module.

CR address	Register	GUNTX
0x27, 0x2B, 0x2F	Manufacturer's ID (IEEE OUI)	0x000EB2
0x33, 0x37, 0x3B, 0x3F	Board ID	0x475458C8

For convenience functions are provided to locate VME64x capable boards in the VME crate.

```
STATUS vmeCRFindBoard(int slot, UINT32 ieee_oui, UINT32 board_id,
                      int *p_slot);
```

To locate the first E-Gun Transmitter in the crate starting from slot 1, the function has to be called following:

```
int slot = 1;
int slot_guntx;
vmeCRFindBoard(slot, 0x000EB2, 0x475458C8, &slot_guntx);
```

If this function returns `OK`, an E-Gun Transmitter board was found in slot `slot_guntx`.

E-Gun Transmitter Function 0 Registers

The E-Gun Transmitter specific register are accessed via Function 0 as specified in the VME64x specification. To enable Function 0, the address decoder compare register for Function 0 in CSR space has to be programmed. For convenience a function to perform this is provided, too:

```
STATUS vmeCSRWriteADER(int slot, int func, UINT32 ader);
```

To configure Function 0 of an E-Gun Transmitter board in slot 3 to respond to A16 accesses at the address range 0x1800-0x1FFF the function has to be called with following values:

```
vmeCSRWriteADER(3, 0, 0x18A4);
```

ADER contents are composed of the address mask and address modifier, the above is the same as:

```
vmeCSRWriteADER(3, 0, (slot << 11) | (VME_AM_SUP_SHORT_IO << 2));
```

To get the memory mapped pointer to the configured Function 0 registers on the E-Gun Transmitter board the following VxWorks function has to be called:

```
MrfGUNTXStruct *pGunTx;
sysBusToLocalAdrs(VME_AM_SUP_SHORT_IO, (char *) (slot << 11),
                  (void *) pGunTx);
```

Register Map

Address Offset	Register	Type	Description
0x000	DelayCh1	UINT32	CH1 pulse delay in RF steps 0 to 2^{32}-1
0x004	WidthCh1	UINT32	CH1 pulse width in RF steps 4 to 2^{32}-1
0x008	DelayCh2	UINT32	CH2 pulse delay in RF steps 0 to 2^{32}-1
0x00C	WidthCh2	UINT32	CH2 pulse width in RF steps 4 to 2^{32}-1
0x010	DelayMeas1	UINT32	CH1 measure trigger delay in RF steps 0 to 2^{32}-1
0x014	Reserved	UINT32	
0x018	DelayMeas2	UINT32	CH2 measure trigger delay in RF steps 0 to 2^{32}-1
0x01C	Reserved	UINT32	
0x020	Reserved	UINT32	
0x024	Reserved	UINT32	
0x028	Reserved	UINT32	
0x02C	Reserved	UINT32	
0x030	Reserved	UINT32	
0x034	Reserved	UINT32	
0x038	Control	UINT32	Control Register
0x03C	Config	UINT32	Configuration Register
0x040	SPSDelayCh2	UINT16	CH2 Sub ps Tune Delay 0 – 4095
0x042	SPSMeasCh2	UINT16	CH2 Sub ps Measurement Tune Delay 0 – 4095
0x044	Reserved	UINT16	
0x046	Reserved	UINT16	
0x048	Reserved	UINT16	
0x04A	SPSDelayCh1	UINT16	CH1 Sub ps Tune Delay (sub ps) 0 – 4095
0x04C	SPSMeasCh1	UINT16	CH1 Sub ps Measurement Tune Delay (sub ps)

			0 – 4095
0x04E	Reserved	UINT16	
0x050	Reserved	UINT16	
0x052	HeatMeas2	UINT16	Heating resistor control for measure delay 2
0x054	Reserved	UINT16	
0x056	Reserved	UINT16	
0x058	Reserved	UINT16	
0x05A	Heat1	UINT16	Heating resistor control for TX delay CH1
0x05C	HeatMeas1	UINT16	Heating resistor control for measure delay 1
0x05E	Heat2	UINT16	Heating resistor control for TX delay CH2
0x060	TempB	UINT16	Temperature sensor B
0x062	TempMeas1	UINT16	Temperature sensor measure delay 1
0x064	Temp1	UINT16	Temperature sensor TX delay CH1
0x066	Reserved	UINT16	
0x068	Temp2	UINT16	Temperature sensor TX delay CH2
0x06A	Reserved	UINT16	
0x06C	TempMeas2	UINT16	Temperature sensor measure delay 2
0x06E	TempA	UINT16	Temperature sensor A
0x070	Reserved	UINT16	
0x072	Reserved	UINT16	
0x074	FineDelayCh1	UINT16	CH1 Fine Delay (~10 ps steps) 0 – 1023
0x076	FineMeas1	UINT16	CH1 Measurement Fine Delay (~10 ps steps) 0 – 1023
0x078	Reserved	UINT16	
0x07A	FineDelayCh2	UINT16	CH2 Fine Delay (~10 ps steps) 0 – 1023
0x07C	FineMeas2	UINT16	CH2 Measurement Fine Delay (~10 ps steps) 0 – 1023
0x07E	Reserved	UINT16	

Control Register

address	bit 31	bit 30	bit 29	bit 28	bit 27	bit 26	bit 25	bit 24
0x038								

address	bit 23	bit 22	bit 21	bit 20	bit 19	bit 18	bit 17	bit 16
0x039			MEAS2 SEL1	MEAS2 SEL0	MEAS1 SEL1	MEAS1 SEL0	CLK SEL1	CLK SEL0

address	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
0x03A								

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x03B								

Bit	Function
MEAS2SEL[1:0]	Channel 2 Measure Source Select 00 – Transmitted signal TX (default after reset) 10 – Reference clock 11 – Received signal (looped back from GUN-RC)
MEAS1SEL[1:0]	Channel 1 Measure Source Select 00 – Transmitted signal TX (default after reset) 10 – Reference clock 11 – Received signal (looped back from GUN-RC)
CLKSEL[1:0]	RF Clock Source Select 00 – Front Panel input (default after reset) 01 – (reserved) 10 – (reserved) 11 – (reserved)

Config Register

address	bit 31	bit 30	bit 29	bit 28	bit 27	bit 26	bit 25	bit 24
0x03C	CH2 MMDE1	CH2 MMDE0	CH1 MMDE1	CH1 MMDE0			CH2 MTRIG	CH1 MTRIG

address	bit 23	bit 22	bit 21	bit 20	bit 19	bit 18	bit 17	bit 16
0x03D	CH2 SWTRIG	CH1 SWTRIG			CH2 MENA	CH1 MENA	CH2 SWENA	CH1 SWENA

address	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
0x03E			CH2 MEAS	CH1 MEAS				

address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0x03F							CH2 FPENA	CH1 FPENA

Bit	Function
CH2MMDE[1:0]	Channel 2 Measurement Trigger Mode 00 – Off 01 – Software trigger 10 – Trigger from CH2 pulse
CH1MMDE[1:0]	Channel 1 Measurement Trigger Mode 00 – Off 01 – Software trigger 10 – Trigger from CH1 pulse
CH2MTRIG	Channel 2 Software Measurement Trigger write ‘1’ to trigger
CH1MTRIG	Channel 1 Software Measurement Trigger write ‘1’ to trigger
CH2SWTRIG	Channel 2 Software Trigger write ‘1’ to trigger Channel 2 by software
CH1SWTRIG	Channel 1 Software Trigger write ‘1’ to trigger Channel 1 by software
CH2MENA	Channel 2 Measurement Enable

CH1MENA	Channel 1 Measurement Enable
CH2SWENA	Channel 2 Software Trigger Enable
CH1SWENA	Channel 1 Software Trigger Enable
CH2MEAS	Channel 2 Measurement Flip-Flop State
CH1MEAS	Channel 1 Measurement Flip-Flop State
CH2FPENA	Channel 2 Front Panel Trigger Input Enable
CH1FPENA	Channel 1 Front Panel Trigger Input Enable

devMrfGUNTX Driver Functions

GunTxSetDelay

Set pulse delay in RF steps for channel

```
STATUS GunTxSetDelay(MrfGUNTXStruct *pGunTx, int channel,
                     UINT32 delay);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
channel	1, 2	Channel to set delay
delay	0 to 0xFFFFFFFF	Delay setting in RF steps
return value		OK on success

GunTxGetDelay

Retrieve pulse delay setting for channel

```
STATUS GunTxGetDelay(MrfGUNTXStruct *pGunTx, int channel,
                     UINT32 *delay);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
channel	1, 2	Channel to get delay value from
delay		Pointer to return delay value to
return value		OK on success

GunTxSetWidth

Set pulse width in RF steps for channel

```
STATUS GunTxSetWidth(MrfGUNTXStruct *pGunTx, int channel,
                     UINT32 width);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
channel	1, 2	Channel to set pulse width
width	0 to 0xFFFFFFFF	Pulse width setting in RF steps
return value		OK on success

GunTxGetWidth

Retrieve pulse width setting for channel

```
STATUS GunTxGetWidth(MrfGUNTXStruct *pGunTx, int channel,
                     UINT32 *width);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
channel	1, 2	Channel to get pulse width value from
width		Pointer to return pulse width value to
return value		OK on success

GunTxFPEnable

Enable Front Panel trigger input for channel

```
STATUS GunTxFPEnable(MrfGUNTXStruct *pGunTx, int channel, int state);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
channel	1, 2	Channel to enable/disable trigger
state	0, 1	0 – disable trigger input 1 – enable trigger input
return value		OK on success

GunTxFPGetState

Retrieve Front Panel trigger state for channel

```
int GunTxFPGetState(MrfGUNTXStruct *pGunTx, int channel);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
channel	1, 2	Channel to get trigger state from
return value		0 – trigger input disabled 1 – trigger input enabled ERROR – could not get state

GunTxSwEnable

Enable Software trigger for channel

```
STATUS GunTxSwEnable(MrfGUNTXStruct *pGunTx, int channel, int state);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
channel	1, 2	Channel to enable/disable software trigger
state	0, 1	0 – disable software trigger 1 – enable software trigger
return value		OK on success

GunTxSwGetState

Retrieve Software trigger state for channel

```
int GunTxSwGetState(MrfGUNTXStruct *pGunTx, int channel);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
channel	1, 2	Channel to get trigger status from
return value		0 – software trigger disabled 1 – software trigger enabled ERROR – could not get state

GunTxSwTrigger

Software trigger channel

```
STATUS GunTxSwTrigger(MrfGUNTXStruct *pGunTx, int channel);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
channel	1, 2	Channel to trigger
return value		OK on success

GunTxSetFTDelay

Set fine delay in approximately 10 ps steps for channel

```
STATUS GunTxSetFTDelay(MrfGUNTXStruct *pGunTx, int channel,
                       UINT32 ftdelay);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
channel	1, 2	Channel to trigger
ftdelay	0 to 0x03ff	Delay in approx. 10 ps steps, 0 to 10 ns delay
return value		OK on success

GunTxShowTemp

Print out temperature of all temperature sensors.

```
STATUS GunTxShowTemp(MrfGUNTXStruct *pGunTx);
```

Parameter	Range	Description
pGunTx		Pointer to E-Gun Transmitter Function 0
return value		OK on success

Network Interface

A 10baseT network interface is provided to upgrade the FPGA firmware and set up boot options. It is also possible to control the module over the network interface.

Changing the IP Address of the Module

The IP address of the module may be changed by sending an ICMP echo request packet with the modules MAC (Media Access Control) address and the IP address the module should respond to. This method is called ARP/PING.

Linux

To set the IP address of an E-Gun Transmitter module to 192.168.1.32 on a Linux machine (as root):

```
# /sbin/arp –s 192.168.1.32 00:0E:B2:00:00:03
# ping 192.168.1.32
```

Now the board should respond to the echo request with echo replies.

Windows

To set the IP address of an E-Gun Transmitter module to 192.168.1.32 on a Windows machine (in command prompt):

```
C:\> arp –s 192.168.1.32 00-0E-B2-00-00-03

C:\> ping 192.168.1.32
```

Now the board should respond to the echo request with echo replies.

Using Telnet to Configure Module

To connect to the configuration utility of the module issue the following command:

```
telnet 192.168.1.32 23
```

The latter parameter is the telnet port number and is required in Linux to prevent negotiation of telnet parameters which the telnet server of the module is not capable of.

The telnet server responds to the following commands:

Command	Description
b	Show/change boot parameters, IP address etc.
h / ?	Show Help
m <address> [<data>]	Read/Write FPGA CR/CSR, Function 0
r	Reset Board
s	Save boot configuration
u	Update IP2022 software
q	Quit Telnet

Boot Configuration (command b)

Command b displays the current boot configuration parameters of the module. The parameter may be changed by giving a new parameter value. The following parameters are displayed:

Parameter	Description
IP address	IP address of module
Subnet mask	Subnet mask of module

Default GW	Default gateway
V2P IP address	(not used)
FPGA mode	FPGA configuration mode 0 – FPGA is not configured after power up 1 – FPGA configured from internal Flash memory 2 – FPGA is configured from FTP server
FTP server	FTP server IP address where configuration bit file resides
Username	FTP server username
Password	FTP server password
FTP Filename	FTP server configuration file name
Flash Filename	Configuration file name on internal flash
UDP Port	UDP server port for FPGA data access

Note that after changing parameters the parameters have to be saved to internal flash by issuing the Save boot configuration (s) command. The changes are applied only after resetting the module using the reset command or hardware reset/power sequencing.

Upgrading FPGA Configuration File

When the FPGA configuration file resides in internal flash memory a new file system image has to be downloaded to the module. Before upgrading it is recommended to retrieve the current image for backup.

This is done using TFTP protocol:

Linux

In Linux use e.g. interactive tftp:

```
$ tftp 192.168.1.32
tftp> bin
tftp> get / backup.bin
tftp> quit
```

Windows

In Windows command prompt issue the following command:

```
C:\> tftp –i 192.168.1.32 GET / backup.bin
```

After this the new image can be downloaded using TFTP protocol:

Linux

In Linux use e.g. interactive tftp:

```
$ tftp 192.168.1.32
tftp> bin
tftp> put filesystem.bin /
tftp> quit
```

Windows

In Windows command prompt issue the following command:

```
C:\> tftp –i 192.168.1.32 PUT filesystem.bin /
```

Now the FPGA configuration file has been upgraded and the new configuration is loaded after next reset/power sequencing. It is recommended to upload the filesystem image back and compare it against the original file to make sure the filesystem was transferred OK before resetting/power sequencing the board.

Temperature Control