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

Document: 4CHTIM-TREF-002
Date: 03 June 2005
Issue: 1

Author: Jukka Pietarinen

Four-Channel Timer 4CHTIM

Technical Reference

Contents

Introduction

The Four-Channel Timer Board (4CHTIM) provides four independent timer channels with programmable delay and pulse width. The channels may be trigger from dedicated trigger inputs from the front panel or a transition board. Common trigger inputs allow triggering several channels from a single trigger source. The timer channels operate synchronous to an externally applied RF clock (typically 500 MHz). The delay and pulse width for each channel may be adjusted from 0 to 2^{32}-1 RF clock cycles. In addition to this the delay of each channel may be fine tuned with approximately 10 ps resolution.

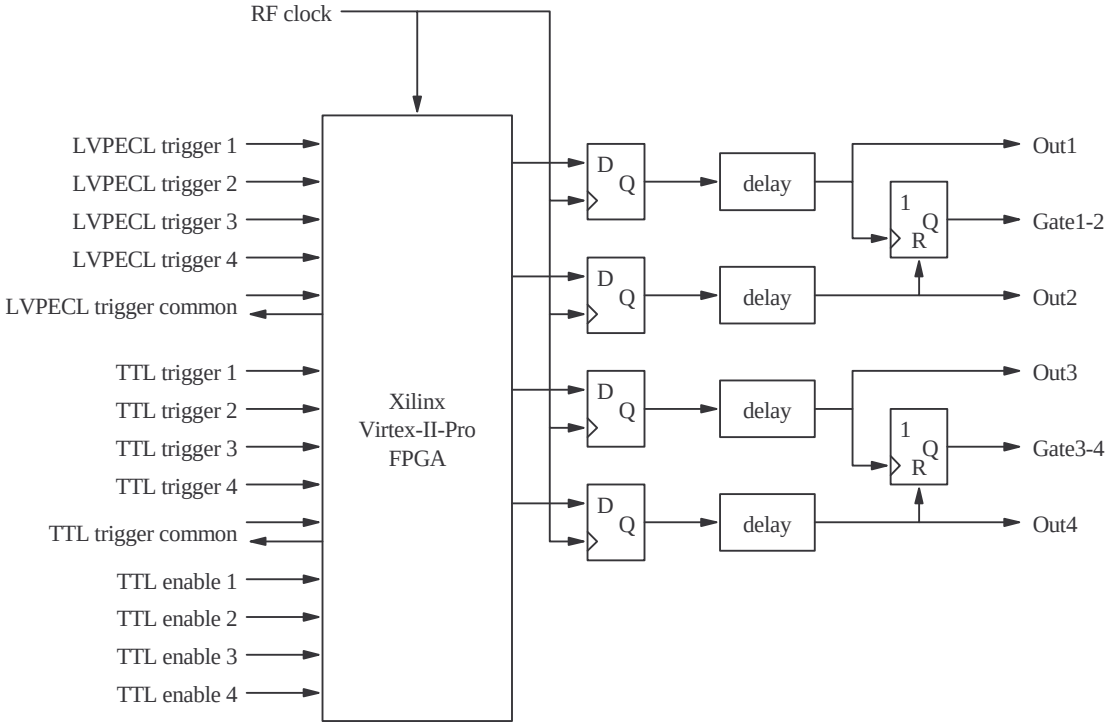

Figure 1: Timer Board

The Four-Channel Timer is provided as a 6U VME64x module.

Four-Channel Timer Block Diagrams

Delay Channel

The Four-Channel Timer board has four identical delay channels. One channel may be triggered from the dedicated front panel differential LVPECL input, transition board trigger input, front panel common trigger input, transition board common trigger input or from software register access. A block diagram of channel triggering is represented in Figure 2. All trigger sourced are combined together and may be enabled simultaneously.

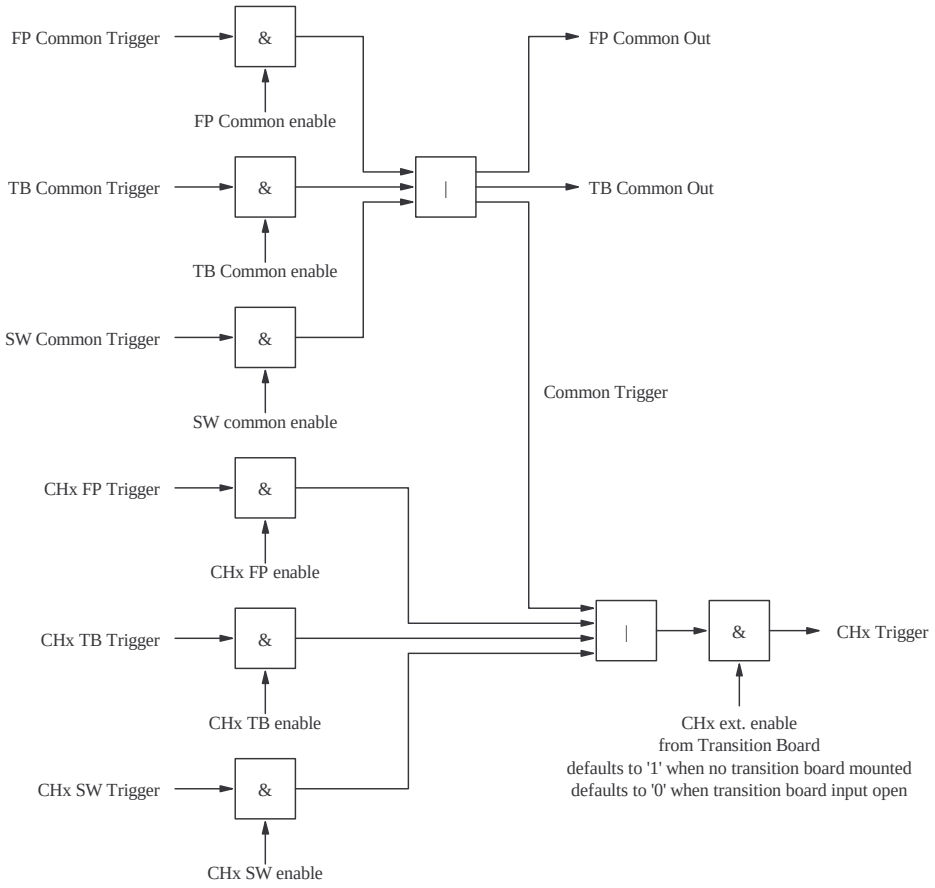

Figure 2: Channel Triggering

The front panel inputs have 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. A block diagram of the delay and width counters is shown in Figure 3.

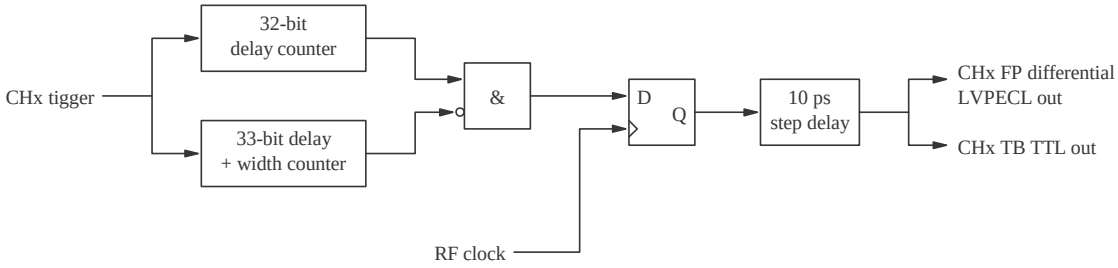

Figure 3: Timer Delay Channel (one channel)

Channel 1 and 2 outputs, and channel 3 and 4 output are combined to generate gated pulses gate-12 and gate-34 as shown in Figure 4.

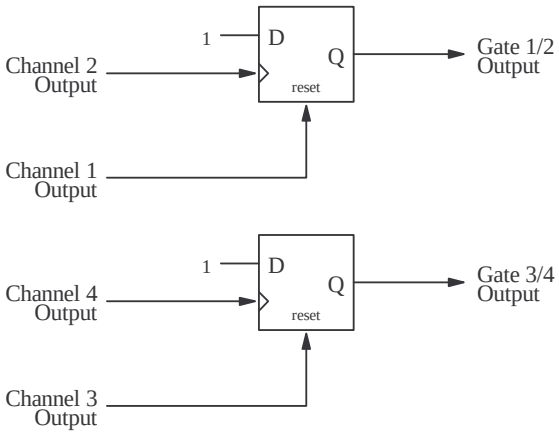

Figure 4: Gate Outputs

RF Clock Source

By default the Four-Channel Timer 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.

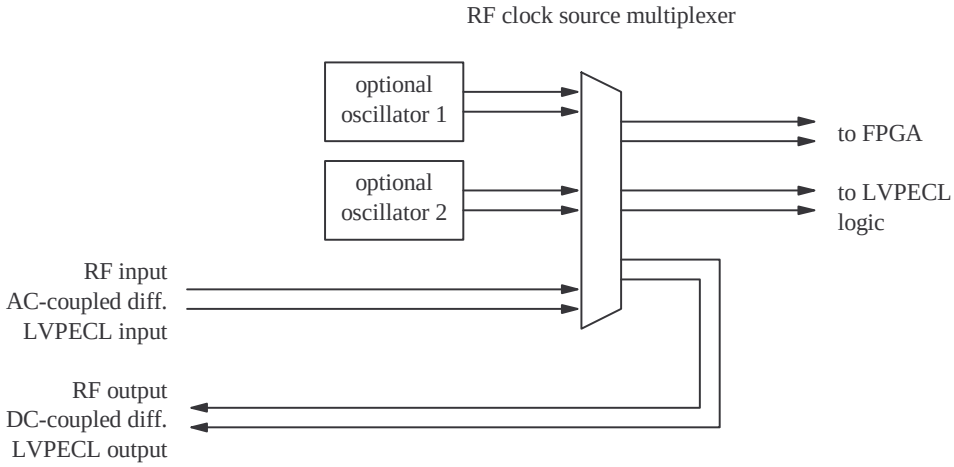

Figure 5: RF Clock Source

FPGA Implementation

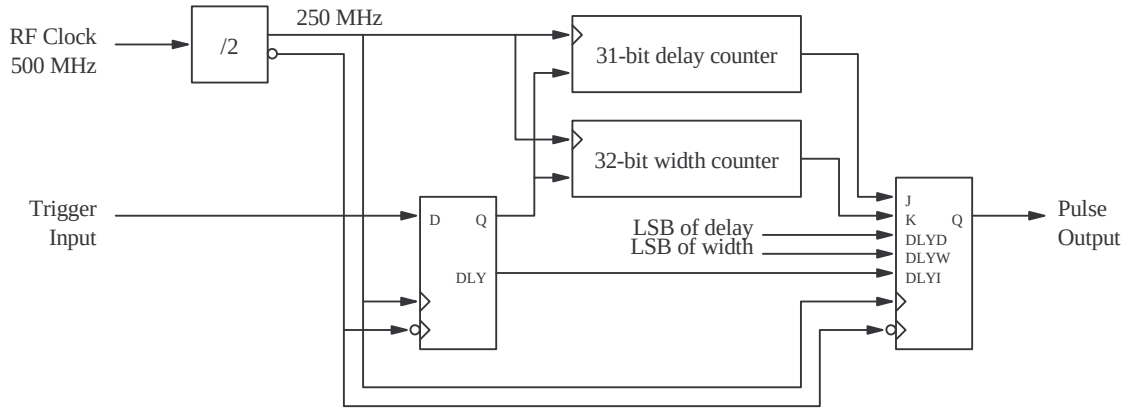

Figure 6: FPGA Implementation of Delay Channel

Connections

The front panel of the Four-Channel Timer is shown in **Error! Reference source not found.**.

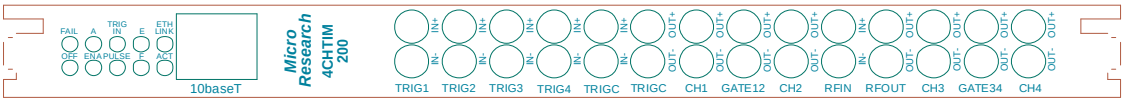

Figure 7: Four-Channel Timer (4CHTIM) Front Panel

The front panel of the Four-Channel Timer includes the following connections and status leds:

Connector / Led	Style	Level	Description
FAIL	Red Led		Module Failure
OFF	Blue Led		Module Powered Down
A	Green Led		
ENA	Green Led		
TRIG IN	Yellow Led		Trigger in
PULSE	Yellow Led		Pulse out on CH1 to CH4
E	Red Led		
F	Red Led		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)
TRIG3	LEMO EPY	diff. LVPECL	CH3 trigger input (DC-coupled)
TRIG4	LEMO EPY	diff. LVPECL	CH4 trigger input (DC-coupled)
TRIGC IN	LEMO EPY	diff. LVPECL	Common trigger input (DC-coupled)
TRIGC OUT	LEMO EPY	diff. LVPECL	Common trigger output (DC-coupled)
CH1	LEMO EPY	diff. LVPECL	CH1 output
GATE12	LEMO EPY	diff. LVPECL	GATE12 output

CH2	LEMO EPY	diff. LVPECL	CH2 output
RFIN	LEMO EPY	diff. LVPECL	RF clock input (AC-coupled)
RFOUT	LEMO EPY	diff. LVPECL	RF clock output (AC-coupled)
CH3	LEMO EPY	diff. LVPECL	CH3 output
GATE34	LEMO EPY	diff. LVPECL	GATE34 output
CH4	LEMO EPY	diff. LVPECL	CH4 output

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

VME P2 User I/O Pin Configuration

The following table lists the connections to the VME P2 User I/O Pins.

Pin	Signal
A1	Transition board ID0
A2	Transition board ID1
A3-A10	Ground
A11	Transition board ID2
A12	Transition board ID3
A13-A15	Ground
A16	Transition board handle switch
A17-A26	Ground
A27-A31	+5V
A32	Power control for transition board
C1	TTL Trigger for Channel 1 (defaults low)
C2	TTL Trigger for Channel 2 (defaults low)
C3	TTL Trigger for Channel 3 (defaults low)
C4	TTL Trigger for Channel 4 (defaults low)
C5	TTL Enable for Channel 1 (defaults high)
C6	TTL Enable for Channel 2 (defaults high)
C7	TTL Enable for Channel 3 (defaults high)
C8	TTL Enable for Channel 4 (defaults high)
C9	(reserved)
C10	(reserved)
C11	(reserved)
C12	(reserved)
C13	(reserved)

C14	(reserved)
C15	(reserved)
C16	(reserved)
C17	(reserved)
C18	(reserved)
C19	(reserved)
C20	(reserved)
C21	(reserved)
C22	(reserved)
C23	(reserved)
C24	(reserved)
C25	TTL Common Trigger Input (defaults low)
C26	TTL Common Trigger Output
C27	TTL Channel 1 Output
C28	TTL Gate-12 Output
C29	TTL Channel 2 Output
C30	TTL Channel 3 Output
C31	TTL Gate-34 Output
C32	TTL Channel 4 Output

Programming Details

CR/CSR Support

The Four-Channel Timer 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 Four-Channel Timer 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 Four-Channel Timer module.

CR address	Register	4CHTIM
0x27, 0x2B, 0x2F	Manufacturer’s ID (IEEE OUI)	0x000EB2
0x33, 0x37, 0x3B, 0x3F	Board ID	0x344354C8

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 Four-Channel Timer in the crate starting from slot 1, the function has to be called following:

```
#include “vme64x_cr.h”
int slot = 1;
int slot_timer;
vmeCRFindBoard(slot, 0x000EB2, 0x344354C8, &slot_timer);
or
vmeCRFindBoard(slot, MRF_IEEE_OUI, MRF_4CHTIM_BID, &slot_timer);
```

If this function returns `OK`, an Four-Channel Timer board was found in slot `slot_timer`.

Four-Channel Timer Function 0 Registers

The Four-Channel Timer 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 a Four-Channel Timer 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 Four-Channel Timer board the following VxWorks function has to be called:

```
Mrf4CHTIMStruct *pTimer;
sysBusToLocalAdrs(VME_AM_SUP_SHORT_IO, (char *) (slot << 11),
                  (void *) pTimer);
```

Register Map

Address Offset	Register	Type	Description
0x000	DelayCh1	UINT32	CH1 pulse delay in RF steps
0x004	WidthCh1	UINT32	CH1 pulse width in RF steps
0x008	DelayCh2	UINT32	CH2 pulse delay in RF steps
0x00C	WidthCh2	UINT32	CH2 pulse width in RF steps
0x010	DelayCh3	UINT32	CH3 pulse delay in RF steps
0x014	WidthCh3	UINT32	CH3 pulse width in RF steps
0x018	DelayCh4	UINT32	CH4 pulse delay in RF steps
0x01C	WidthCh4	UINT32	CH4 pulse width in RF steps
0x020	DelayMeas12	UINT32	CH1/2 measure trigger delay in RF steps

0x024	WidthMeas12	UINT32	CH1/2 measure trigger delay in RF steps
0x028	DelayMeas34	UINT32	CH3/4 measure trigger delay in RF steps
0x02C	WidthMeas34	UINT32	CH3/4 measure trigger delay in RF steps
0x038	Control	UINT32	Control Register
0x03C	Config	UINT32	Configuration Register
0x040	FTDelay3	UINT16	Analog fine tune CH3
0x042	FTMeas34	UINT16	Analog fine tune measure CH3/4
0x044	FTDelay4	UINT16	Analog fine tune CH4
0x046	FREQT2	UINT16	Frequency fine tune for VCXO 2
0x048	FREQT1	UINT16	Frequency fine tune for VCXO 1
0x04A	FTDelay1	UINT16	Analog fine tune CH1
0x04C	FTMeas12	UINT16	Analog fine tune measure CH1/2
0x04E	FTDelay2	UINT16	Analog fine tune CH2
0x050	Heat3	UINT16	Heating resistor control for CH3
0x052	HeatMeas34	UINT16	Heating resistor control for measure CH3/4
0x054	Heat4	UINT16	Heating resistor control for CH4
0x05A	Heat1	UINT16	Heating resistor control for CH1
0x05C	HeatMeas12	UINT16	Heating resistor control for measure CH1/2
0x05E	Heat2	UINT16	Heating resistor control for CH2
0x060	TempB	UINT16	Temperature sensor B
0x062	TempMeas12	UINT16	Temperature sensor measure CH1/2
0x064	Temp1	UINT16	Temperature sensor CH1
0x066	Temp2	UINT16	Temperature sensor CH2
0x068	Temp3	UINT16	Temperature sensor CH3
0x06A	Temp4	UINT16	Temperature sensor CH4
0x06C	TempMeas34	UINT16	Temperature sensor measure CH3/4
0x06E	TempA	UINT16	Temperature sensor A

devMrf4CHTIM Driver Functions

TimerSetDelay

Set pulse delay in RF steps for channel

```
STATUS TimerSetDelay(Mrf4CHTIMStruct *pTimer, int channel,
                     UINT32 delay);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
channel	1 to 4	Channel to set delay
delay	0 to 0xFFFFFFFF	Delay setting in RF steps
return value		OK on success

TimerGetDelay

Retrieve pulse delay setting for channel

```
STATUS TimerGetDelay(Mrf4CHTIMStruct *pTimer, int channel,
                     UINT32 *delay);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
channel	1 to 4	Channel to get delay value from
delay		Pointer to return delay value to
return value		OK on success

TimerSetWidth

Set pulse width in RF steps for channel

```
STATUS TimerSetWidth(Mrf4CHTIMStruct *pTimer, int channel,
                     UINT32 width);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
channel	1 to 4	Channel to set pulse width
width	0 to 0xFFFFFFFF	Pulse width setting in RF steps
return value		OK on success

TimerGetWidth

Retrieve pulse width setting for channel

```
STATUS TimerGetWidth(Mrf4CHTIMStruct *pTimer, int channel,
                     UINT32 *width);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
channel	1 to 4	Channel to get pulse width value from
width		Pointer to return pulse width value to
return value		OK on success

TimerFPEnable

Enable Front Panel trigger input for channel

```
STATUS TimerFPEnable(Mrf4CHTIMStruct *pTimer, int channel, int state);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
channel	1 to 4	Channel to enable/disable trigger
state	0, 1	0 – disable trigger input 1 – enable trigger input
return value		OK on success

TimerFPGetState

Retrieve Front Panel trigger state for channel

```
int TimerFPGetState(Mrf4CHTIMStruct *pTimer, int channel);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
channel	1 to 4	Channel to get trigger state from
return value		0 – trigger input disabled 1 – trigger input enabled ERROR – could not get state

TimerTBEnable

Enable Transition Board trigger input for channel

```
STATUS TimerTBEnable(Mrf4CHTIMStruct *pTimer, int channel, int state);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
channel	1 to 4	Channel to enable/disable trigger
state	0, 1	0 – disable trigger input 1 – enable trigger input
return value		OK on success

TimerTBGetState

Retrieve Transition Board trigger state for channel

```
int TimerTBGetState(Mrf4CHTIMStruct *pTimer, int channel);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
channel	1 to 4	Channel to get trigger state from
return value		0 – trigger input disabled 1 – trigger input enabled ERROR – could not get state

TimerSwEnable

Enable Software trigger for channel

```
STATUS TimerSwEnable(Mrf4CHTIMStruct *pTimer, int channel, int state);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer 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

TimerSwGetState

Retrieve Software trigger state for channel

```
int TimerSwGetState(Mrf4CHTIMStruct *pTimer, int channel);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer 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

TimerFPCommonEnable

Enable Front panel Common trigger

```
STATUS TimerFPCommonEnable(Mrf4CHTIMStruct *pTimer, int state);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
state	0, 1	0 – disable FP common trigger input 1 – enable FP common trigger input
return value		OK on success

TimerTBCommonEnable

Enable Transition Board Common trigger

```
STATUS TimerTBCommonEnable(Mrf4CHTIMStruct *pTimer, int state);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
state	0, 1	0 – disable TB common trigger input 1 – enable TB common trigger input
return value		OK on success

TimerSWCommonEnable

Enable software common trigger

```
STATUS TimerSWCommonEnable(Mrf4CHTIMStruct *pTimer, int state);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
state	0, 1	0 – disable SW common trigger 1 – enable SW common trigger
return value		OK on success

TimerSwTrigger

Software trigger channel

```
STATUS TimerSwTrigger(Mrf4CHTIMStruct *pTimer, int channel);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
channel	1 to 4	Channel to trigger
return value		OK on success

TimerSwCommonTrigger

Software common trigger

```
STATUS TimerSwCommonTrigger(Mrf4CHTIMStruct *pTimer);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
return value		OK on success

TimerSetFTDelay

Set fine delay in approximately 10 ps steps for channel

```
STATUS TimerSetFTDelay(Mrf4CHTIMStruct *pTimer, int channel,
                       UINT32 ftdelay);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer Function 0
channel	1 to 4	Channel to trigger
ftdelay	0 to 0x03ff	Delay in approx. 10 ps steps, 0 to 10 ns delay
return value		OK on success

TimerShowTemp

Print out temperature of all temperature sensors.

```
STATUS TimerShowTemp(Mrf4CHTIMStruct *pTimer);
```

Parameter	Range	Description
pTimer		Pointer to Four-Channel Timer 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 a Four-Channel Timer module to 192.168.1.32 on a Linux machine (as root):

```
# /sbin/arp –s 192.168.1.32 00:0E:B2:00:00:07
# 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 Four-Channel Timer module to 192.168.1.32 on a Windows machine (in command prompt):

```
C:\> arp –s 192.168.1.32 00-0E-B2-00-00-07

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. This is done 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.

Four-Channel Timer Transition Board (4CHTIM-TB)

The Four-Channel Timer Transition Board delivers four TTL level trigger signals, four TTL level channel enable signal and one TTL level common trigger signal to the Four-Channel Timer via the P2 connector. TTL level outputs are provided for each of the four channels and two gate signal. A TTL level common trigger output is provided, too. LEMO connector style EPL.00.250.NTN is used in the transition board front panel. A status led indicates that the board is in operation.

Figure 8: Four-Channel Timer Transition Board (4CHTIM-TB) Front Panel

The front panel of the Four-Channel Timer Transition Board includes the following connections and status leds:

Connector / Led	Style	Level	Description

POWER	Green Led		Transition Board Powered Up
CH1TRIG	LEMO-EPL	TTL 50 ohm	Trigger for Channel 1
CH2TRIG	LEMO-EPL	TTL 50 ohm	Trigger for Channel 2
CH3TRIG	LEMO-EPL	TTL 50 ohm	Trigger for Channel 3
CH4TRIG	LEMO-EPL	TTL 50 ohm	Trigger for Channel 4
CH1ENA	LEMO-EPL	TTL 50 ohm	Enable for Channel 1
CH2ENA	LEMO-EPL	TTL 50 ohm	Enable for Channel 2
CH3ENA	LEMO-EPL	TTL 50 ohm	Enable for Channel 3
CH4ENA	LEMO-EPL	TTL 50 ohm	Enable for Channel 4
TRIGIN	LEMO-EPL	TTL 50 ohm	Common Trigger Input
TRIGOUT	LEMO-EPL	TTL 50 ohm	Common Trigger Output
CH1OUT	LEMO-EPL	TTL 50 ohm	Channel 1 Output
GATE12OUT	LEMO-EPL	TTL 50 ohm	Gate (CH1/CH2) Output
CH2OUT	LEMO-EPL	TTL 50 ohm	Channel 2 Output
CH3OUT	LEMO-EPL	TTL 50 ohm	Channel 3 Output
GATE34OUT	LEMO-EPL	TTL 50 ohm	Gate (CH3/CH4) Output
CH4OUT	LEMO-EPL	TTL 50 ohm	Channel 4 Output