Inhaltsverzeichnis: TIM6U_V2-info

Timing Module

in the Level 1 Global Trigger

6U-Version

H. Bergauer, K. Kastner, M. Padrta, A. Taurok

Aug-05

Version To be updated for version V2

1 Abstract

Preliminary!!!

The TIM module contains the TTCrx chip that receives the common CMS timing and synchronization signals. A programmable TIM chip distributes the central 40 MHz clock, the common synchronization signals and the L1Accept signal to all modules in the GT (= Global Trigger) or DTTF (=Drift Tube Track Finder) crate. The TIM chip can simulate all TTC signals to run the GT crate during tests in stand-alone mode. Optionally for the Global Trigger crate the TIM chip contains memories to monitor input signals and a Readout Processor to append the monitored data bits to the GT events.

2 Design-questions

- There are no net-rules defined yet. The graphic implementation of the rules should take place on sheet 2 of top-of-hierarchy schematics.

 - Ich weiß noch nicht, wie wir die Längenunterschiede von ca 1.5-2cm zw. den verschiedenen Backplane Signalen definieren sollen. (L1A, RESET, BCRES,CLK)

- Länge Clock signale

- FAST SIGNAL are to be defined!!!
- RESET_TIM from VME chip??? ==>yes SYSRES* from VMEbus???? ==>yes
- Serial R between LVCH16245 and TIM chip at external Lemo lines????
- Enable signals from TIM chip to RO_INTERFACE and LVDS_DRIVER changed!! See tim_check\tim_chip.xls

3 Logic description

3.1 Overview

3.2 TTCrx

TTCrx signals:

Clock:	Clock 40, Clock 40Des1, Clock 40Des2;
Channel A:	L1Accept Programmable *Delay in bx*
Channel B:	
Broadcast Data Interface:	Brcst[7:2], EvCntRes, BCntRes; BrcstStr1, BrcstStr2;
Data Interface:	Dout[7:0], SubAddr[7:0], DQ[3:0], DoutStr
Counter Interface:	BCnt[11:0], EvCntHStr, EvCntLStr, BCntStrb

Internal Registers:

BCcntr 12 bits, EvCntr 24 bit...reset by broadcast Reset

L1Acc:

3.3 Timing signals to back-plane and front panel

Programmable Delays

3.4 Reliability checks

Direct Connections to TCS……

3.5 Signal Emulation

3.6 L1A and readout of data

This chapter describes logic only used in the Global trigger crate. The functional simulation has not been done completely until now.

Started by a L1A request the Readout Processor (ROP) extracts data from the Ringbuffer memories. Then the data are sent over a multiplexer to the Channel Link chip to be transferred to the GTFE Readout board. The multiplexer accepts monitoring data on the other port and sends them every 2nd clock cycle also to via the Channel Link chip to the GTFE board. Event and monitoring data are flagged by the identifier word to check the transfer logic.

3.6.1 L1A QUEUE

The L1A queue is used to store incoming L1A signals in a FIFO. For every L1A data of a number of bunch crossings are extracted from the Ring Buffer.
A new L1A writes the first address of the data packet into a FIFO. This address is taken from a bunch crossing counter and the BCRES signal for this counter is delayed to consider the L1A latency.
The extraction logic fetches the first address from the FIFO and loads it into the RingBuffer Read-Address Counter. At the same time a Readout Length counter is updated from the RO_LENGTH register. Then one address after the other is applied to the Ring Buffer Dual Port Memory to extract all data words for the actual L1A. The contents are then stored in the derandomizing buffer. The logic extracts data until the RO-Length counter becomes =0. If there is still another L1A request pending the next start address is read from the FIFO and the procedure is repeated as described above.
A new L1A can write also a control bit into the FIFO. At the same time a ‘Waiting Time’ counter is started that runs until the control bit appears at the FIFO output port.
If the waiting time is longer then the time corresponding to 75% of the Ringbuffer a warning bit is set. If safety margin decreases to 1/16-th of the Ringbuffer then the error bit ‘L1A_TOO_OLD’ is set. A new L1A check can be done only if the previous check procedure has been finished.
An additional warning message will be sent if more then 63 L1As are pending.
In case of calibration event a second control bit is written with the L1A-start address into the FIFO.

3.6.2 RING BUFFER

The Ring Buffer Dual Port memories receive data continuously. A bunch crossing counter provides the write addresses. The reset signal for this address counter is delayed to consider the latency time of the input data. Therefore data of bunch crossing ‘NN’ are written into the address ‘NN’ modulo 1024. After 1024 clock cycles old data are overwritten.
As described above the extraction logic just applies addresses to the Ring buffer memories to extract data.
If enabled the content of the Ring buffer will be ‘frozen’ in case of an error. Also a VME-instruction can freeze the Ring Buffer immediately.

3.6.3 DERANDOMIZING BUFFER

The derandomizing buffer, called RO_BUF (=readout buffer) is implemented as FIFO. It receives the extracted data bits and identifier bits to flag calibration events.
An additional readout buffer receives the corresponding bunch crossing numbers also flagged by 2 bits.

DATA IDENTIFIER bits 17,16	
00	No data/ header words
01	event data
10	calibration event data
11	bunch crossing number

3.6.4 READOUT PROCESSOR

The Readout Processor (ROP) implemented as a state machine runs as long as there data waiting in the derandomizing buffers and as long as the GTFE board is ready to accept event data. The ROP creates event records consisting of an Identifier, Event Number, trigger data, Word Count and EOR. Between records its sends IDLE words via the Channel Link chips to the GTFE board.
First ROP broadcasts a common read instruction to all derandomizing buffers to move data bits of a bunch crossing into registers. Then it resets the Word Counter, sends the IDENTIFIER, the high Event Number bits and the low Event Number bits.
Now it collects one data word after the other from the registers. Then it broadcasts a new read signal to all derandomizing buffers to collect the data bits from the next bunch crossing. This is done until all data bits of an L1A are transferred.
The ROP appends then the Count and the EOR identifier to finish the event record.

3.6.5 Data format

Bits 15- 0: trigger data or BC numbers or identifiers, word count, idle-id
Bits 17-16: Data Identifier bits. *// See table above.*
Bits 19-18: 0 0
~~Bits 26-20: incrementing number~~
~~Bit 27: = 0 EVENT data, =1 Monitoring data~~
Remark: This format has to be changed to get a 28 bit IDLE code. Not done in schematic.
Bits 25-20: incrementing number
Bit 27-26: =00 IDLE; =01 Event; =10 Monitoring; =11 xxx

3.7 Messages from TCS via TTC

3.8 Fast Signals to TCS

The TIM board sends the standard set of Fast Signals to the TCS board like all other GT boards.

ROBUF= Readout Buffer FIFO
RIBUF= RING BUFFER for data 1 kwords
L1A-Queue =FIFO to store new L1A

TIM_ERROR signals:
- BAD_MAX_BC *// BCR comes later than expected by the BC-counter*
- BAD_LOCAL_BC *// The TTC and local BC number do not agree.*
- DBERR *// Double bit error from the TTCrx chip*

TIM_OUT_OF_SYNC signals
- ROBUF_SYNCERR *// ROBUFs become not empty at the same time.*
- ROBUF_OVF *// Readout Buffer FIFO are full and a write access is pending.*
- L1A_TOO_OLD *// The L1A have to wait too long in the L1A queue.* *// Data in the RingBuffer might be overwritten.*
- TOO_MANY_L1A *// More than 63 L1As are waiting in the L1A queue.*

TIM_WARNING_OVFLO
- L1A_OLD_WARN *// More than 75% of the RingBuffer has been overwritten since the requested L1A data*
- WARNING_ROBUF_OVF *// The ROBUF FIFOs are 75% full.*

TIM_READY
- =TIM_SETUPDONE bit=1 and TTC_READY=1 and TIM_BUSY=0 *// The TIM board is ready to run.*

TIM_BUSY
- =TIM_SETUPDONE bit=0

// The setup procedure has not been finished yet.

4 Clocks and interfaces

4.1 Selection of Clock Sources

a) Select clock for PLL circuit

JP37 = 3-2 ➔ CK_TO_PLL= CLOCK40DES1
select TTC clock directly from TTCrx chip ***(default)***

JP37 = 1-2 ➔ CK_TO_PLL= CLK_OSC
select oscillator clock *(for tests only)*

b) Make CLOCK_TTC using original or improved TTCrx clock

JP36 = 1-2 ➔ CLOCK_TTC= CLOCK40DES1
select original TTCrx clock ***(default)***

JP36 = 3-2 ➔ CLOCK_TTC= CLK40PLL
select improved TTC clock from PLL circuit

c) Select CLK_EXT, the external clock for TIM chip:

JP35 = 3-2 ➔ CLK_EXT= CLOCK_TTC
select TTC clock ***(default)***

JP35 = 1-2 ➔ CLK_EXT= CLK_X
select ECL/NIM clock from the Front Panel LEMO

d) Select clock inside TIM chip:

JP10 = 3-2 ➔ SEL_TTCLK = '1' select TTC clock ***(default)***

JP10 = 1-2 ➔ SEL_TTCLK = '0' select oscillator clock CLK_LOCAL

e) Select source for CLK_BACK going to the back-plane

Insert only one of 4 jumpers!

JP3= ON	selects CLK_TIM, clock of TIM chip ***(default)***
JP4= ON	selects CK_OSCB, oscillator clock
JP5= ON	selects CKTTCB, clock from TTCrx or from PLL circuit
JP31= ON	selects CK_XB, external clock from LEMO connector

f) Select source for VME chip:

Insert only one of 3 jumpers!

JP32= ON	selects CK_OSCV, oscillator clock ***(default)***
JP33= ON	selects CKTTCV, clock from TTCrx or from PLL circuit
JP34= ON	selects CK_XV, external clock from LEMO connector

g) Select source for CLK_LEMO that feeds the front panel LEMO connectors CKO1..12

Insert only one of 4 jumpers!

JP30= ON	selects CK_XF, external clock from LEMO connector
JP8= ON	selects CKTTCP, clock from TTCrx or from PLL circuit
JP6= ON	selects CK_OSCP, oscillator clock
JP7= ON	selects CLK_TIM_P, clock of TIM chip ***(default)***

The DLL circuit inside the TIM chip eliminates any delay of the TTCrx clock signal. The other fast signals (L1A, BCRES,...) going to the backplane are adjusted to the clock signal of the TIM chip.
For the data taking run select TTC clock in the TIM chip and send the TIM clock to the backplane, in other words set all switches and jumpers to their default positions. The other jumper and switch positions are used for various tests.

4.2 Front Panel Inputs

Optical TTC fibre from a TTCex board to the TTCrx mezzanine board that delivers the common CMS clock and synchronization signals.
External Clock CLK_X …=default LHC-clock input for GT crate.

Actually AC-coupling allows to apply either ECL or NIM signals for tests.
Later it might be changed to DC coupled negative ECL levels.

External L1A_X **//Input circuit changed to NIM levels for tests**
used to run readout tests to find highest possible L1A rate. Burst tests.
External BCRES_X (Bunch Counter Reset)
//Input circuit changed to NIM levels for tests.
//Later it might be changed to DC coupled negative ECL levels to receive the ORBIT signal from the TTCmi , the LHC –machine interface.
- to run with different orbit lengths,
- to run synchronously with other Trigger electronics (local tests) etc..

4.2.1 TTCrx signals

The signals are generated by a TTCvi board, go to a TTCex board and are sent via an optical fibre to the TTCrx receiver chip that is mounted on a TTCrx Mezzanine board on the TIM module.

Clock:	Clock 40, Clock 40Des1, Clock 40Des2;
Channel A:	L1Accept *with programmable Delay in bx*
Channel B:	
Broadcast Data Interface:	Brcst[7:2], EvCntRes, BCntRes; BrcstStr1, BrcstStr2;
Data Interface:	Dout[7:0], SubAddr[7:0], DQ[3:0], DoutStr
Counter Interface:	BCnt[11:0], EvCntHStr, EvCntLStr, BCntStrb

Internal Registers:
BCcntr 12 bits, EvCntr 24 bit...reset by broadcast Reset

4.3 Front Panel Outputs

MONX : returns the external CLK_X (LEMO input on front panel).
MON : general monitoring LEMO output. The SW6 switch selects the signals sources:

5-1	RESET_PAN	// RESET delayed for the Front Panel; from TIM chip
5-2	L1A_PAN	// L1A delayed for the Front Panel; from TIM chip
5-3	CLK_OSCM	// oscillator
5-4	CKTTCMON	// CLOCK_TTC = original or improved TTC clock

BCRES_TTC // BCRES delayed for the Front Panel; from TIM chip
CKO_1...12 Clock outputs; 40 MHz, 50 Ohm ABT driver (TTL level).
Source selected by jumpers. See 4.1 above. *In the GT-crate the clock outputs are connected to the Fast Signal Conversion boards and the Tracker Emulators (APVE).*
CKO_13...24 An additional set of 12 CLK signals is available if the TIM board is used in the GT crate and implemented with a 9U frontpanel.

4.4 Front Panel Buttons and LEDs

SW3 INACTIVE // If pushed it sets the TIM board into the 'inactive' state.
SW4 RUNNING // If pushed it sets the TIM board into the 'running' state.

DIO7 GREEN:	**L1A**	// as sent to the back-plane
DIO7 RED:	**NTTCRX_ERR**	// shows error in TTCrx chip
DIO6 GREEN:	**RUNNING**	
DIO6 RED:	**INACTIVE**	
DIO5 GREEN:	**TTCREADY**	// TTCrx chip is ok
DIO5 RED:	**VME access** is active	

4.5 Control signals via back-plane

4.5.1 Clock, L1A, BCR, L1Reset distribution up to version V1002

The TIM chip receives the common CMS 40 MHz clock and from the TTCrx chip the L1A (Level 1 Accept) and the messages Bunch Counter Reset (BCR or BCRes) and L1Reset. It sends the 4 signals as differential point-to-point signals (LVDS) via the back-plane to each board in the crate. The signals for each slot can be disabled by software if boards are not plugged in.
The signals L1A, RESET(or RESYNC), EVCNT_RES are encoded according to the table below. The signal BCRES is not encoded and is sent with a different delay.

			TIM signals via backplane	Delays applied
x	0	0	NOP	----
x	0	1	**RESET/RESYNC**	L1A_DLY_H/L
x	1	0	**L1A**	L1A_DLY_H/L
x	1	1	**EVCNT_RES**	L1A_DLY_H/L
1	x	x	**BCRES**	RES_DLY_H/L

4.5.1.1 Version V1003 DESIGN CHANGE to be done (Oct.2003)

L1A could arrive concurrently with BCRES or concurrently with STOP/GO.
The BGO commands are never sent concurrently and can be coded.
Agreement 20. Oct 03 with J. Eroe:
Therefore the encoding shown in table below will be implemented to send 5 BGO commands via 3 signal lines to the DTTF/GT boards.
.

L1A	BCRES	L1RES	Command
0	0	1	L1RES /RESET/RESYNC
0	1	0	BCRES
0	1	1	EVENT CNTR RESET
1	0	0	L1A
1	0	1	GO/STOP *
1	1	0	Concurrent L1A , BCR
1	1	1	ORBIT CNTR RESET

Table 1 Encoded L1A and BGO commands

- The GO/STOP command forces an inactive circuit into the RUN state and a data taking circuit into the STOP state. The L1RES signal forces the circuit always into the stop-state.
- If L1A and GO/STOP appear at same time then GO/STOP will be sent at the next clock tick.

  Remarks:
  The other BGO commands (Test_EN, Private_GAP, Private_Orbit, HardRes) are not used in DTTF. DTTF sends calibration events like any other events during data taking runs. The calibration events are flagged in the data records.
  DTTF and GT ignore Private Gaps and Orbits and also ‘private’ BGO commands.
  It is assumed that no ‘official’ BGO cmds are sent during Private Gaps and Orbits.
  The TIM chip inhibits L1A during STOP periods. Therefore GO and STOP commands are not required by GT, DTTF boards anymore?

4.5.1.2 Remark for GT crate

In the GT crate on each card a PLL clock driver chip regenerates the clock signal and broadcasts it to all chips on the board with a maximum phase difference of less than 1 ns. The PLL circuits are synchronised 40ms after the start of the clock signal. A 40 MHz on board oscillator can be used instead of the TTC clock for stand-alone tests.

4.5.2 Signals between TIM and TCS board (GT crate)

4.5.2.1 8 TIM to TCS signals ('Fast Signals')

The TIM board sends 5 Fast Signals to the TCS board:

(ATTENTION The TIM to TCS signals will be changed!!
We will encode the status bits and send them to the FDL board, to be combined with the status bits from the other GT boards!)

TIM_ERR *composed by an OR of:*

BAD_LOCAL_BC: *Difference between local BC counter and the TTCrx BC-number.*
BAD_MAX_BC: *The BCR signal arrives not at local BC count=3564.*
DBERR: *double bit error from TTCrx chip.*

TIM_OUT_OF_SYNC *composed by an OR of:*

ROBUF_SYNCERR

In addition to the derandomizing Readout Buffer FIFO for extracted data another FIFO containing the corresponding bunch crossing numbers runs in parallel. If the 'EMPTY' flags of both FIFOs disagree then this error flag will be set.

ROBUF_OVF

If the derandomizing Readout Buffer is full the next write access sets this error flag.

L1A_TOO_OLD

A L1A arrives with a constant latency at the L1A queue and waits to extract the corresponding data from the Ring Buffer memory to move them into the derandomizing Readout Buffer (RO-Buffer). If the waiting time of a L1A exceeds the equivalent of 15/16 of the Ring Buffer size then a monitoring circuit sets the error flag L1A_TOO_OLD.

TOO_MANY_L1A

If more than 63 L1A are waiting in the L1A-queue then this error flag appears.

TIM_WARNING_OVFLO:

L1A_OLD_WARN

A L1A arrives with a constant latency at the L1A queue and waits to extract the corresponding data from the Ring Buffer memory to move them into the derandomizing Readout Buffer (RO-Buffer). If the waiting time of a L1A exceeds the equivalent of 75% of the Ring Buffer size then a monitoring circuit sets the warning flag L1A_OLD_WARN.

WARNING_ROBUF_OVF

If 75% of the derandomizing Readout Buffer are occupied then this warning flag will be set.

TIM_READY

The TTCrx chip has to send a TTC-ready flag and the initialization program has to set the command register bit TIM_SETUPDONE =1 to tell the Trigger Control system on the TCS board that the TIM board is ready to run. The TTC-ready flag can be emulated by software when running without the TTC link..

TIM_BUSY

TIM_BUSY = 1 (active) as long as the initialization program has not set the command register bit TIM_SETUPDONE =1.

Other signals to the TCS board:

L1_RESET

To be explained later.

TI_INHIBIT_PHYS_L1A

To be explained later.

TI_INHIBIT_ALL_L1A

To be explained later.

4.5.2.2 8 TCS to TIM signals

L1A_FROM_TCS *will be used to check if the same L1A arrives also via the TTC optical link. Other 7 signals are not defined yet!*

4.5.3 Signals between TIM and FDL board (GT crate)

4.5.3.1 8 TIM to FDL signals

Not defined yet!

We will encode the status bits and send them to the FDL board, to be combined with the status bits from the other GT boards!)

4.5.3.2 8 FDL to TIM signals

Not defined yet!

4.5.4 Signals between TIM and GTFE board (GT crate)

4.5.4.1 1 TIM to GTFE signal

Not defined yet!

4.5.4.2 1 GTFE to TIM signal

GTFE_READY

GTFE_READY =1 allows the Readout Processor in the TIM chip to send event records as long as there are any in the RO-Buffer.

5 Synchronization and monitoring

Achtung dieser Teil muss noch erneuert werden

5.1 BX-Synchronisation inside the GT-crate by BcntRes

The BCRes is sent to the GTF and all PSB boards and starts at the same time the bunch crossing counters of the synchronisation circuits. At the end of the LHC cycle the contents of all Bunch counters on all boards are stored and then checked by a monitoring program. BCRes is sent every LHC cycle and resynchronises without loosing bx-data.

5.1.1 Readout of data: **See description of PSB too.**

In case of a L1Accept the TTCrx sends the Event counter high and low part and the bunch crossing number. An offset is subtracted from the BX number and a Event/Monitor Identifier is added to the Event-number. Then all three words and the 3 strobe signals are stored consecutively in a short FIFO and afterwards they are broadcasted in the same order to the GTF and all PSB modules, but with a frequency of 10 or 20 MHz to avoid time problems on the back-plane. The BcntrStrobe signal starts a Readout processor (ROP) on every board, which collects data from all Dual Port Memories and moves them to the GTFE link. As the first word the ROP sends the event number with the identifier.

5.1.2 Fast Readout of Monitoring data:

The Global Monitoring circuit on the TIM board extracts data from the DPMs of all boards simulating a L1Accept and uses the links of the event readout to collect data on the GTFE board. The Readout processors on the GTF and the PSBs insert a **monitoring identifier** ***into the event number word, which is used to move the monitoring data to the Monitor memory on the GTFE board. In average 2 monitoring request can be sent between two L1Acc. On the GTFE board monitoring data go into a special memory, which is read separately.***

5.2 Orbit Monitoring request and special trigger to read statistics data

There are several counters in the GT crate which should be read every nth bunch crossing: Dead time counters, rate counter etc.
The pretrigger+trigger sequence is generated on the TIM module and starts with a data taking run. A 1/n counter with programmable rate generates this trigger.

5.2.1 Orbit Monitoring request:

There are several counters in the GT crate which should be read every nth bunch crossing. This trigger is generated on the TIM module and starts with a data taking run (calibration, monitoring, private or physics run). The rate is programmable.
A 1/n counter generates this trigger. The NewRun resets and the BcRes signal increments the counter. Every n orbits a EN_ORBIT_RESET saves and then resets all counters in the GT-crate with the next BcRes signal. During the next LHC orbit send a Mon_Trig Request to read all data to the FED module.

5.2.2 XON/XOFF: See GT-Overview too!!!

If the CMS DAQ cannot accept new event anymore it sends a XOFF signal. New L1Acc from the TTC system are inhibited but data for all pending readout requests are collected and transferred to the GTFE modules, where they wait in the DPMs for transfer. The Trigger logic continues to work but new triggers to the TTC system are suppressed.
In case of a fatal DAQ breakdown all events in the readout chain are cancelled.
The number of incoming but suppressed L1Acc's is counted.
BETTER: The CMS-DAQ should give a STOP message to the TTC system. The TTC should send an XOFF message to start dead time counters everywhere. Then it should send an XON to prepare all readout systems for the following L1Acc requests.

NO: ONE Deadtime counter in GT ONLY !!!!!

TTCrx: Normally the content of the TTCrx Bunch counter is present on the BCnt[11:00].
When a L1A arrives the following sequence of data is put onto the BCnt[11:00] bus.
L1Acc sequence:

Control Register(1,0)	**Cycle**	**Sequences**
00	0	Event counter low on bunch counter bus
01	*0*	*Bunch counter on bunch counter bus*
10	***0***	***Event counter low on bunch counter bus***
	1	***Event counter high on bunch counter bus***
11	**0**	**Bunch counter on bunch counter bus**
	1	**Event counter low on bunch counter bus**
	2	**Event counter high on bunch counter bus**

6 Power circuits

6.1 Backplane power

The DTTF crate provides +5V, +3.3V and the GT crate in addition +2.5V and 1.8V.
In the DTTF crate the voltage and GND pins occupy column ‘C’ of the 2 mm connectors.
In the GT crate column ‘C’ of the upper A, B, C 2-mm connectors is reserved for GND pins.
Column ‘C’ of the lower A, B, C 2-mm connectors are still undefined. The voltages in the GT crate occupy pins of the 160-pin VME connector.
The TIM board receives +5V, +3.3V via VME connector pins. Two GND pins, a +5V and a +3.3V pin make contact first to bias the VME signals and to disable output signals to the backplane if the TIM board is plugged into a living crate. See also chapter about Hot Swap circuit.

6.2 Onboard power-supply

There will be a +1,5V power-supply for the VIRTEX-II chip with 3A output-current.
National LP3966, low-drop-out, adj. 1,5V/3A, TO-220 or TO-263.
Input voltage: 3.3V

Other possible components:
Linear regulators:
National LP3965, low-drop-out, adj. 1,5V/1,5A, TO-220 or TO-263
National LM1085, adj. 1,5V/1,5A, TO-220 or TO-263
National LM1086, adj. 1,5V/3A, TO-220 or TO-263
Switching regulators:
Maxim MAX1843, 1,5V/2,7A, QFN-28

7 VME chip

8 Timing chip

8.1 Definition of Left-Right Slots in the 6u prototype GT crate

This table has been copied from B. Neuherz and is probably not controlled by others.

TIM card	BACKPLANE 6U	SLOT NR
L1	GTL2	14
L2	GTL1	13
L3	PSB6	12
L4	PSB5	11
L5	PSB4	10
L6	PSB3	9
L7	GMU1	7
L8	PSB1	5
R1	FDL1	16
R2	FDL2	17
R3	GTFE	18
R4	GTL3	19
R5	PSB7	20
R6	PSB8	21
R7	GMU2	8
R8	PSB2	6

8.2 VME addresses

A31 ......A24	A23 …A20	A19	A18		*Address Modes*
BASE ADDRESS_GT	xxxx	x	x		*Extended Base address*
xxxx xxxx not available	BASE ADDRESS_DTTF				*Standard Base address*

	A17	A16	A15…12	A11...8	A7...4	A3...A1, x
TIM chip	**0**	**1**	**a a a a**	**a a a a**	**a a a a**	**a a a 0**
registers	**0**	**1**	**0x**	**0x**	**00 –5E**	
TTCrxdump	**0**	**1**	**0x**	**0x**	**80 – 9E**	
Free space	**0**	**1**	**0x**	**0x**	**A0 – FE**	

Free space	**0**	**1**	**0x**	**100 – 1FFE**
BC-Table 4k W16	**0**	**1**	**2000 – 3FFE**	
RING BUFFER 1k W16	**0**	**1**	**4000 – 47FE**	
free *nn k W16*	**0**	**1**	**4800 – FFFE**	
Free space	*2 0000 – 3 FFFE*			

8.2.1 TIM chip registers

WARNING: 11.11.02 A.T.:
All register addresses have been changed to apply delays for up to 16+16=32 bx for L1A and L1_RESET/RESYNC and for the BCRES signals.

8.2.1.1 Delay Registers for boards on left and right side

The signals L1A, RESET(or RESYNC), EVCNT_RES are encoded according to table below. The signal BCRES is not encoded and is sent with a different delay.

- L1A = 'Level 1 Accept' trigger signal to read an event.
- RESET = signal to resynchronize the boards (other names L1_RESET or RESYNC).
- EVCNT_RES = resets the event counters after a resynchronization procedure.

Signals on backplane			2 bits are encoded to send EVCNT_RES	Delays applied	Remarks
BCres	L1a	Reset			
x	0	0	NOP	----	
x	0	1	**RESET/RESYNC**	L1A_DLY_H/L	*Use same delay value for 3 signals*
x	1	0	**L1A**	L1A_DLY_H/L	
x	1	1	**EVCNT_RES**	L1A_DLY_H/L	
1	x	x	**BCRES**	RES_DLY_H/L	

The total delay consists of (DLY_L +1) + (DLY_H +1). Each hex-number programs a 15bx delay circuit. The value 'F' means 'no delay'. The minimum delay 'FF' =0 bx and the maximum delay 'EE'=30 bx.
DLY_L1 is the delay for the next board on the left side of the TIM6U module, DLY_L2 for the 2nd on the left side and so on. DLY_R1…R8 define the delays for boards on the right side.

Address A17-A0	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0000	DLY_L1	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 0002	DLY_R1	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 0004	DLY_L2	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 0006	DLY_R2	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 0008	DLY_L3	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 000A	DLY_R3	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 000C	DLY_L4	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 000E	DLY_R4	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 0010	DLY_L5	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 0012	DLY_R5	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 0014	DLY_L6	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			
1 0016	DLY_R6	L1A_DLY_H				L1A_DLY_L				RES_DLY_H				RES_DLY_L			

1 0018	DLY_L7	L1A_DLY_H	L1A_DLY_L	RES_DLY_H	RES_DLY_L
1 001A	DLY_R7	L1A_DLY_H	L1A_DLY_L	RES_DLY_H	RES_DLY_L
1 001C	DLY_L8	L1A_DLY_H	L1A_DLY_L	RES_DLY_H	RES_DLY_L
1 001E	DLY_R8	L1A_DLY_H	L1A_DLY_L	RES_DLY_H	RES_DLY_L
1 0020	DLY_L9	L1A_DLY_H	L1A_DLY_L	RES_DLY_H	RES_DLY_L
1 0024	DLY_TIM*	L1A_DLY_H	L1A_DLY_L	RES_DLY_H	RES_DLY_L
1 0026	DLY_PAN$^+$	L1A_DLY_H	L1A_DLY_L	RES_DLY_H	RES_DLY_L

**) GT-only: DLY_TIM defines the delays for the Readout logic in the TIM chip.*
$^+$) DLY_PAN defines delays for the front panel signals RESET_PAN, BCRES_PAN and L1A_PAN.

8.2.1.2 DISABLE Boards in Crate

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0022	DIS_ BOARDS	L 8	R 8	L 7	R 7	L 6	R 6	L 5	R 5	L 4	R 4	L 3	R 3	L 2	R 2	L 1	R 1
	default values	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*

EXAMPLE: BIT D14=1 disables board L(eft) 7
***DIS_BOARD_L9:** See COMMAND register bit11 in 8.2.1.10.*

8.2.1.3 CRATE delays

If the ORBIT ECL signal is used as the bunch counter reset signal BCRES then the DLY_CRATE_ECL is used to adjust the DTTF respectively the GT crate to the LHC orbit.
If BCRES from the TTCrx chip is used then the adjustment is done either by programming the TTCrx chip or by programming the DLY_CRATE_TTC register.
Total delay = 1 + delay[15:0].
Warning: The circuit uses a 16-bit counter and therefore the delay value has to be set smaller then 3564. Otherwise the previous BCRES will be suppressed.

Address A17-A1	Registername	D15 ……………D0
1 0028	DLY_CRATE_TTC	*value < 3564; default =0*
1 002A	DLY_CRATE_ECL	*value < 3564*

8.2.1.4 UNUSED Registers

The registers are free. Old version contained CHIP_ID H/L, that is now at 1 0060.

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 002C	xxxxxxxxx	*free*															
1 002E	xxxxxxxxx	*free*															

8.2.1.5 SIMULATION PERIODS

The registers define the time (unit=1 LHC orbit) between active orbits. During an active orbit simulated Trigger signals or BGO commands or UserMessages are sent according to the values in the BC-Table. If xx_PERIOD=0 then the messages are sent every orbit. See also chapter PERIODIC SIMULATION

Address A17-A1	Registername	D15 ………… D0

1 0030	TRIG_PERIOD	*Period for L1A and MonRqst signals*
1 0032	BGO_PERIOD	*Period for Bgo signals and UserMessages*

8.2.1.6 OBIT LENGTH

Address A17-A1	Registername	D15-D12	D11 – D8	D7 – D4	D3 – D0
1 0034	ORBIT_ LENGTH	16 bit number			
	Default value	***0***	***D***	***E***	***A***

The ORBIT_LENGTH defines the length of the LHC orbit. Default value =3564-2 (= 0DEA hex) bunch crossings. *The BC counters run from 0 until 3564 – 1=3563. For simulation the value 200 -2='00C6\H' has been used.*
The orbit length is used to reset the bunch crossing counter if the BCRES signal is missing, for example when running without LHC signals in LHC orbit simulation mode.
The logic consists of a 16-bit counter+16 bit comparator. The upper 4 bits are always zero.
Check1: The programmed BC LIMIT is compared against the content of the local BC-counter at the arrival time of the common BCRES signal. Any difference sets the error flag BAD_MAX_BC. *The reason for this error could be a bad clock signal or a bad BCRES signal.*
Check2: The local BC-counter is compared against the BC-number from the TTCrx chip. The difference can be read by VME. A change in the difference sets the error status bit BAD_LOCAL_BC.
Remark: For Heavy Ion runs every 5th tick contains a bunch crossing.

8.2.1.7 TTC_Message_Subaddress register

Address A17-A1	Registername	D15-D12	D11 – D8	D7 – D4	D3 – D0
1 0036	TTC SUBADDRESS	Last TTC Message *(read only)*		TTC Subaddress *(write/read)*	

- TTC Subaddress defines the address for individual addressed messages/commands. The command byte is stored in the last address ("…F") of the TTC_DUMP memory. See also chapter 8.2.2. Functions for individual messages are neither defined nor implemented.
- Last TTC Message contains the last system and user message code that has been received from the TTCrx chip.

8.2.1.8 COMMAND PULSE

Set the COMMAND REGISTER bits before sending COMMAND PULSES (bit11…0)
Warning: The VME instruction generates a pulse when a data bit is set equal 1. This "register" cannot be read back ('write only').
The command bits 0…9 are used to simulate the corresponding BGo commands, which are received during data taking by the TTCrx link. See also CMD register 8.2.1.10. **The command pulses (bits 11 to 0) can be used only if the SELECT bits in the Command Register have been set before.**
HARD_RES_VME will stop in any case the BC-Table signal generation.

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0038	COMMAND PULSE	*See description of bits below.*															
	default values	-	-	-	-	-	-	-	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*

Bit 15: **RESET_TTCRX**
RESET_TTCRX =1 sets the RESET_TTCRX Flip-Flop=1 to put the TTCrx into the 'RESET' status. *RESET_TTCRX must not be done during a data-taking run!!!*

Bit 14: **RELEASE_TTCRX**
After several microseconds send RELEASE_TTCRX =1 that clears the RESET_TTCRX Flip-Flop to remove the 'RESET' status of TTCrx chip.

Bit 13: **MONRQST_VME**
MONRQST_VME =1 sends a Monitoring Request.

Bit 12: **SEND_TESTDATA**
SEND_TESTDATA =1 sends test data to all ROPs

Bit 11: **L1A_VME**
L1A _VME=1 simulates a L1A signal, *if SEL_L1A_[2:0]=000 has been set before.*

Bit 10: *not used*

Bit 9: **DO_TEST_EN_VME+)**
DO_TEST_EN_VME =1 sends the command to all GT board to run a calibration cycle, *if SEL_BGO_1,0 =00 has been set before.*

Bit 8: **DO_PRIV_GAP_VME+)**
DO_PRIV_GAP_VME =1 sends the command to all GT board to run a private gap procedure, *if SEL_BGO_1,0 =00 has been set before.*

Bit 7: **DO_PRIV_ORBIT_VME+)**
DO_PRIV_ORBIT_VME =1 sends the command to all GT board to run a private orbit procedure, *if SEL_BGO_1,0 =00 has been set before.*

Bit 6: **RES_ORBIT_VME+)**
RES_ORBIT_VME =1 sends a RESET ORBIT counter command, *if SEL_BGO_1,0 =00 has been set before.*

Bit 5: **START_RUN_VME+)**
START_RUN_VME =1 sets RUN_FF to allow L1A signals to be sent to the boards, *if SEL_BGO_1,0 =00 has been set before.*
The RUN_FF can also be changed by a periodic BGo command, by a BGo from TCS or by a TTC message.

Bit 4: **STOP_RUN_VME+)**
STOP_RUN_VME =1 clears RUN_FF to inhibit L1A signals, *if SEL_BGO_1,0 =00 has been set before.*

Bit 3: **EVCNT_RES_VME *)**
Resets the Event Counter by VME, *if SEL_* EVRES*_1,0 =00 has been set before.*

Bit 2: **L1RES_VME *)**
Send a L1RESET to all boards (alias names RESET, RESYNC), *if SEL_BGO_1,0 =00 has been set before.*
See also chapter 8.3.2 below.

Bit 1: **HARD_RES_VME+)**
Used inside TIM chip only, *if SEL_BGO_1,0 =00 has been set before.*
See also chapter 8.3.1 below.

Bit 0: **BCRES_VME*)**
BCRES_VME =1 sends a bunch counter reset signal, *if SEL_BCRES_[2:0] =000 has been set before.*
For internal orbit generation send BCRES_VME once to start the BC counter logic if there is no TTC connected. Afterwards LHC orbits will be simulated according to the ORBIT_LENGTH register.

**) This command is also sent sent as a fast LVDS signal via the back-plane to all boards in the DTTF and GT crate.*
+) This 'slow command' is sent via the back-plane ROP bus to all boards in the GT crate; but not in the DTTF crate.

8.2.1.9 STATUS Register

Warning:
The read-only STATUS register has got the same VME address as the CMD-Pulses.

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0038	STATUS Register	*See description of bits below.*															

Bit15 – 6 show the status of the TIM Readout circuits and used in the GT crate only.

Bit 15: **OV_BAD_TTC**
OV_BAD_TTC =1: Overflow bit of BAD_L1A_TTC counter counting the number of not concurrent L1A from TCS and TTC.
Bit 14: **TOO_MANY_L1A**
TOO_MANY_L1A = 1: More than 64 L1A are waiting in the request queue whose data have to be extracted from the RingBuffer. *(GT crate only.)*
Bit 13: **L1A_TOO_OLD**
L1A_TOO_OLD = 1: The L1A are waiting in the request queue for more than 960 bunch crossings corresponding to 15/16 of the RingBuffer size. The write pointer is only 64 bunch crossings behind the read pointer and will overtake it soon overwriting the events of the pending L1A's. *(GT crate only.)*
Bit 12: **L1A_OLD_WARNING**
L1A_OLD_WARNING = 1: The L1A are waiting in the request queue for more than 768 bunch crossing corresponding to ¾ of the RingBuffer size. The distance between the write- and the read pointer has decreased already to ¼ of the Ringbuffer. *(GT crate only.)*
Bit 11: **ROBUF_OVF** *// overflow of derandomizing readout buffer*
ROBUF_OVF = 1: The readout buffer FIFO containing the event data is full. More than 'nn' events are already waiting to be transferred to the GTFE board.
'nn' = 1024/3 =341 for readout of 3 bx per L1A.
Bit 10: **WARNING_ROBUF_OVF**
WARNING_ROBUF_OVF = 1: The readout buffer FIFO is filled up to the warning level of 75%. *The TCS (Trigger Control) should decrease the trigger rate.*
Bit 9: **ROBUF_SYNCERR**
ROBUF_SYNCERR = 1: The readout processor ROP didn't read the event data correctly because the FIFO's for trigger data and the bx-number became empty at different time.
Bit 8: **EVNR_OVF**
EVNR_OVF = 1: There was an overflow of the 24 bit Event counter. The bit is used just for information. *It is not an error bit!*
Bit 7: **BAD_LOCAL_EV**
The local and the TTCrx event number are compared to each other. In case of any difference the error bit BAD_LOCAL_EV is set to 1.
Bit 6: ***not used***
Bit 5: ***not used***
Bit 4: **BAD_MAX_BC**

BAD_MAX_BC=1: The local bunch crossing counter does not agree with the ORBIT_LENGTH at arrival time of the BCRES signal.

Bit 3: **BAD_LOCAL_BC**

The local and the TTCrx bunch crossing numbers are compared to each other. If the difference changes then the error bit BAD_LOCAL_BC is set to 1.

Bit 2: **SINERR_TTCRX**

SINERR_TTCRX=1: There was a single bit error in the TTCrx chip.

Bit 1: **DBERR_TTCRX**

DBERR_TTCRX=1: There was a double bit error in the TTCrx chip. It was not corrected.

Bit 0: **TTC_READY**

TTC_READY=1: The TTCrx chip is working correctly.

8.2.1.10 COMMAND Register

Set the COMMAND REGISTER bits before sending COMMAND PULSES.

Address A17-A1	Registername	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1 003A	COMMAND Register							SEL_EVRES		SEL_BGO		SEL_BCRES				SEL_L1A	
	default values	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*

Bit 15: **TIM_SETUPDONE**

TIM_SETUPDONE =1 to tell the TCS board that the setup of the TIM board is done. *This bit sets the Fast Signals TIM_READY and clears TIM_BUSY that can be checked on the TCS board. This bit should be set at the end of a TIM-board-Setup-Program. See also 3.8 Fast Signals to TCS board.*

Bit 13: **TTC_RDY_VME**

The bit is used to simulate a TTC_RDY status´.

Bit 12: **CHECK_TTC_CHAIN**

CHECK_TTC_CHAIN =1 checks if every L1A received directly from the TCS board has also been received via the optical TTC fiber. *The L1A_TCS_DLY delays the L1A sent by the TCS board over the backplane so that it arrives concurrently with the L1A sent via the TTC fiber. See also L1A_TCS_DLY register in 8.2.1.12 below.*

Bit 11: **DIS_BOARD_L9**

DIS_BOARD_L9 =1 stops timing signals to the L9 board. See also 8.2.1.2 for other boards.

Bit 10: **DIS_RO_BUS**

DIS_RO_BUS =1 disables the Readout Request bus (GT crate only).

SELECT 'EVENT COUNTER RESET'

Bit9: SEL_EVRES_1,
Bit8: SEL_EVRES_0

Code Bits 9-8	Selected source of EVENT COUNTER RESET command
00	Only the VME generated EVENT COUNTER RESET is allowed.
01	Take EVCNTRES signal of the TTCrx chip //=default in DTTF and GT crates
10	Take EVENT COUNTER RESET from the active/selected BGO source

11	*Inhibit any EVENT COUNTER RESET*

SELECT source of BGO commands
Bit7: SEL_BGO_1,
Bit6: SEL_BGO_0

Code Bits 7-6	Selected source of BGO commands
00	Only VME generated BGO commands are allowed.
01	BGO from TTCrx chip //=default in DTTF and GT crates
10	Periodic BGO internally generated
11	BGO from TCS board via back-plane

The same selection is valid also for the 'USER MESSAGES', which are generated either periodically or by the TTC system.

SELECT BCRES
Bit5: SEL_BCRES_2
Bit4: SEL_BCRES_1
Bit3: SEL_BCRES_0

Code Bits 5-3	Selected source of BCRES signal
000	Only VME command 'BCRES_VME' is allowed.
001	BCNTRES from TTCrx chip // = default in DTTF crates
010	ORBIT_X from Front Panel (ECL/NIM signal) // =default in GT crate
011	Periodic BCRES internally generated by BC-counter and comparator.
100	BGO command decoder. See also selected source of BGO commands
others	*Codes 101..111 inhibit all sources of BCRES*

Select the BCNTRES signal from the TTCrx chip only if it is sent every orbit.
Use the internally generated BCRES if the TTC system does not send a BCNTRES every orbit. In that case also the ORBIT_LENGTH has to be loaded with the correct value.
The periodic BCRES generator is started either by infrequent BCNTRES of the TTC system or by a VME instruction.
The Global Trigger will take the ECL CLK and ORBIT_X signals to run as precise as possible.

SELECT L1A
Bit5: SEL_L1A_2
Bit4: SEL_L1A_1
Bit3: SEL_L1A_0

Code Bits 2-0	Selected source of L1A signal
000	Only VME command 'L1A_VME' is allowed.
001	L1ACCEPT from TTCrx chip //=default in DTTF and GT crates
010	L1A_X from Front Panel (ECL/NIM signal)
011	Periodic L1A internally generated using the BC-Table
100	L1A from TCS board via back-plane
others	*Codes 101..111 inhibit all sources of L1A*

8.2.1.11 Readout_Command Register

This register is used in GT crate only.

The register contains the control bits to extract data on the TIM chip in case of a L1A.
This logic might not be used.

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 003C	ROCMD _REG	*See description of bits below.*															
	default values	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*

Bit15-12: Write and read accesses are possible but the bits are not used by the control logic.

Bit 11: **RO_LINK_ON**

RO_LINK_ON=1 enables the Channel Link chip to allow transmission of event data to the GTFE readout board.
Default = 0 because normally TIM data are not included into the event data.

Bit10-9: Write and read accesses are possible but the bits are not used by the control logic.

Bit 8: **EN_BC_CHECK =1**

- Checks BC number at arrival time (=3564) of BCRES
    - ➔ *Status bits: ERR_MAX_BC, BAD_MAX_BC*
- Compare local BC counter with TTC BCnr
    - ➔ *Status bits: ERR_LOCAL_BC, BAD_ LOCAL_BC*

Bit 7: **EN_TTC_CHECK=1**

Checks if L1A from TTC and L1A from TCS arrive concurrently.
- ➔ *Status bits: ERR_L1A_TTC, BAD_ L1A_TTC*

Bit 6: **EN_EVNR_CHECK=1**

Compares with TTC EVnr with local Evcntr
- ➔ *Status bits: ERR_LOCAL_EV, BAD_ LOCAL_EV*

Bit 5: **EN_L1AQUEUE_CHECK=1**

The L1AQUEUE will be checked all the time. The check logic generates the warning bit L1A_OLD_WARN and the sync-error bits L1A_TOO_OLD and TOO_MANY_L1A. See bit 1 below how to stop the L1AQUEUE completely.
EN_L1AQUEUE_CHECK=1 allows to send the warning and sync-error states as Fast Signals to the TCS board.

Bit 4: **EN_ROBUF_CHECK**

EN_ROBUF_CHECK=1 checks if the readout buffer ROBUF is almost or really full. In these cases the warning bit WARNING_ROBUF_OVF and the sync-error bit ROBUF_OVF are set and sent also to the TCS board.

Bit 3: **FREEZE_RIBUF_IF_ERROR**

FREEZE_RIBUF_IF_ERROR =1 inhibits data transfer into the RING BUFFER in case of an error. This bit is useful to check the monitored data after an error.

Bit 2: **FREEZE_RIBUF**

FREEZE_RIBUF =1 inhibits data transfer into the RING BUFFER. This bit is used for tests with constant RING BUFFER content.

Bit 1: **INHIB_L1A_ON_TIM**

INHIB_L1A_ON_TIM =1 inhibits L1A's on the TIM board to remove TIM data from the Event data

Bit 0: **INVERT_ROPMUX**

INVERT_ROPMUX =1 inverts the clock for the ROP multiplexer. The multiplexer combines the 40 MHz L1A-event and monitoring data into phase A and phase B of 80 MHz parallel data to be transmitted by a Channel Link to the GTFE readout board. See page 9 of TIM chip schematic. The bit defines time order.
Default: INVERT_ROPMUX =0 sends the L1A-event first in phase A.
(to be checked???)

8.2.1.12 Delay L1A from TCS Register

The L1A_TCS_DLY delays the L1A sent by the TCS board over the backplane so that it arrives concurrently with the L1A sent via the TTC fiber. See also bit 12 in the CMD register 8.2.1.10 above.

The total delay consists of (DLY_L +1) + (DLY_H +1). Each hex-number programs a 16bx delay circuit. ***The value 'F' means 'no delay'***. The minimum delay 'FF' =0 bx and the maximum delay 'EE'=32 bx.

Address A17-A1	Registername	D15 ....D8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 003E	DLY_L1A_TCS	Not used	RES_DLY_H				RES_DLY_L			

8.2.1.13 ROBUF_PAR Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0040	ROBUF_ PAR	*NR_ROBUF*								*RO_LENGTH*							

Used by GT only.
Write and read access.
RO_LENGTH = < 255 (1024 / #of BC per event)
Examples: max=1024/3 =341; max=1024/5=204)

8.2.1.14 Record Identifier Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0042	IDENTIFIER	*Record Identifier for readout data*															

Used by GT only.
Write and read access.

8.2.1.15 IDLE_VALUE Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0044	IDLE_ VALUE	*Code for IDLE word between readout records*															

Used by GT only.
Write and read access.

8.2.1.16 EOF_VALUE Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0046	EOF_ VALUE	*Code for EOF (=end of file) word in readout record*															

Used by GT only.
Write and read access.

8.2.1.17 TESTDATA Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0048	TESTDATA	*Test data for RO_RQST bus*															

Used by GT only.
Test data to be sent via the RO-RQST bus to the boards in the crate.
Write and read access.

8.2.1.18 MON_RQST ID Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 004A	MON_RQST _ID	*Identifier for Monitoring RQST*															

Used by GT only.
Identifier is used to distinguish Monitoring data from L1A data in the GTFE board.
Write and read access.

8.2.1.19 ROBUF_BX FIFO Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 004C	ROBUF_BX FIFO	**BC number corresponding to trigger data in ROBUF_A FIFO**															

Used by GT only.
NO Write Access!! Read access only.

8.2.1.20 ROBUF_A FIFO Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 004E	ROBUF_A FIFO	**Data bits of BC as stored in ROBUF_BX FIFO**															

Definition of data bits has to be done.
Used by GT only.
NO Write Access!! Read access only.

8.2.1.21 NBAD_L1A_TTC Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0050	BAD_L1A_ TTC Register	*See description of bits below.*															

Read access only.

8.2.1.22 BCDIFF Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0052	BC_DIFF Register	*BC difference between local BC counter and TTCrx*															

Read access only. To check for hardware errors.

8.2.1.23 MAX_BCNR Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0054	MAX_ BCNR Register	*Maximum value of Bunch Crossing Counter*															

Read access only. To check for hardware errors.

8.2.1.24 TTC_BCNR Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0056	TTC_BCNR Register	*Bunch counter number from TTCrx chip.*															

Read access only.

8.2.1.25 LOC_EVNR_H Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0058	LOC_EVNR _H Register	*High part of Local Event number*															

Read access only.

8.2.1.26 LOC_EVNR_L Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 005A	LOC_EVNR _L Register	*Low part of Local Event number*															

Read access only.

8.2.1.27 TTC_EVNRH Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 005C	TTC_EVNR H	*High part of Event number from the TTCrx chip*															

Read access only.

8.2.1.28 TTC_EVNRL Register

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 005E	TTC_EVNR L	*Low part of Event number from the TTCrx chip*															

Read access only.

8.2.1.29 CHIP IDENTIFIER Registers

The DAQ group wants 32 bit identifiers for the chips. This address is reserved for that purpose. The bit format is preliminary.

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0060	CHIP_ID_H	*chip type bits 31...16: =0001 for GT crate*															
1 0062	CHIP_ID_L	*chip type bits 15...0: =42x1 /hardwired by design*															

Bits 15-12: = 4 for TIM card	Bits 12 - 8: = 2 for TIM chip
Bits 7 - 4: = card#	Bits 3 - 0: = 1 chip# *//There is only 1 TIM chip on board.*

8.2.1.30 CHIP VERSION Registers

Version numbers 1…1000(hex) are test designs.
Version numbers 1000…FFFF FFFF (hex) are standard designs.

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0064	CHIP_ VERSION_H	*Version number bits 31…16*															
1 0066	CHIP_ VERSION_L	*Version number bits 15…0*															

Example: Version_1001: CHIP_VERSION_H = 0000; CHIP_VERSION_L = 1001

8.2.2 TTC dump addresses

The 16 addresses below contain the TTCrx registers of the last dump action. See also description of TTCvi module and of TTCrx chip.
Only lower 8 bits are used. Read access only.

Address A17-A1	Registername	D 15	D 14	D 13	D 12	D 11	D 10	D 9	D 8	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
1 0080	xxxx	------------------								*xxxx*							
1 0082	xxxx	------------------								*xxxx*							
1 0084	xxxx	------------------								*xxxx*							
1 0086	xxxx	------------------								*xxxx*							
1 0088	xxxx	------------------								*xxxx*							
1 008A	xxxx	------------------								*xxxx*							
1 008C	xxxx	------------------								*xxxx*							
1 008E	xxxx	------------------								*xxxx*							
1 0090	xxxx	------------------								*xxxx*							
1 0092	xxxx	------------------								*xxxx*							
1 0094	xxxx	------------------								*xxxx*							
1 0096	xxxx	------------------								*xxxx*							
1 0098	xxxx	------------------								*xxxx*							
1 009A	xxxx	------------------								*xxxx*							
1 009C	xxxx	------------------								*xxxx*							
1 009E	xxxx	------------------								*xxxx*							

8.2.3 BC – Table for Simulation

Address range: 0 2000 – 0 3FFE for 4k memory of BC table
The address corresponds to the bunch-crossing (BC) number. During Signal generation a BC-counter provides the read addresses. If a bit in the BC-Table is set to '1' at address 'aa' then a signal pulse will be sent at BC-number 'aa'. The signals are sent every n-th orbit as defined by the SIMULATION PERIOD register.
Bit 15-12 not implemented; VME access is not possible
Bit11,10: function not defined; VME access is possible
Bit 9: PER_MONRQST // sends a trigger to read out monitoring data.
Bit 8: PER_L1A // sends a trigger to read event data; periodic simulation of L1A
Bit 7: MESSG_SIM7 // simulate message bit 7 for user messages
Bit 6: MESSG_SIM6 // simulate message bit 6 for user messages
Bit 5: SIM_USR_MESSG_STRB // simulate user message strobe
Bit 4: PER_BGO_4 // simulate a BGO STROBE command

Bit 3: PER_BGO_3	// simulate bit3 of a BGO command
Bit 2: PER_BGO_2	// simulate bit2 of a BGO command
Bit 1: PER_BGO_1	// simulate bit1 of a BGO command
Bit 0: PER_BGO_0	// simulate bit0 of a BGO command

8.2.3.1 BGO codes

0000 = *not used*
0001 = *'BC0'...not used in TIM chip*
0010 = TEST_ENABLE
0011 = PRIVATE_GAP
0100 = PRIVATE_ORBIT
0101 = L1RESET (or RESYNC)
0110 = HARD_RESET
0111 = RESET_EVENT_COUNTER
1000 = RESET_ORBIT
1001 = START RUN
1010 = STOP RUN
1011…1111*….free for private purpose*

8.2.4 RING BUFFER 1k memory

Address range: 0 4000 – 0 47FE for 1k memory of Ring buffer.
First set FREEZE_RIBUF=1 to stop any input data and set INHIB_L1A_ON_TIM=1 in the RO_CMD register to stop triggered readout. Then it is possible to access the Ring Buffer memory by VME.
To check external or simulated signals often just freeze the Ring Buffer and read data from all addresses.
INPUT bits for the Ring-buffer:

Bit15: L1A_FROM_TCS	*// arrives via the back-plane from the TCS board*
Bit 14: L1A_FROM_TCS_DLYED	*// check programmed delay*
Bit13: L1A_FROM_TTC	*// Bit 13 and 14 should appear at the same time*
	// if the delay for L1A_TCS is set correctly.
Bit12: L1A_FROM_LEMO	*// External trigger input*
Bit11: PER_MONRQST	*// simulated periodic Monitoring Request*
Bit10: PER_L1A	*// simulated periodic L1A*
Bit 9: 0	*// not used*
Bit 8: RES_EVCNT	*// RESET Event Counter generated by any source*
Bit 7: ORBIT_P	*// Pulse at begin of ORBIT signal (LEMO, ECL)*
Bit 6: BCRES_LEMO	*// Delayed BCRES from Orbit signal*
Bit 5: BCNT_RES_TTC	*// Bunch Counter Reset from TTC*
Bit 4: BCRES_TTC	*// Bunch Counter Reset from TTC after optional delay*
	// BCRES_LEMO and BCRES_TTC should appear at the
	// same time if both are connected.
Bit 3: L1_RESET	*// generated by any source*
Bit 2: PRIV_ORBIT	*// generated by System Message or any BGO command*
Bit 1: PRIV_GAP	*// generated by System Message or any BGO command*
Bit 0: TEST_EN	*// generated by System Message or any BGO command*

8.3 RESET trees

The L1_RESET signal is forwarded to all boards in the VME crate. HARD_RES is used only inside the TIM chip. Both reset signals are generated either by software (VME) or by BGo signals arriving from the TCS (Trigger Control System) via the TTC optical link or by Message signals from the TTC link or are simulated periodically by the BC Table

8.3.1 HARD_RES

Signal sources:

- HARD_RES_VME *(software)*
- HARDRES_MSG: re*ceived as MESSAGE bits from TTC*
- HARDRES_BGO:
    - TCS_BGO: BGO signals received from TCS via TTC links
    - PER_BGO: BGO signals simulated by the BC-TABLE inside the TIM chip

Functions:

- HARD_RES_VME resets EN_BCTABLE circuit.
- HARD_RES = HARD_RES_VME + HARDRES_MSG + HARDRES_BGO
    - Clears the local EVENT COUNTER
    - Is combined with L1_RESET to make CLR_ALL (see below).

8.3.2 L1_RESET

Signal sources:

- L1RES_VME *(software)*
- L1RES_MSG: *received as MESSAGE bits from TTC*
- L1RES_BGO:
    - TCS_BGO: BGO signals received from TCS via TTC links
    - PER_BGO: BGO signals simulated by the BC-TABLE inside the TIM chip

Functions:

- L1_RESET resets/resynchronizes all boards in the VME crate.
- **Inside the TIM chip???**
- Is combined with HARD_RES to make CLR_ALL (see below).

8.3.3 CLR_ALL

- Resets error flag and checking counters
- Resets the RUN_FF, TEST_ENABLE
- Resets the L1A_queue and the ROP_EVENT controller circuit (for GT crate only)

8.3.4 STOP_RUN

- STOP_RUN also clears TEST_ENABLE FF

8.4 Pin assignment TIM chip

The TIMING chip is a FPGA from XILINX, called XC2V1000-4FG456C. There is an EXCEL-file containing the pin assignment of the TIM CHIP (see tim_chip.xls). The pintable is extracted from the XILINX datasheet of the FG456-package (ds031-4.pdf). (Extraction was done using Acrobat Reader 4.0 with ***Zusatzmodule/ACE*** enabled or using Acrobat4. Select the table that should be extracted and use right-mouse-button ***Extract Table.*** Save it as text file. Start EXCEL , open the text file and convert it. Do the same for each page.

The batch-file ..*\tim_check\make_pin_nr_tim_chip.bat* provides a possibility "**to make" pin-numbers** of symbols in VIEWDRAW from the EXCEL-file.

The batch-file ..*\tim_check\check_symbol_tim_chip.bat* provides a possibility "**to compare" pin-numbers** of symbols and the EXCEL-file (text-file of EXCEL-sheet). The comparision output is written into the file ..*\tim_check\tim_check_symbol.log*

8.5 Symbol names

The naming convention of the symbols for VIEWDRAW schematics of Timing chip (..\Tim6U\sym):

tim.1	→	Timing chip
tim_clk.1	→	CLocK generation for distribution in GTL crate
tim_conf.1	→	XILINX CONFiguration pins of Timing chip
tim_jtag.1	→	JTAG pins of Timing chip

tim_pan.1	→	signals from/to front-I/O (PANel)
tim_rop.1	→	ReadOutProcessor unit in Timing chip
tim_rorq.1	→	ReadOutReQuest unit in Timing chip
tim_term.1	→	TERMination-resistors feature in Timing chip
tim_tsig.1	→	fast Timing SIGnals
tim_ttc.1	→	signals from/to TTCrx chip
tim_ttfg.1	→	signals from Timing chip to TCS, FDL card and GTFE card
tim_vme.1	→	signals from/to VME chip

8.6 Configuration ??

PROM CHIP….Preis, Lieferzeiten ???

Text fehlt noch!!!!

Configuration methods
Configure by VMEbus
Configure by PROM
Configure by JTAG

8.7 Special functions

8.7.1 Power-On Power Supply Requirements

The V_{CCINT}, V_{CCAUX}, and V_{CCO} power supplies shall ramp on no faster than 100 ms and no slower than 50 ms. V_{CCAUX} and V_{CCO} for bank 4 must be connected together. If any V_{CCO} bank powers up before V_{CCAUX}, then each bank draws up to 600 mA (=transient current peak; does not harm the device)

Power On current	XC2V1000
I_{CCINT_MIN}	500 mA
I_{CCAUX_MIN}	250 mA
I_{CCO_MIN}	10 mA

8.7.2 Power-down sequence

The command register bit PWRDWN=1 in the VME chip assigns the PWRDWN_B signal (active low) to set the TIM chip into a low-power, inactive mode. The bi-directional IO-pin of the VME chip is connected to a tri-state driver and an input buffer to sense also the status of the Virtex-II chip after releasing the driver. The PWRDWN bit in the status register of the VME chip reflects the state of the Virtex-II chip.

(From the *Virtex-II User Guide.)*
The power-down sequence enables a designer to set the device into a low-power, inactive state. The sequence is initiated by pulling the PWRDWN_B pin Low. The BitGen PWRDWN_STAT option is no longer supported. To monitor power-down status, observe the PWRDWN_B pin. When asserted, power-down has completed. After a successful wake-up, the status pin de-asserts. While powered down, the only active pins are the PWRDWN_B and DONE. All inputs are off and all outputs are 3-stated. While in the POWERDOWN state, the Power On Reset (POR) circuit is still active, but it does not reset the device if VCCINT, VCCO or VCCAUX falls below its minimum value. The POR circuit waits until the PWRDWN_B pin is released before resetting the device. Also, the PROG_B pin is not sampled while the device is in the POWERDOWN state. The PROG_B pin becomes active when the PWRDWN_B pin is released. Therefore, the device cannot be reset while in the POWERDOWN state. The wake-up sequence is the reverse of the power-down sequence.

8.7.3 Maximum input voltage =+3.6V

Never higher than 4.0 V !!!

8.7.4 Termination Resistors

The VirtexII chip contains DCI circuits to control the impedance of the io-pins digitally. This property saves many termination resistors on the board and avoids stubs on terminated nets. For each bank a pair of resistors

that is connected to VCCO=3.3V and GND, is used as reference for the termination. The DCI function is enabled in the TIM chip design for each io-pin individually.
The pins VRN are connected over R=50 Ohm 1% to VCCO=3.3V and the VRP pins are connected over R=50 Ohm, 1% to GND.

8.7.5 Hot Swap Enable

(From the *Virtex-II User Guide.)*
Prior to configuration, all outputs not involved in configuration are forced into their high-impedance state. The pull-down resistors and the weak-keeper circuits are inactive. The dedicated pin HSWAP_EN controls the pull-up resistors prior to configuration. By default, HSWAP_EN is set high, which disables the pull-up resistors on user I/O pins. When HSWAP_EN is set low, the pull-up resistors are activated on user I/O pins.
(From the *Virtex-II User Guide pg.81)*
Depending on the system design, several configuration modes are supported, selectable via mode pins. The mode pins M2, M1 and M0 are dedicated pins. An additional pin, HSWAP_EN is used in conjunction with the mode pins to select whether user I/O pins have pull-ups during configuration. By default, HSWAP_EN is tied High (internal pull-up) which shuts off the pull-ups on the user I/O pins during con-figuration. When HSWAP_EN is tied Low, user I/Os have pull-ups during configuration.
TIM board:
The signal HSWAP_EN is connected on the board to a jumper to connect HSWAP_EN to GND to enable the internal pull-up resistors during configuration. If the jumper is removed (=default) the pull-up resistors are disabled during configuration as described above.

9 JTAG

TDI-TDO chain: Insert a jumper for each chip to remove it from the chain if necessary.

There are some questions with the JTAG chains on board which have to be discussed:

- How many chains on board?
- What to do with TTCrx JTAG chain?
- How to implement the JTAG solution of DTTF-crate which is made by VMEbus?

10 Front panel

TO BE DEFINED!!!!

On the front panel there are the following components arranged:

10.1 LEDs

- Red LED as indicator that the module is in an INACTIVE state, ready for removal.
- Green LED for RUNNING state, module is able to run in the specified meaning.
- INACTIVE and RUNNING LED should be placed near „Interlock"-switch on the top of front panel.
- Red LED for TTCRX_ERR, indicator of errors on the TTCrx-board. This signal is a status or a pulse ????? Where to place??
- Green LED for TTCREADY. This signal is a status or a pulse ????? Where to place??
- Red LED for L1ACCEPT. How long should to pulse be for LED ??????? Where to place??
- Green LED for VMEbus-access. VME_LED signal out of VME-chip indicates a VMEbus-access to the module (AS* is active) ??????? How long should to pulse be for LED ?? Where to place??

10.2 Switches

- „Interlock"-switch forces the module in INACTIVE/RUNNIG state. Should be placed on the top of the front panel.
- CLK_LEMO-switch selects CLOCK40DES1 or LOCAL_CLK to lemo-connector. Where to place??

- LOCAL_CLK-switch selects oszillatorclock or external clock to LOCAL_CLK. Where to place??
- SEL_TTCLK-switch ???????????. Where to place??

10.3 LEMO-connectors

- LEMO-outputs for 8 clock signals (CKO1..8), level 1 accept signal (L1A_TTC), bunch counter reset signal (BCRES_TTC) and a reset signal (RESET_TTC) from TTCrx board. Where to place??
- The source for the 8 clock output signals can be selected by jumpers.
- Another LEMO output is used to monitor either the TTCrx or the Oscillator clock signal, that has been selected by a front panel switch.
- LEMO-inputs of external clock signal (CLK_X), external level 1 accept signal (L1A_X), external bunch counter reset signal (BCRES_X) and an external reset signal (RESET_X). Where to place??
- Remark: The ABTE16245 have been selected as 50 Ohm drivers and provide +90/-60 mA. The 25 Ohm resistors at the B-side of the ABTE16245 reduce under/overshoot of the signals. Therefore no other termination resistors are foreseen between the LEMO output driver and the clock sources.
- Optical-link to/from TTCrx-board
- The TTCrx-board is placed near the front of the module so that the optical-link-connector is connectable.

11 Hot Swap

The TIM-card is designed to work in a hot-swap-system. There are made a set of precaution to prevent damages or incorrect behaviours of the used devices.
Two push-buttons on the front panel are forseen to set the module in a defined state to handle with it. Inserting the module in a powered system is possible because all drivers are disabled with the INACTIVE signal, which is generated at powerup of VCCBIAS, which is supplied with staged length contact of the 160 pin VME64 connector (D32). All the devices which generate the enable signals for the drivers to the backplane are supplied with VCCBIAS or LV3V3BIAS. LV3V3BIAS is made from VCCBIAS (D1) when used in the system with the 6U-backplane. In the 9U-backplane LV3V3BIAS will be supplied on a staged length contact of the 160 pin VME64 connector (D1).
Inserting the module in a powered crate keeps the module in the INACTIVE state until the RUNNUNG push-button is pressed. If the module is inserted in a unpowered crate, after powerup the INACTIVE state is set, but the RUNNING state is set with signal SET_RUNNING which is generated in the VME chip by VMEbus SYSRES*.
Removing the module from a powered system is made by pressing the INACTIVE push-button to set the module in an INACTIVE state which disables the drivers and remove the module from crate.
The INACTIVE and RUNNING states are indicated with leds.

The definitions of the hot-swap-system for GT-crate modules are written in the file GT_liveinsertion.doc.- ALTE VERSION!!!

11.1 VME64 connector 160 pins

The connector contains 4 leading pins D1, D32 for +5V and D2, D31 for GND.

11.2 VMEbus buffer driver SN74ABTE16245DL

The A port is foreseen for the VMEbus side (I_{OH}=-60mA, I_{OL}=90mA, 25 Ohm incident wave switching).

- Internal pull-up resistor on OE keeps outputs in high-impedance state during power up or power down.
- V_{CC} BIAS pin minimizes signal distortion during Live Insertion.

Live Insertion SDYA012.pdf from Texas Instruments:
The ETL circuits (for example, SN74ABTE16245) have an additional supply voltage connection (V_{CC} BIAS). This feeds the circuit, which generates the voltage bias mentioned above and, together with the V_{CC} connection, controls the switching on and off of the voltage bias. Figure 12 shows the simplified circuit diagram of this part of the circuit. It does not include the power-up 3-state circuit (see Figure 2), which also is contained in these bus interface circuits, and which switches all outputs into the high-impedance state (3-state) at a supply voltage below about 2.5 V.
TIM board:
The leading +5V voltage pins are connected to the VCCBIAS pin of the ABTE16245. It's /OE pin is controlled by the INACTIVE signal.

11.2.1 Interlock Switch

The interlock switch is made of two push-buttons one to setting the module INACTIVE and one to set it RUNNING.Functionality in a powered system:

Insertion	
Insert the module	→ INACTIVE state, all drivers disabled
Push RUNNING button	→ RUNNING state, all drivers enabled
Removing	
Push INACTIVE button	→ INACTIVE state, all drivers disabled
Remove the module	

Functionality in an unpowered system:

powerup	→ INACTIVE state, all drivers disabled
with SYSRES*	→ SET_RUNNING signal, RUNNING state, all drivers enabled

11.3 VME chip

The VME chip does not drive directly any VMEbus signals. Until the end of configuration all IO-pins are in high impedance state. No special protection is foreseen.

11.4 DTACK and BERR driver 74F38

The open collector outputs of the 74F38 are insensitive to high voltage levels as long as they are below +7 Volt. No protection is necessary for live insertion. Moreover the outputs are kept inactive by INACTIVE signal.

11.5 LVDS drivers SN75LVDS387

The SN75LVDS387 drive point-to-point lines. Therefore no bus problems arise. The drivers are in high impedance state until the TIM chip has been configured. Before removing the board their outputs are locked to high impedance by the interlock switch via the TIM chip.
When not powered up the outputs of the SN75LVDS387 see only differential inputs of LVDS receivers.

11.6 Signal driver ABT18245

The ABT18245 drive point-to-point lines. Therefore no bus problems arise. The drivers are in high impedance state until the TIM chip has been configured. Before removing the board their outputs are locked to high impedance by the interlock switch via the TIM chip.
When not powered up the ABT18245 outputs are insensitive against voltage levels below 7 Volt.

11.7 MOS-FET swich 74CBTLV16800

This device is used to isolate signals from the backplane, which are driven from the Virtex-chip.

11.8 TIM chip Virtex-II XC2V1000FG456-4

Voltages more than +0.7V higher than the actual VCCOUT level can destroy Virtex-II pins because of the diode from the pin to VCCOUT. Therefore no IO-pin of the Virtex-II chip is connected to the back-plane to protect the chip during live insertion. The connection to +5V devices like ABT18245 is made with a serial R of 50Ω.

See XAPP251 Hot-Swapping Virtex-II Devices from Xilinx.
Each IO-pin contains a diode from pin to VCCO and from GND to pin.
In the best case, ground and VCC pins mate first and the VCC distribution on the board feeds all the positive supply pins before any signal pins mate. When the on-board VCC distribution is slow and signal pins mate before the supply voltage is completely powered, then any active High signal pin might drive current through the diode into the VCC pin.

11.9 JTAG

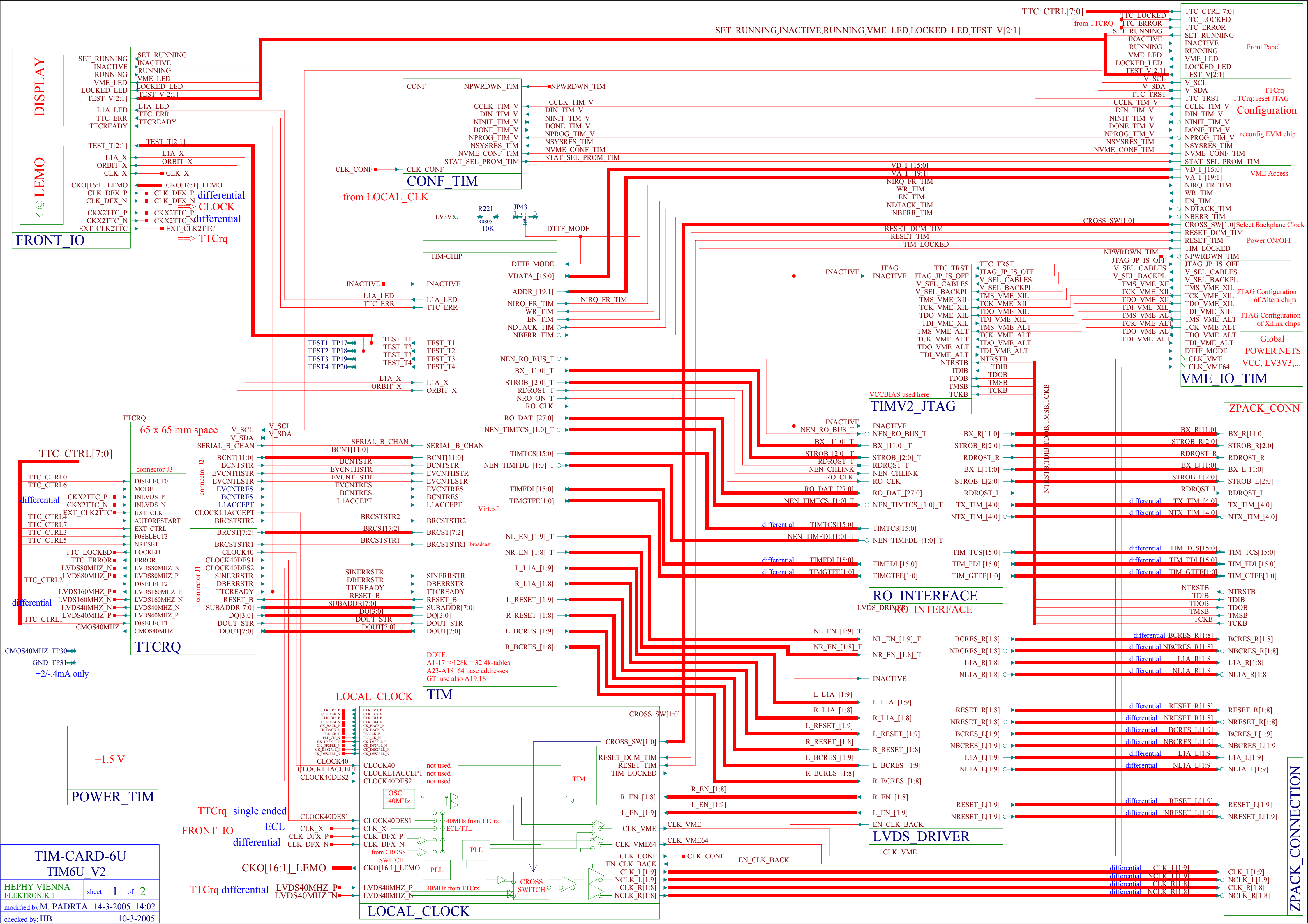
JTAG signals are isolated from the backplane with a 74CBT3245A device.

11.10 Hot swap questions

ESD
IACK IN-OUT on VMEbus

11.11 Clock Signals on 6U-Backplane

TimCard	Backplane	Slot
L1	GTL2	14
L2	GTL1	13
L3	PSB6	12
L4	PSB5	11
L5	PSB4	10
L6	PSB3	9
L7	GMU1	7
L8	PSB1	5
R1	FDL1	16
R2	FDL2	17
R3	GTFE	18
R4	GTL3	19
R5	PSB7	20
R6	PSB8	21
R7	GMU2	8
R8	PSB2	6



TIM-CARD-6U

TIM6U_V2

HEPHY VIENNA
ELEKTRONIK 1

sheet 1 of 2

modified by M. PADRTA 14-3-2005 14:02

checked by: HB 10-3-2005

default attribute values for nets, see sheet 1

Rules for design:

Never use unterminated ABT signals!!!!

ABT to Virtex nets have to be terminated, even very short line.
Without term., oscillations up to 5.5V destroy Virtex pins

Virtex2_24mA driver with serial 50ohm is better than termination at receiver side.
Virtex2_24mA slow+serial 50ohm also over backplane and 2mm conn is ok.

ALS157 can drive jtag TCK up to 20 MHz without termination (simulated with 6 rec.)

CLK_L. bzw _R. und NCLK_L. bzw _R. sind differentiell gleich lang, dist=..

CLK_L8 +2cm = CLK_L7 usw. bis _L1, auch für R

BCRES_R. bzw _L. und NBCRES_R. bzw _L. sind differentiell gleich lang, dist=..
L1A_R. bzw _L. und NL1A_R. bzw _L. sind differentiell gleich lang, dist=..
RESET_R. bzw _L. und NRESET_R. bzw _L. sind differentiell gleich lang, dist=..

TX_TIM. und NTX_TIM. sind differentiell gleich lang, dist=..
TXCLK_TIM. und NTXCLK_TIM. sind differentiell gleich lang, dist=..

Längenunterschied max. 2cm??

CRITICAL NETS (as short as possible): CLK_X, CLK40DES1
CKPLL, CLK40PLL, L1A_X, ORBIT_X
CLK_BACK, CLK_EXT, CLK_OSC, RO_CLK

NETS from/to LEMOs: distance = 2-3x of width, 50 OHM

POWER- AREAs

Dedicated Plane for LV3V3BIAS+VCCBIAS???

LVM2, LVM5, LVM5_PLL kleine Flächen und sehr breite Leitung in MIXED-Plane

Nets to POWER- pins on IC38,39,40,41,43...moeglichst breit

GND-AREA in Top-layer below IC41:..free, not covered by isolation, with many GND-Vias
See V040ME01 (Z-Comm) : AN101 about soldering

VME nets from P1as short as possible: VME Rule 6.24: <5 cm

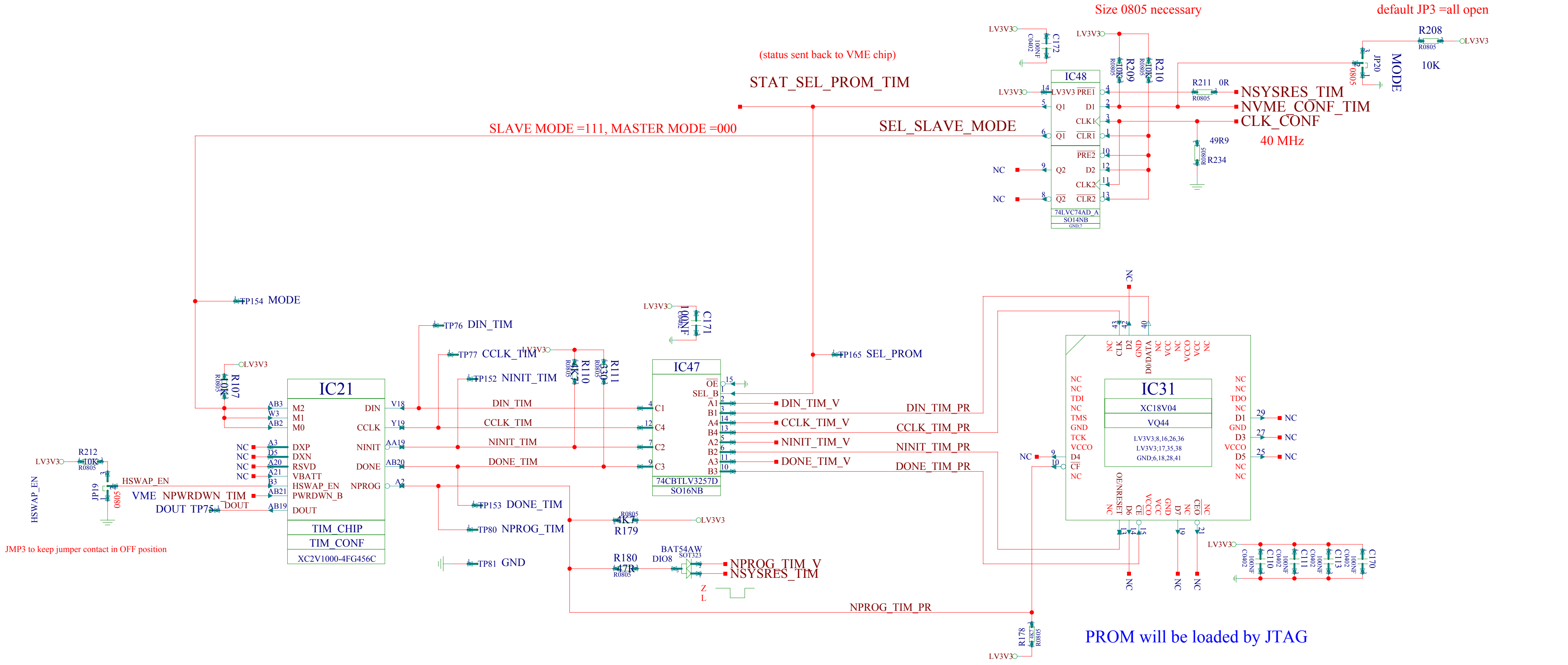
TIM-CARD-6U	
TIM6U_V2	
HEPHY VIENNA ELEKTRONIK 1	sheet 2 of 2
modified by: H. BERGAUER	9-6-2002_14:03
checked by: CHECKER	0-00-0000_00:00

SWITCH BETWEEN PROM - OR VME- PROGRAMMING

MODE 2-3 MASTER MODE permanent set: M2=M1=M0=0 <== Proms
MODE 1-2 SLAVE MODE permanent set: M2=M1=M0=1 <== VME

MODE open MODE selected by VME <== default

NVME_CONF_TIM =1 ==> MASTER MODE = configure by PROM ..default
NVME_CONF_TIM =0 ==> SLAVE MODE = configure by VME



NPROG_TIM_V can also start reconfiguration from Proms if FPGA=master mode.

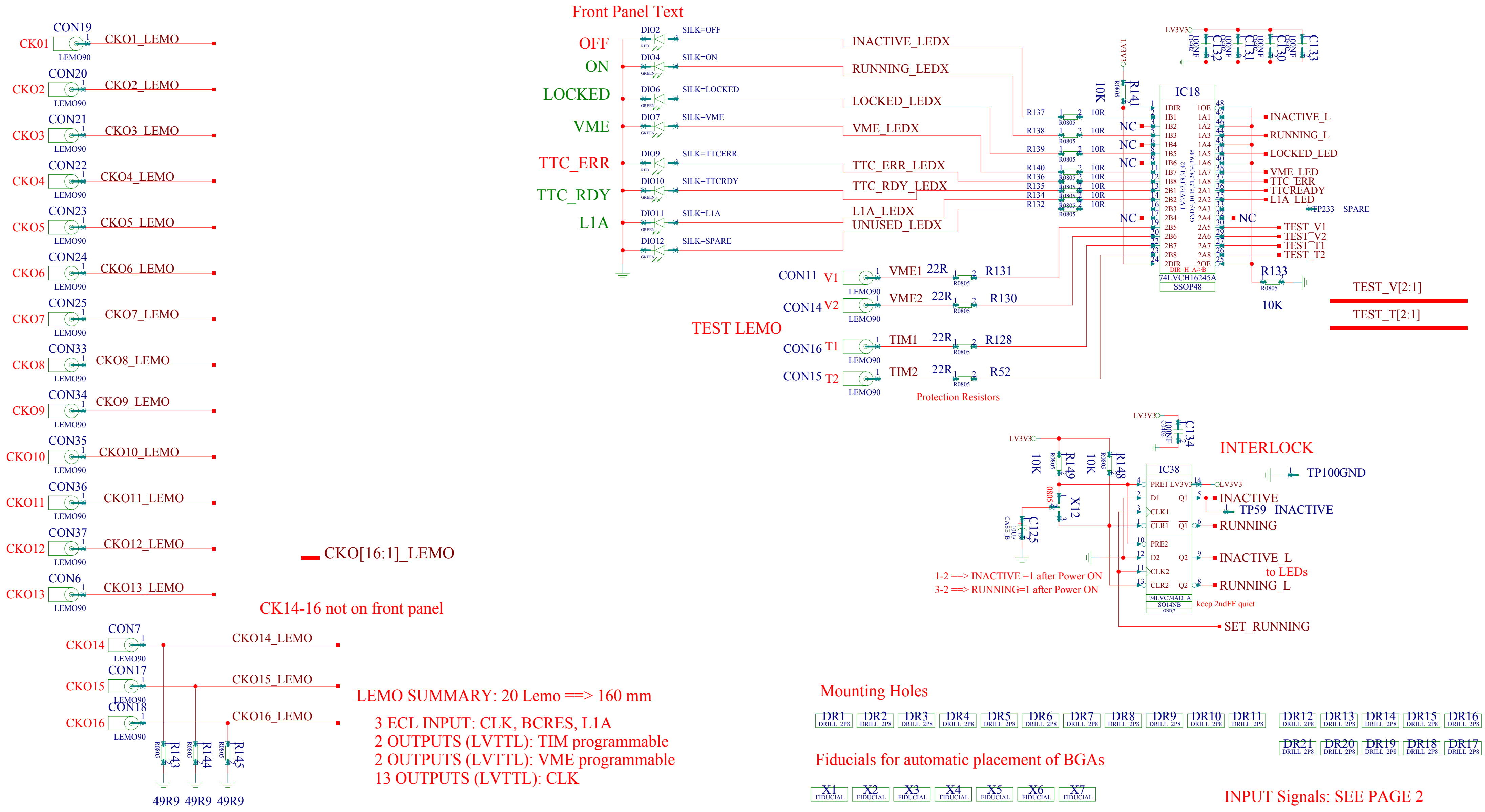
NSYSRES_TIM switches to Master Mode and starts reconfiguration from Prom.

TIM6U_V2

CONF_TIM

HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by: A.TAUROK	10-3-2005_9:18
checked by: HB	10-3-2005

LEDS mounted on Front Panel beside LEMOs



LEMO SUMMARY: 20 Lemo ==> 160 mm

3 ECL INPUT: CLK, BCRES, L1A
2 OUTPUTS (LVTTL): TIM programmable
2 OUTPUTS (LVTTL): VME programmable
13 OUTPUTS (LVTTL): CLK

INPUT Signals: SEE PAGE 2

unused PLL CLK OUTPUTS terminated with 50 Ohm

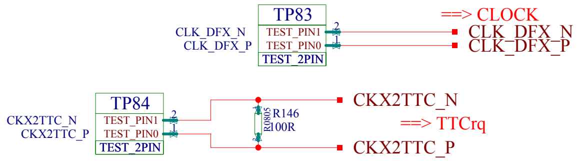
Auxiliary outputs: LEMOs positions on left lower edge of board
TIM6U used in Global Trigger 9U crate:
Auxiliary outputs will be connected to lower 3U part of a 9U front panel
to front panel -"through" connectors
TIM6U used in DTTF_ 9U crate:
Auxiliary outputs not used.

200 mm available on 6U front panel (1 mounting block in middle)
7.5mm/Lemo: ==>8mm x20LEMOs=160mm

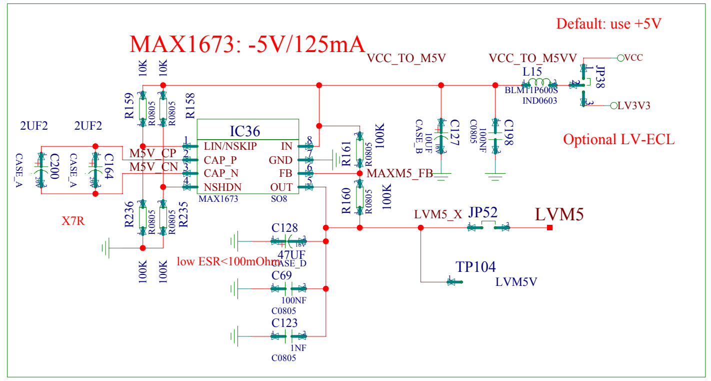
TIM6U_V2	
FRONT_IO	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 2
modified by: M. PADRTA	4-4-2005_14:57
checked by: HB	10-3-2005

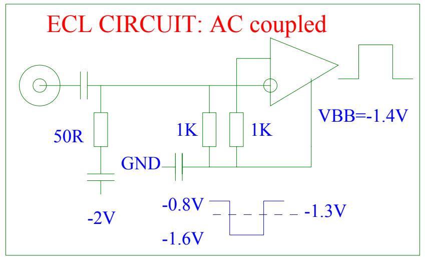
INPUT Signals

Optional LVDS CLOCK INPUT for TTC_QPLL and CLOCK circuit

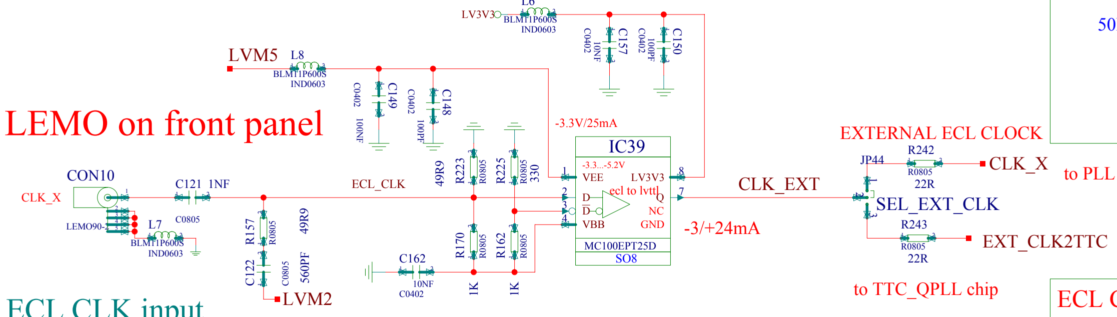


PECL+3.3V,10k: Vih=2.2....2.5 Vil=1.4....1.75, VBB=2.0
PECL+5V,10k: Vih=3.9..4.0..4.2, Vil=3.0..3.3...3.5, VBB=+3.7
ECL-5V,10k: Vih=-1.1..-1.0...-0.8, Vil=-1.9..-1.7..-1.5, VBB=-1.3
PECL(+5V) to LVTTL: LVELT23, EPT23, EPT21
LVPECL(+3.3V) to LVTTL: LVELT23, EPT23, EPT21,EPT26
ECL (-5.2V) to TTL: MC10H125(4x), H601(9x), ELT25
ECL(-5.2V) to LVTTL: EPT25(diff)
LVECL (-3.3V) to TTL: ELT25, //to LVTTL: EPT25(diff)

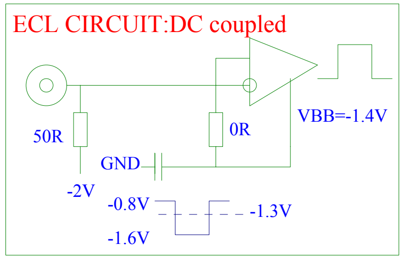


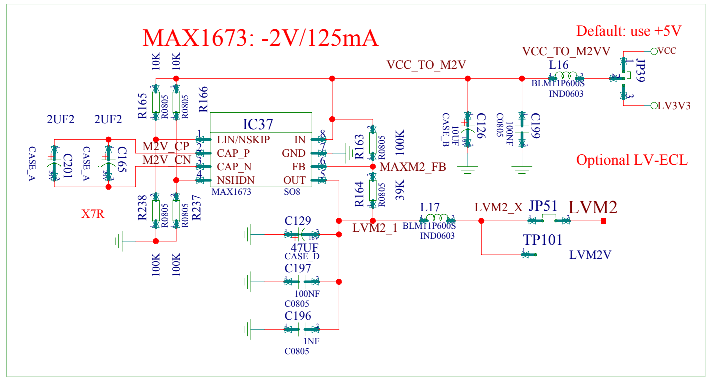


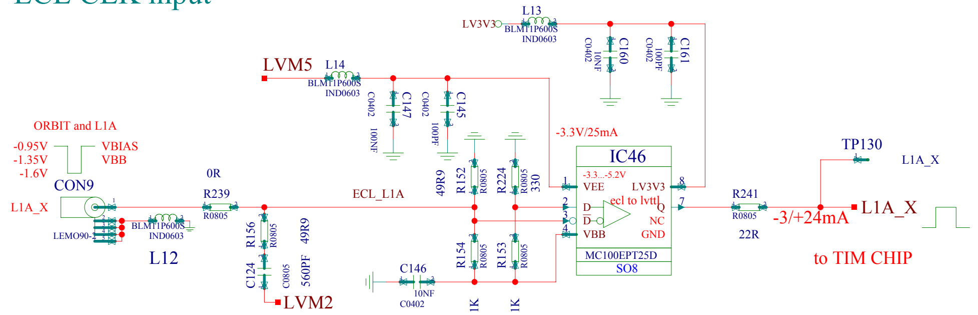
LEMO on front panel

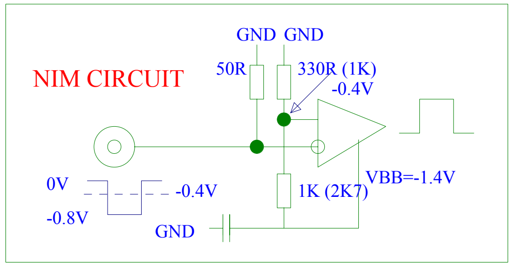


ECL CLK input



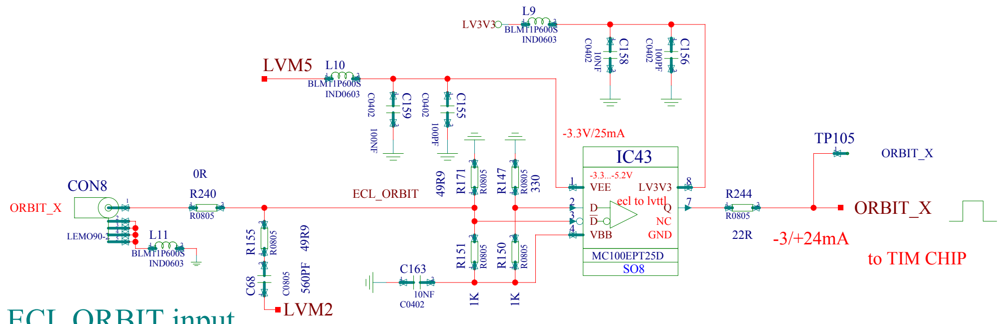






Ripple noise 150mV with: 2.2 and 22 uF
Ripple noise 50mV with: 4.7 and 47 uF

Max. I_term: (2-.8)V/50=24mA ==> 72mA for LVM2
Max. IEE/chip= 25mA==> 75 mA for LVM5



ECL CLK : fine time adj.: length of LemoCable
ECL CLK : coarse time adj.: select 0/90/180/270 phase in TIM chip

ECL ORBIT input

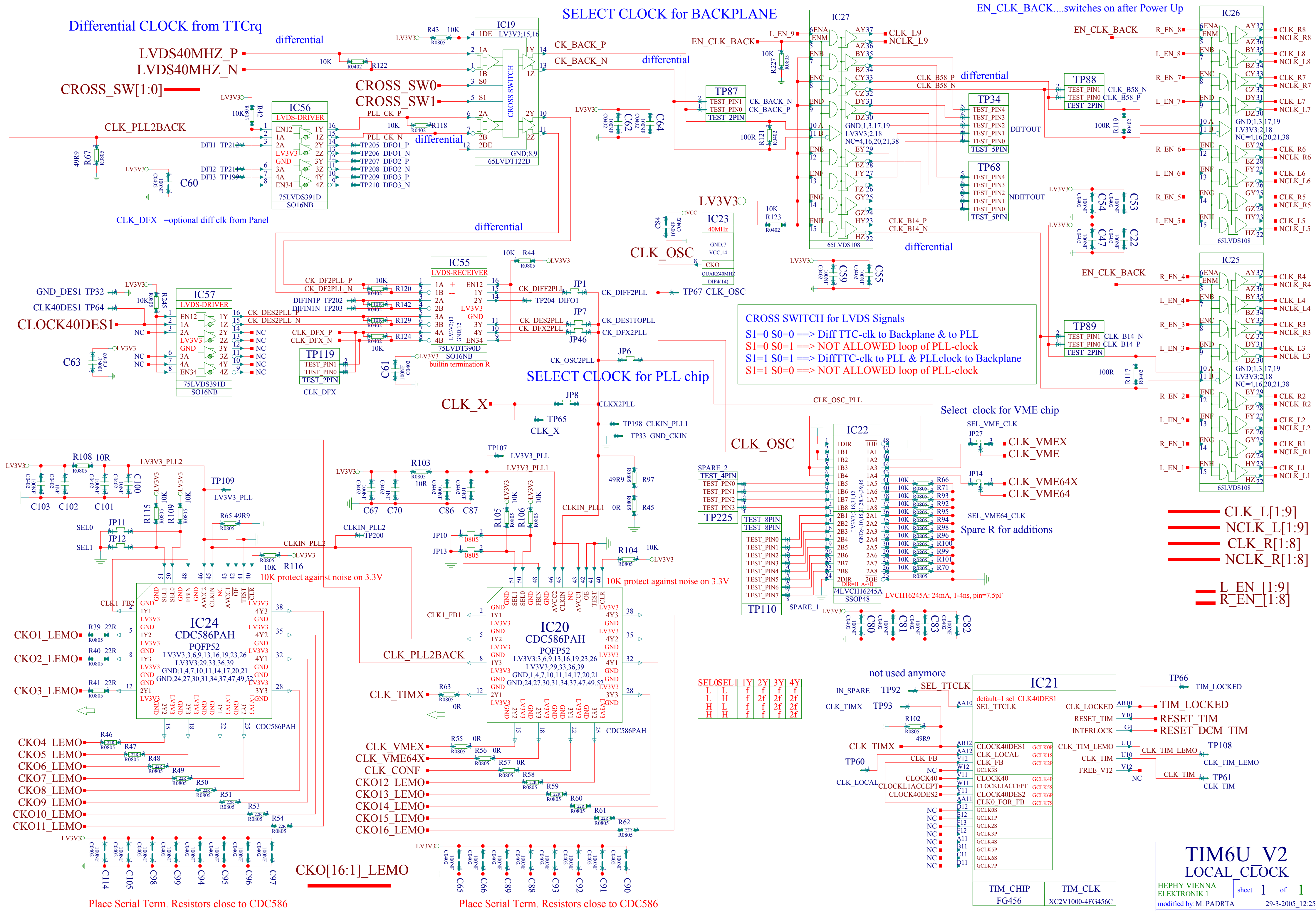
ECL (-5.2V) to TTL: 4 gates, MC10H125FN plcc20, 46 Stk a $3.24

ON-Semiconductor onsemi.com
MC100EPT21D: diff LVPECL to LVTTL, SOIC8 98Stk a $5.8
MC100EPT23: 3.3V 2x diff PECL to 24mA LVTTL, SOIC8 98Stk a $5.8
MC100EPT25: diff ECL/LVECL to LVTTL
MC100LVELT23D 2x diff 3.3VPECL to LVTTL24mA, SOIC8 98Stk a $4.44

0 mm available on 6U front panel (1 mounting block in middl

MICREL.com
SY10ELT21: diff PECL to TTL (Micrel)
SY10ELT21L: diff 3.3V PECL to TTL (Micrel)

TIM6U_V2	
FRONT_IO	
HEPHY VIENNA ELEKTRONIK 1	sheet 2 of 2
modified by: M. PADRTA	15-3-2005_11:52
checked by: HB	10-3-2005



CROSS SWITCH for LVDS Signals

S1=0 S0=0 ==> Diff TTC-clk to Backplane & to PLL

S1=0 S0=1 ==> NOT ALLOWED loop of PLL-clock

S1=1 S0=1 ==> DiffTTC-clk to PLL & PLLclock to Backplane

S1=1 S0=0 ==> NOT ALLOWED loop of PLL-clock

SEL0	SEL1	1Y	2Y	3Y	4Y
L	L	f	f	f	f
L	H	f	2f	2f	2f
H	L	f	f	2f	2f
H	H	f	f	f	2f

Place Serial Term. Resistors close to CDC586

Place Serial Term. Resistors close to CDC586

Virtex2: unused pins can be left open

TIM6U_V2

LOCAL_CLOCK

HEPHY VIENNA ELEKTRONIK 1 sheet 1 of 1

modified by: M. PADRTA 29-3-2005_12:25

checked by: HB 10-3-2006

CLOCK PHASE inside FPGAs can be shifted:
ADJUST time for data io inside FPGAs.
Xilinx Spartan2: CLK0,90,180,270 phase, 2.f, ...f/16
XilinxVirtexE: CLK90,180,270 for DLL 90MHz, CLK180 for DLLHF 180MHz
XilinxVirtex2: CLK90,180,270 for DLL 180MHz, CLK180 for DLLHF 360MHz
XilinxVirtex2: Phase Shift fix or online, step=1/256, -255..0..+255 steps

Altera APEX: CLK90,180,270 phase and delay shift available
Altera APEX -X suffix: CLK multiplication1x2x4x, PLL
Altera APEX2: like APEX
Altera ACEX: PLL, frequ. 1x,2x, clk only to reg's

Place both chips below close to each other.

The Virtex chip sends all signals with 50 serial termination.

L9 to SLOT 5: L1AOUT board

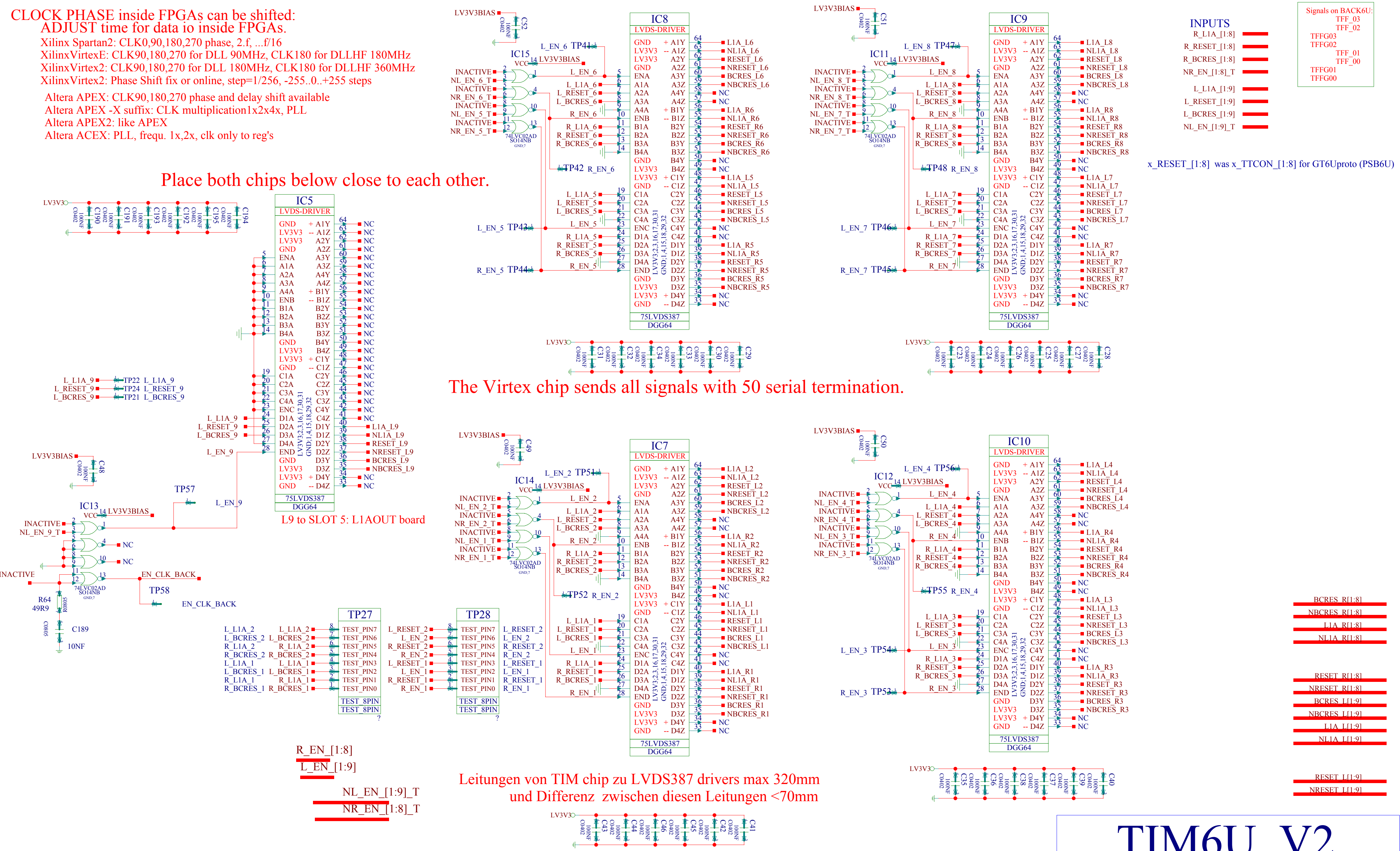
x_RESET_[1:8] was x_TTCON_[1:8] for GT6Uproto (PSB6U)

Leitungen von TIM chip zu LVDS387 drivers max 320mm
und Differenz zwischen diesen Leitungen <70mm

CLOCK signals from 1 chip to get min. skew.

LVDS387: dly= ris and fall 0.9-1.6-2.9 // edges skew dt=500ps
LVDS387: skew chan-chan 0.15ns, part-part 1.5ns
LV047A: dly=fall 0.5-0.9-1.7/ris0.5-1.2-1.7 // edges skew dt=300ps
LV047A: skew chan-chan 0.5ns, part-part 1.0ns
LV110T: dly= ris and fall 2.2-2.8-3.6 // edges skew dt=20-340ps//channskew<91ps

COMPENSATE DIFFERENT NET LENGTHS ON BACKPLANE
by net lengths on TIM board
TIM signals should arrive at same time on all boards.
TIM at Slot 15 in GT-6U proto backplane



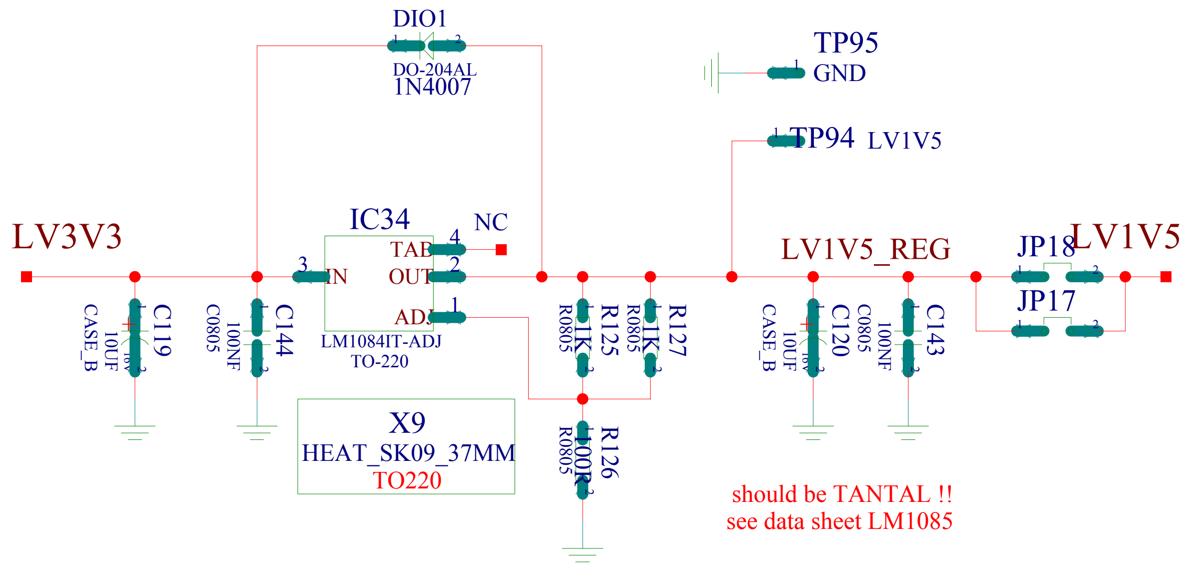
TIM6U_V2
LVDS_DRIVER

HEPHY VIENNA
ELEKTRONIK 1

sheet 1 of 1

modified by: A.TAUROK 8-4-2005_9:57

checked by: HB 10-3-2005



TIM6U_V2

POWER_TIM

HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by: AT	4-4-2005_14:58
checked by: HB	10-3-2005

Alternative Driver for RO-bus

LIVE INSERTION

SN74ABTE16245 HotSwap, incidentwave switching

SN74ABTE16245 A-side=bus ETL, B-side=25ohm

LVCH16245A: OUTPUTS not protected for live insertion [Vout<Vcc(instant.)+0.5V]

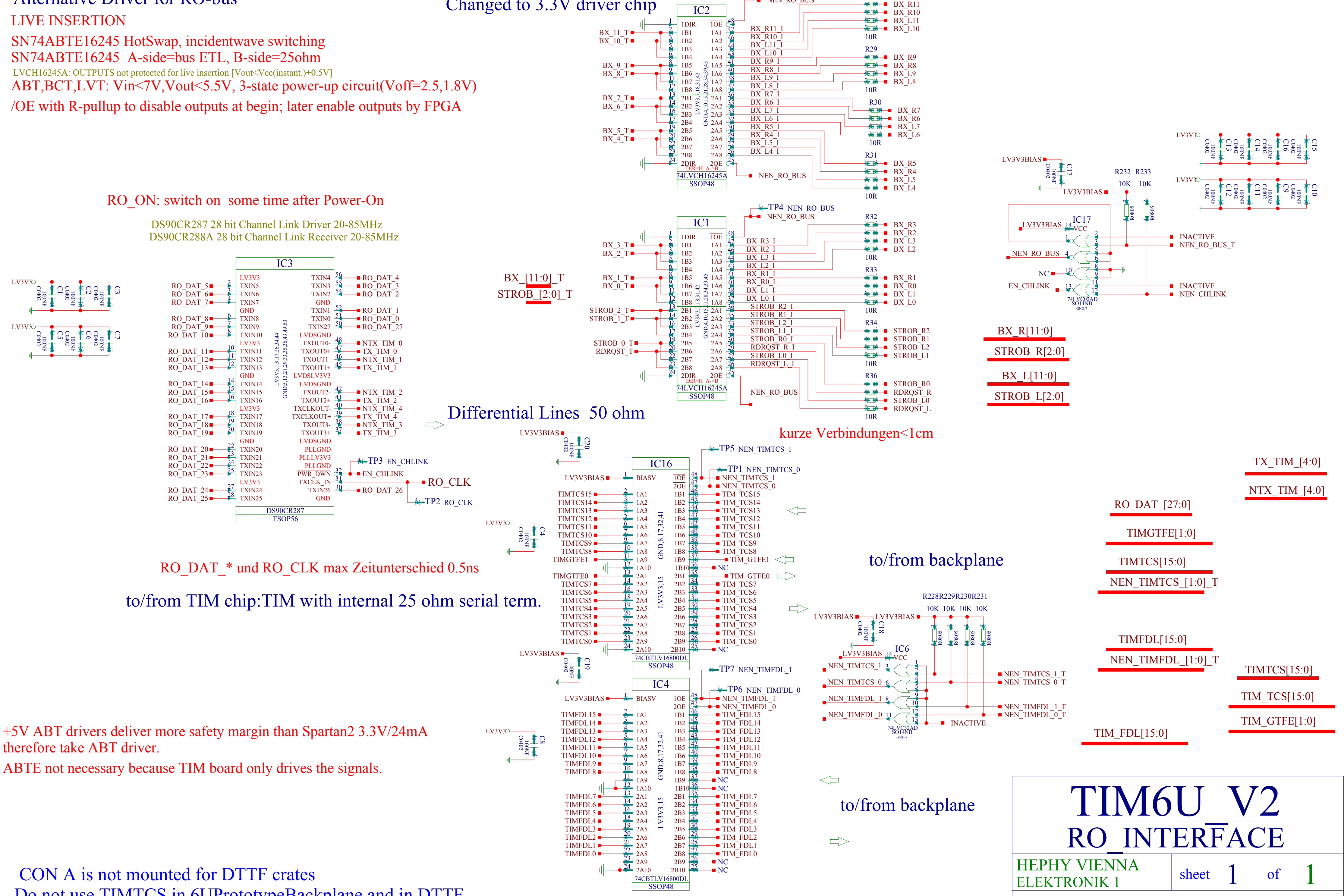
ABT,BCT,LVT: Vin<7V,Vout<5.5V, 3-state power-up circuit(Voff=2.5,1.8V)

/OE with R-pullup to disable outputs at begin; later enable outputs by FPGA

Changed to 3.3V driver chip

RO_ON: switch on some time after Power-On

DS90CR287 28 bit Channel Link Driver 20-85MHz
DS90CR288A 28 bit Channel Link Receiver 20-85MHz



Differential Lines 50 ohm

kurze Verbindungen<1cm

RO_DAT_* und RO_CLK max Zeitunterschied 0.5ns

to/from TIM chip:TIM with internal 25 ohm serial term.

to/from backplane

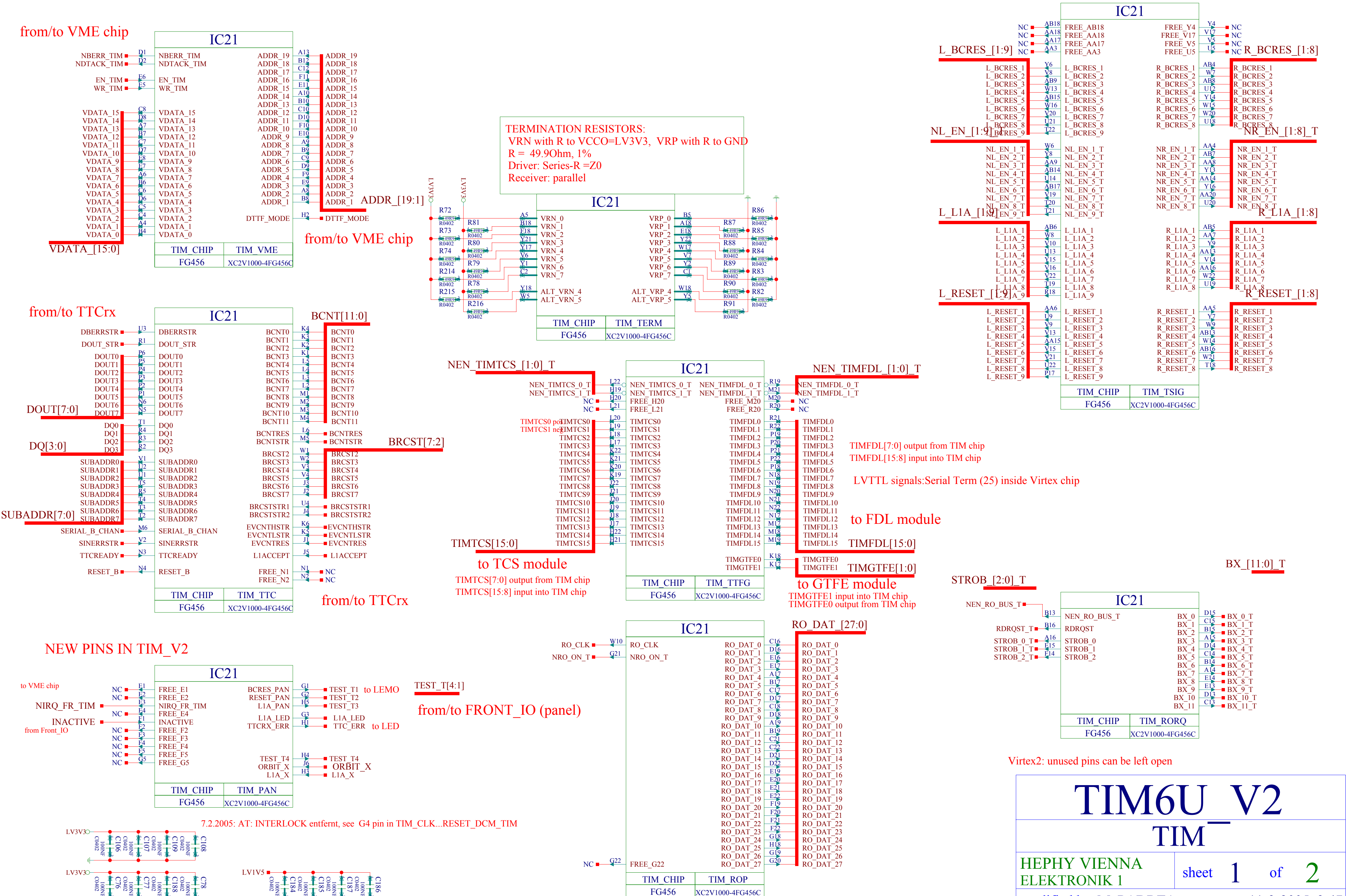
to/from backplane

+5V ABT drivers deliver more safety margin than Spartan2 3.3V/24mA
therefore take ABT driver.

ABTE not necessary because TIM board only drives the signals.

CON A is not mounted for DTTF crates
Do not use TIMTCS in 6UPrototypeBackplane and in DTTF

TIM6U_V2	
RO_INTERFACE	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by: A.T	10-3-2005_9:19
checked by: HB	10-3-2005



GND;M9,M10,M11,M12,M13,M14,N9,N10,N11,N12,N13,N14,P9,P10,P11,P12,P13,P14,W4,W19,Y3,Y20,AA2,AA21,AB1,AB22
GND;A1,A22,B2,B21,C3,C20,D4,D19,J9,J10,J11,J12,J13,J14,K9,K10,K11,K12,K13,K14,L9,L10,L11,L12,L13,L14
LV1V5;F6,F17,G7,G8,G15,G16,H7,H16,R7,R16,T7,T8,T15,T16,U6,U17
LV3V3;A12,B1,B22,F7,F8,F15,F16,G6,G9,G10,G11,G12,G13,G14,G17,H6,H17,J7,J16,K7,K16,L7,L16,M7,M16,M22,N7,N16,P7,P16,R6,R17
LV3V3;L7,T6,T9,T10,T11,T12,T13,T14,T17,U7,U8,U15,U16,AA1,AA22,AB11

TIM6U_V2
TIM

HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 2
modified by: M. PADRTA	11-3-2005_9:47
checked by: HB	10-3-2005

I2C acces via VME and via external frontside connector
2 I2Cadresses   Find I2C bus definition!!!!

BROADCAST COMMANDS.
BCNTRES, EVCNTRES reset also internal TTCrx counters.
Broadcast message: BRCST[0] = BCNTRES   delayed by coarse dly[3:0], pulse=1bx
Broadcast message: BRCST[1] = EVCNTRES delayed by coarse dly[3:0], pulse=1bx
System brdcast message: BRCST[5:2]   delayed by coarse dly[3:0], synchronous to CLK40DES1,   =register
User broadcast message: BRCST[7:6]   delayed by coarse dly[7:4], synchronous to CLK40DES2 or 1, =register

INDIVIDUAL COMMANDS: 14 bit ID used
2 Fine Dly regs, coarse delay reg, control reg
INDIVIDUAL COMMANDS to all TTCrx: ID=0

INTERNAL COMMANDS: 14 bit ID used
ERDUMP: sends int.err.counters to DOUT[7:0], DQ=1..4, DoutStr
CRDUMP: sends int.regs to DOUT[7:0], DQ=5..a, DoutStr
RESET via TTC: afterwards send BCNTRES and EVCNTRES to synchronise TIM to TTCrx chip.

VME instructions in TIM chip:
Reset TTCrx chip

FG676:
Bank0: GCLK 4S,5P,6S,7P at c13,d13,e13,f13
Bank1: GCLK 0S,1P,2S,3P at f14,g14,h15,h14
Bank4: GCLK 0P,1S,2P,3S at ad14,ac14,ab14,aa14
Bank5: GCLK 4P,5S,6P,7S at y13,aa13,ab13,ac13

TIM_CHIP FUNCTIONS:

Decode individual TTC instructions (DOUT,SUBADDR,DQ,DOUTStr,) defined by us

Decode broadcast TTC instructions: defined by...see CalibrWorkingGroup
Send Reset over RO-rqst bus

Simulate TTC instructions periodically
INSTRUCTION-table 4kx 16bits (16bits= instruction)

Simulate LHC orbit: BCNTRES resets Local BunchCounter
Simulate L1A signals /MonRqsts periodically, aligned to BCNT, immediately by VME instr.
BCNT-table 4kx 4bits (4bits= L1A,MonEvents,MonStatistics, BCNTRES)
Address=BC_counter

Monitoring Readout Request logic:
insert Monitoring readout request

L1A Readout Request logic:
send L1A readout request
DEFAULT: L1A only with Ev-cnter to get min. dead time
Check if local EVcntr agrees with TTC-evnr.
Add local BCnr, because TTC-bcnr doeas not arrive in default mode.
BCNT[11:0] :bx-number of L1A, compare it to local BCntr (=not default, maybe in test mode)
..but I don't know which mode is running when L1A arrives!! see pg 24 of TTCrx_manual
If there is no L1A, BCNT[11:0]= depends from ControlReg[1:0]

BCNTRes: reset local BCNTR, send it to all boards, delay it by n-bx ??
BCNTRes makes the local GT-time
Check if local BCNT=3564 when BCNTRes arrives.

VME-logic part runs with internal or external(test) Clock
TIM-logic runs with internal or external(test) Clock or TTC (CLOCK40DES1)

CLOCK40DES2 can be used for special case.
CLOCK40 is connected but not used.

TIM monitoring:
store L1A's arrival times.
error by L1A overflow
warning by L1A overflow

TTCrx monitoring:
Write TTCrx register contents into registers/mem? (DOUT,DQ,DOUTStr)
Set ERR flags after DBErrStr, SINErrStr (err-counter?)

OUTPUT
CLK40, BCNTRES, TTCON, L1A

SPECIAL Virtex2 PINS:

see JTAG schematic

M0	io/ INIT_B	CCLK	TCK
M1	io/ DOUT	PROG_B	TDI
M2	io/ D0	DONE	TDO
			TMS

HSWAP_EN
PWRDWN_B
DXN
DXP
VBATT
RSVD

io/ VRN_x   x=bank_nr
io/ VRP_x   x=bank_nr
io/ VREF_x   x=bank_nr

VCCAUX   8pins =+3.3V
VCCINT xx pins =+1.5 V
VCCO ...xx pins per bank =3.3 V

io/ GCLK0,2,4,6S or P
io/ GCLK1,3,5,7P or S
io/ rdwr_b, cs_b, d7..0

M2	M1	M0	
0	0	0	MASTER SERIAL
1	1	1	SLAVE SERIAL
1	0	1	BOUNDARY SCAN

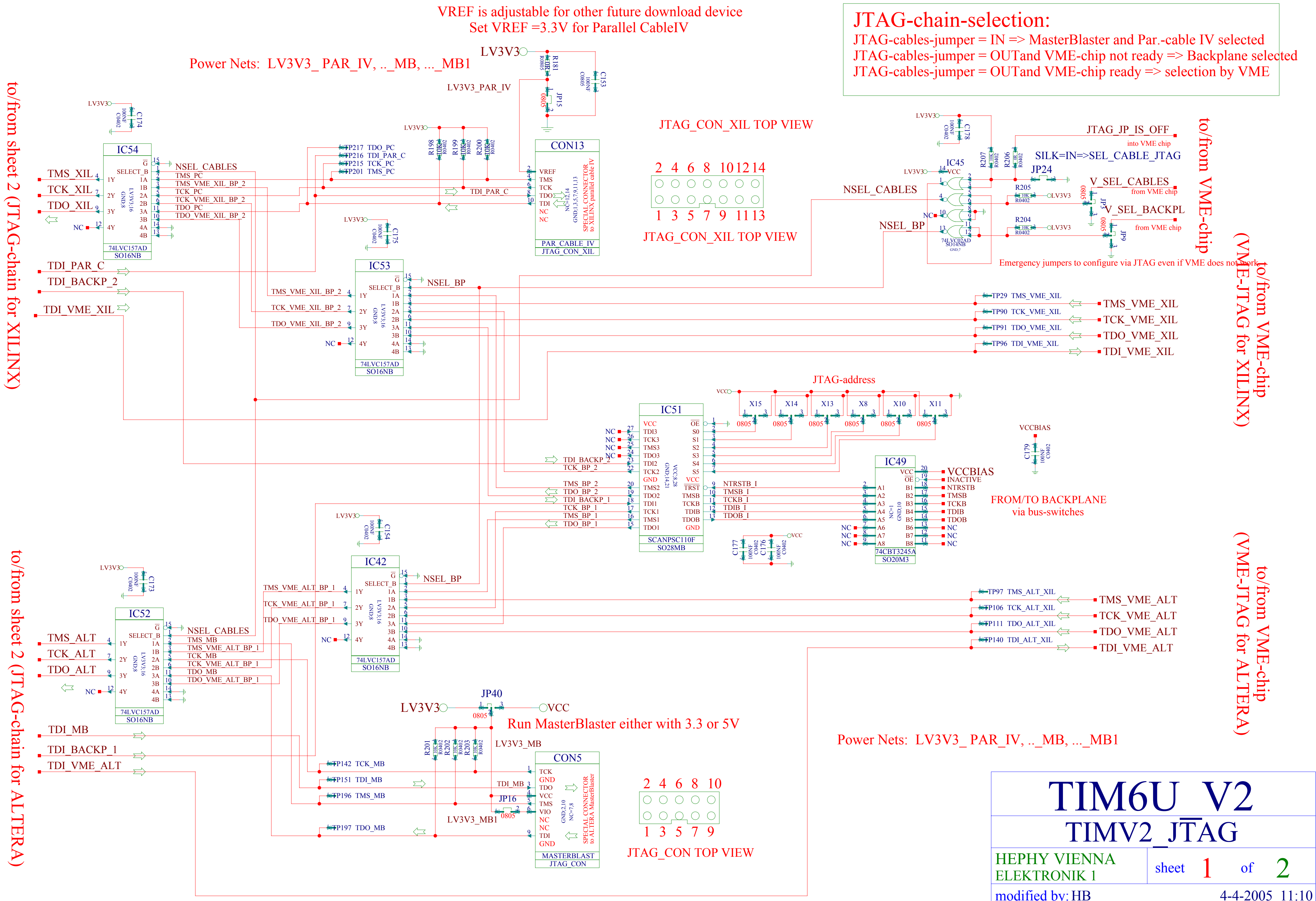
PACKAGES:
FG456: 1.00mm, 324 io 23x23 mm; XC2V250,500,1000
FG676: 1.00mm, 484 io 27x27 mm; XC2V1500,2000,3000
BG575: 1.27mm, 408 io 31x31 mm; XC2V1000,1500,2000
BG728: 1.27mm, 516 io 35x35 mm; XC2V2000,3000

Use of CLKL1A is unclear to me.
...disable it in TTCrx chip!

TIM6U-V2

TIM

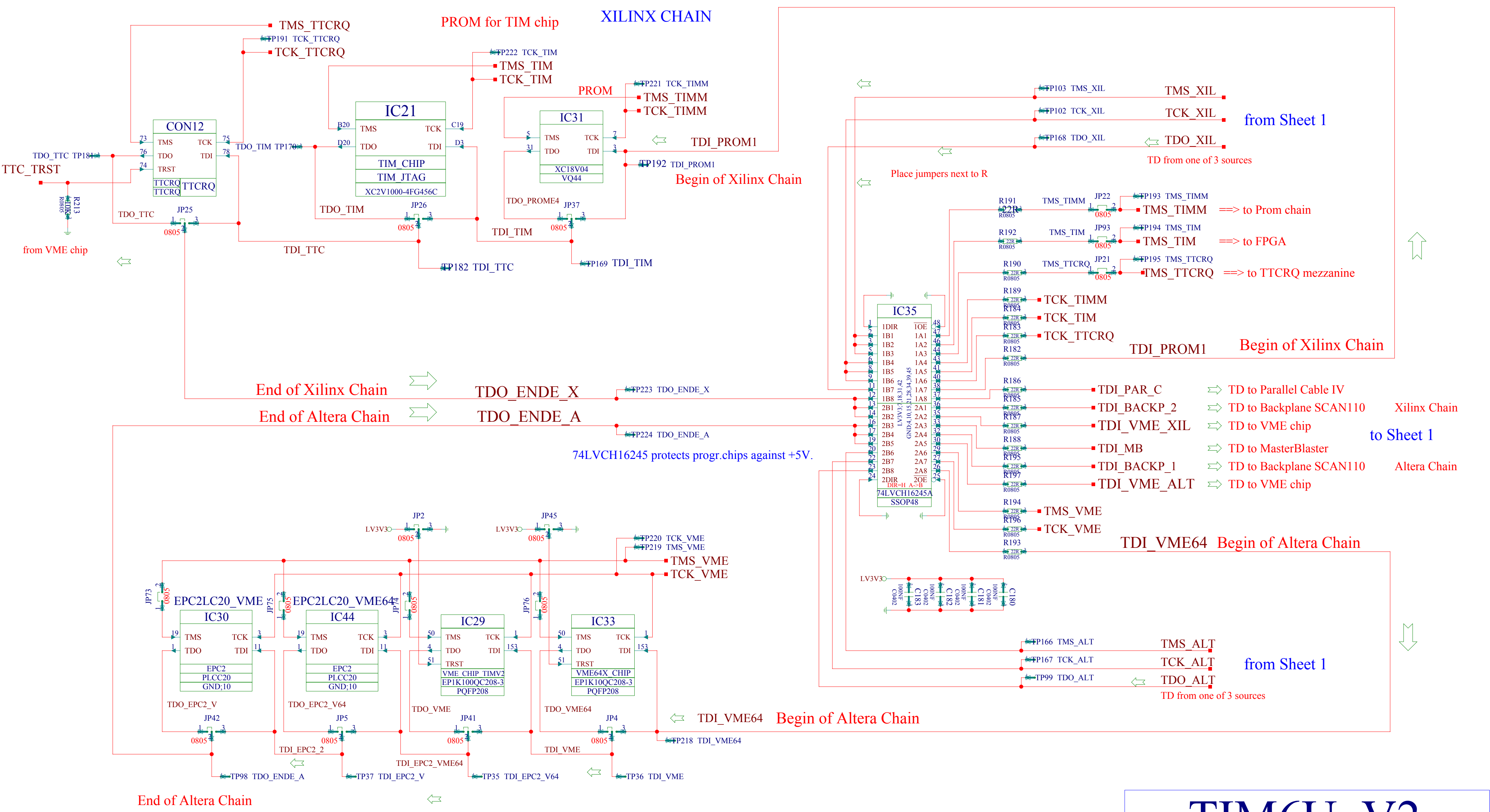
HEPHY VIENNA
ELEKTRONIK 1
sheet 2 of 2
modified by: H. BERGAUER   10-3-2005_9:01
checked by: HB   10-3-2005



TIM6U_V2

TIMV2_JTAG

HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 2
modified by: HB	4-4-2005_11:10
checked by: HB	10-3-2005

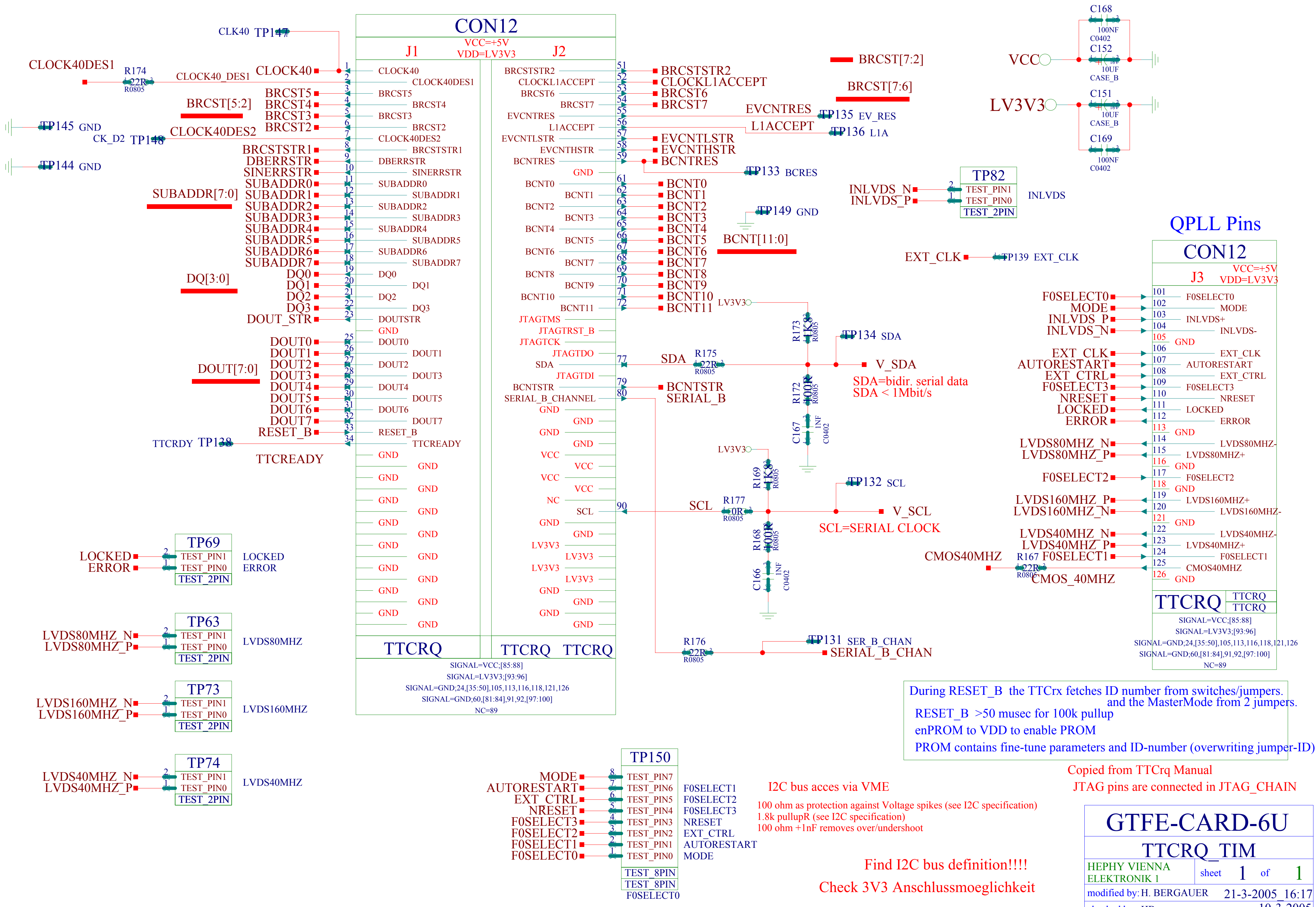


ALTERA CHAIN: Proms, EP1K10 = ACEX chips

bypass capacitors for proms are on page CONF_TIM

VME64X-chip: EP1K30QC208-3 verwenden!! AT+HB 230305

TIM6U_V2	
TIMV2_JTAG	
HEPHY VIENNA ELEKTRONIK 1	sheet 2 of 2
modified by: AT	23-3-2005_10:44
checked by: HB	10-3-2005



During RESET_B the TTCrx fetches ID number from switches/jumpers.
and the MasterMode from 2 jumpers.

RESET_B >50 musec for 100k pullup

enPROM to VDD to enable PROM

PROM contains fine-tune parameters and ID-number (overwriting jumper-ID)

Copied from TTCrq Manual

JTAG pins are connected in JTAG_CHAIN

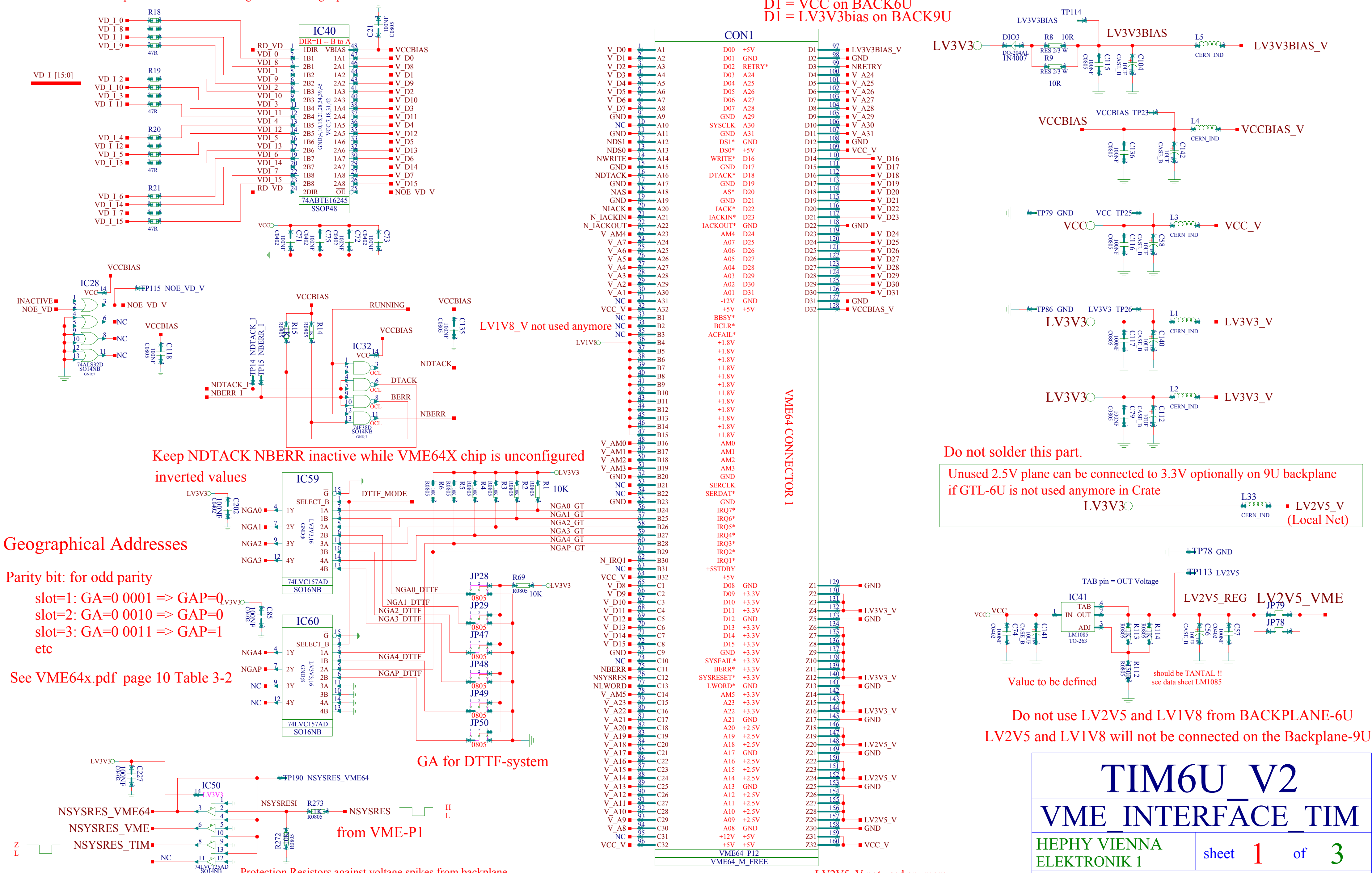
I2C bus acces via VME

100 ohm as protection against Voltage spikes (see I2C specification)
1.8k pullupR (see I2C specification)
100 ohm +1nF removes over/undershoot

Find I2C bus definition!!!!

Check 3V3 Anschlussmoeglichkeit

GTFE-CARD-6U		
TTCRQ_TIM		
HEPHY VIENNA ELEKTRONIK 1	sheet 1	of 1
modified by: H. BERGAUER	21-3-2005_16:17	
checked by: HB	10-3-2005	



47 Ohm resistors protect the Virtex drivers against overvoltage spikes.

PSB9U must not be inserted into the 6U Backplane!!!

D1 = VCC on BACK6U
D1 = LV3V3bias on BACK9U

LV1V8_V not used anymore

Keep NDTACK NBERR inactive while VME64X chip is unconfigured

inverted values

Do not solder this part.

Unused 2.5V plane can be connected to 3.3V optionally on 9U backplane
if GTL-6U is not used anymore in Crate

LV2V5_V (Local Net)

TAB pin = OUT Voltage

Value to be defined

should be TANTAL !!
see data sheet LM1085

Do not use LV2V5 and LV1V8 from BACKPLANE-6U
LV2V5 and LV1V8 will not be connected on the Backplane-9U

Geographical Addresses

Parity bit: for odd parity

slot=1: GA=0 0001 => GAP=0
slot=2: GA=0 0010 => GAP=0
slot=3: GA=0 0011 => GAP=1
etc

See VME64x.pdf page 10 Table 3-2

from VME-P1

Protection Resistors against voltage spikes from backplane.

NSYSRES reconfigures all Altera and Xilinx programmable chips.

Stecker und Signale mit GTL und TIM schematic vergleichen

LV2V5_V not used anymore

TIM6U_V2

VME_INTERFACE_TIM

HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 3
modified by: H. BERGAUER	8-4-2005_10:35
checked by: HB	10-3-2005

VME64X-chip: EP1K30QC208-3 verwenden!! AT+HB 230305

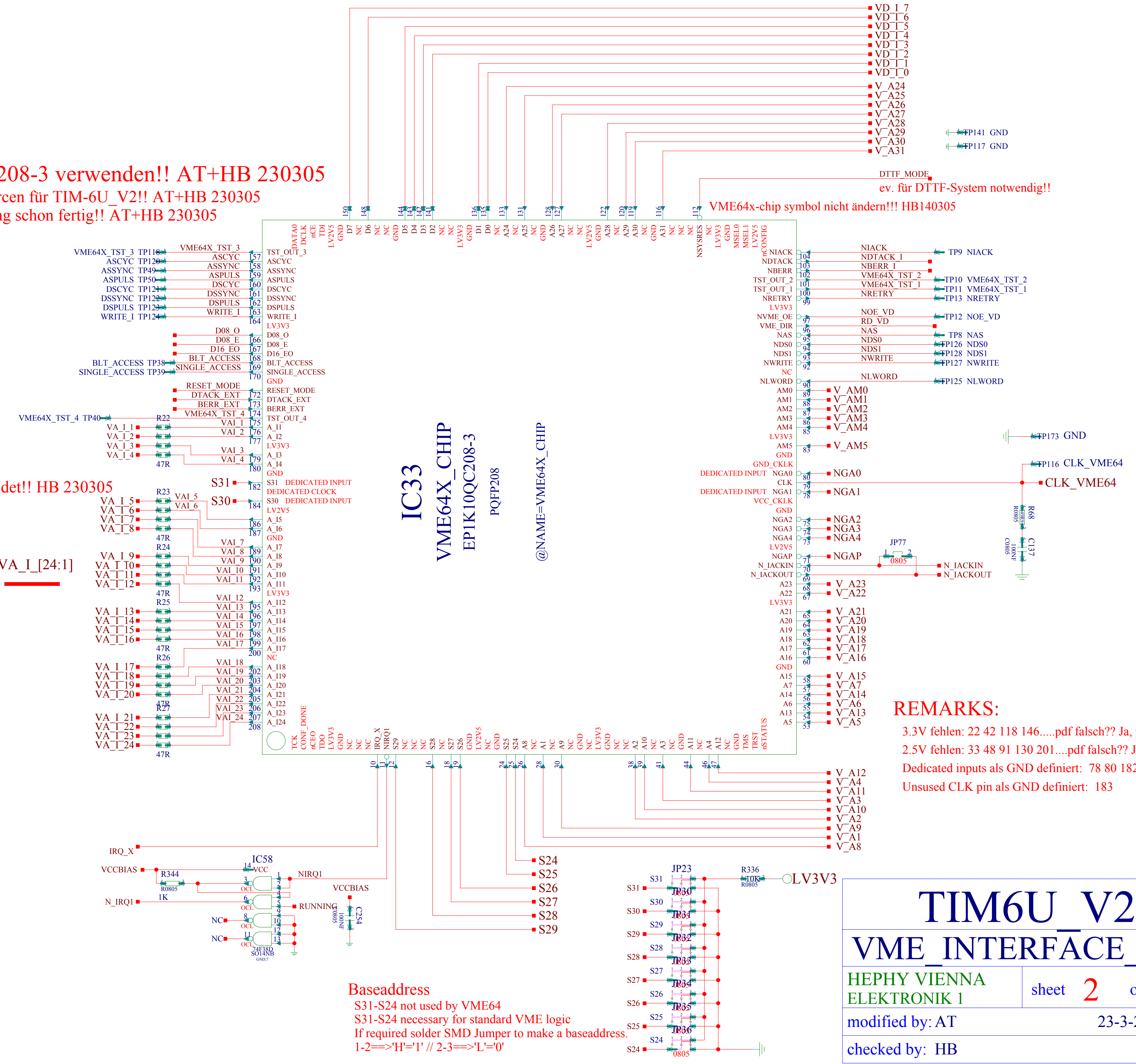
EP1K10QC208-3 hat zu wenig Ressourcen für TIM-6U_V2!! AT+HB 230305

Keine Änderung in schematic, da routing schon fertig!! AT+HB 230305

S31 und S30 auf GND, falls nicht verwendet!! HB 230305

ev. für DTTF-System notwendig!!

VME64x-chip symbol nicht ändern!!! HB140305



Baseaddress

S31-S24 not used by VME64

S31-S24 necessary for standard VME logic

If required solder SMD Jumper to make a baseaddress.

1-2==>'H'='1' // 2-3==>'L'='0'

REMARKS:

3.3V fehlen: 22 42 118 146.....pdf falsch?? Ja, pinfile ist Refer

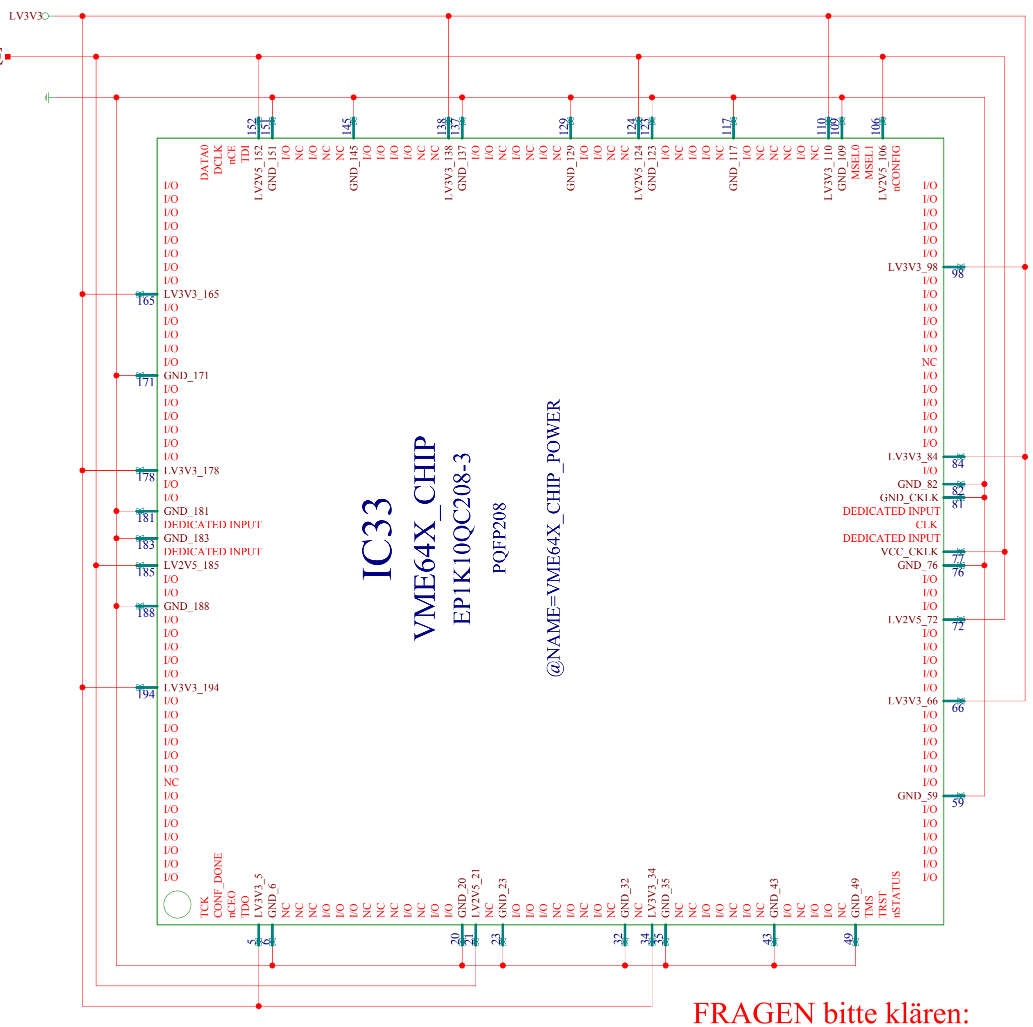
2.5V fehlen: 33 48 91 130 201....pdf falsch?? Ja, pinfile ist Ref

Dedicated inputs als GND definiert: 78 80 182 184

Unused CLK pin als GND definiert: 183

TIM6U_V2	
VME_INTERFACE_TIM	
HEPHY VIENNA ELEKTRONIK 1	sheet 2 of 3
modified by: AT	23-3-2005_11:02
checked by: HB	10-3-2005

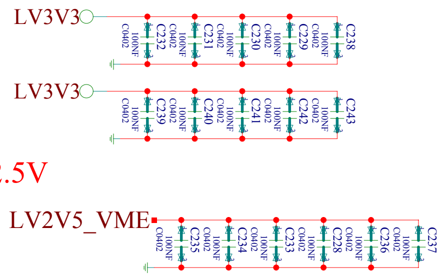
VME64X-chip: EP1K30QC208-3 verwenden!! AT+HB 230305

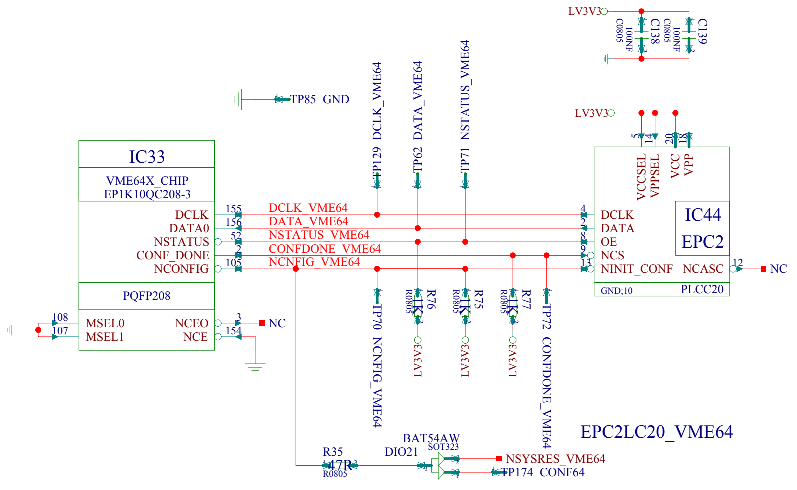


This pin is the power or ground for the ClockLock and ClockBoost circuitry of a PLL. To ensure noise resistance the power and ground supply to the ClockLock and ClockBoost circuitry should be isolated from the power and ground to the rest of the device.
If the PLL is not used, this power or ground pin should be connected to VCCINT or GNDINT, respectively.

10 pins for 3.3V

6 pins for 2.5V





See configdevices.pdf:
Do not insert 1K resistor when internal pullup R are used in IC20: EPC2

FRAGEN bitte klären:

3.3V fehlen: 22 42 118 146.....pdf falsch?? Ja, pinfile ist Referenz, HB 200803

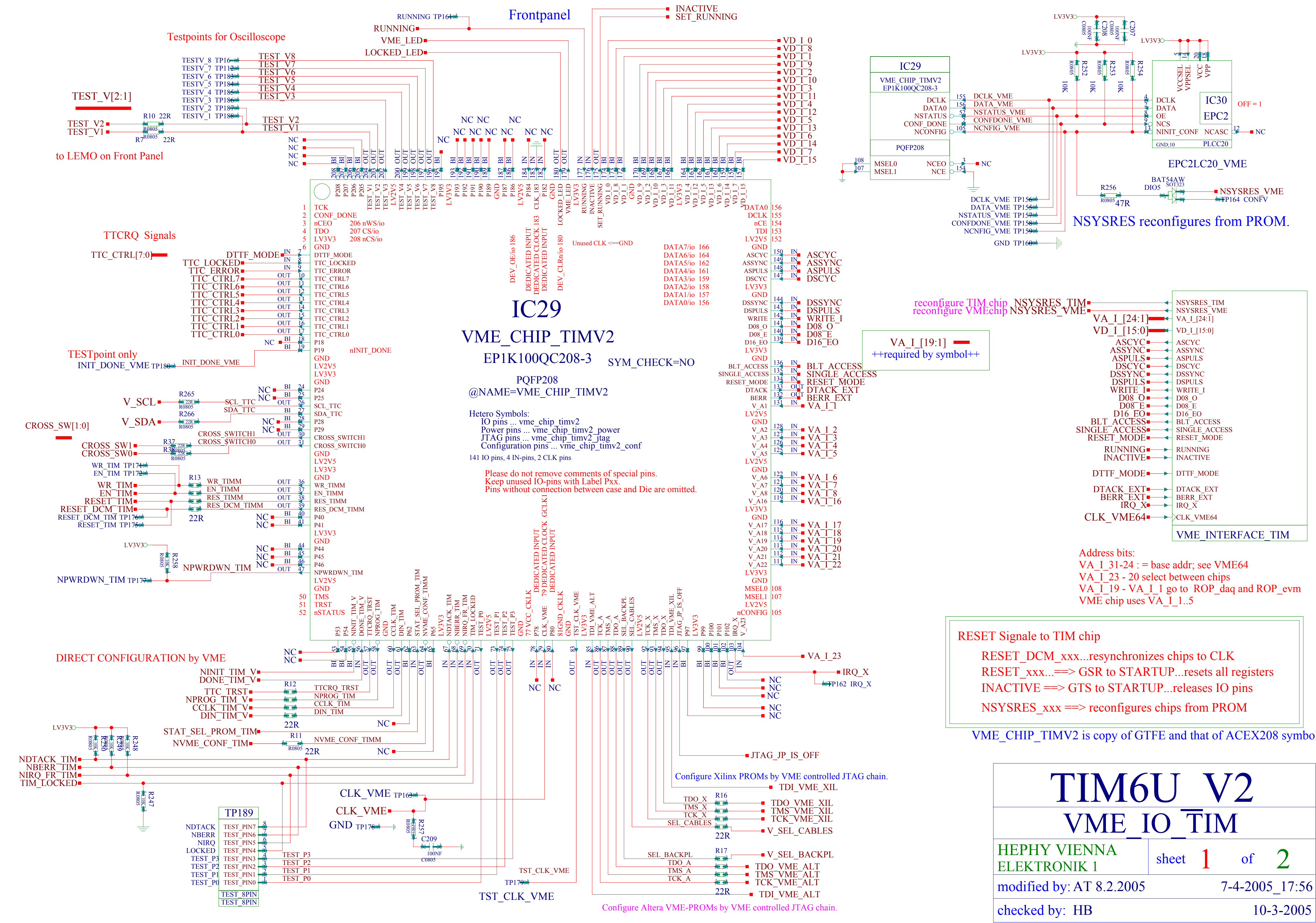
2.5V fehlen: 33 48 91 130 201....pdf falsch?? Ja, pinfile ist Referenz, HB 200803

Dedicated inputs als GND definiert: 78 80 182 184

Unused CLK pin als GND definiert: 183

INIT_DONE used as I/O: 19

TIM6U_V2	
VME_INTERFACE_TIM	
HEPHY VIENNA ELEKTRONIK 1	sheet 3 of 3
modified by: H. BERGAUER	23-3-2005_11:02
checked by: HB	10-3-2005


Frontpanel
Testpoints for Oscilloscope
TEST_V[2:1]
to LEMO on Front Panel
TTCRQ Signals
TESTpoint only
DIRECT CONFIGURATION by VME
IC29
VME_CHIP_TIMV2
EP1K100QC208-3
SYM_CHECK=NO
PQFP208
@NAME=VME_CHIP_TIMV2
Hetero Symbols:
IO pins ... vme_chip_timv2
Power pins ... vme_chip_timv2_power
JTAG pins ... vme_chip_timv2_jtag
Configuration pins ... vme_chip_timv2_conf
141 IO pins, 4 IN-pins, 2 CLK pins
Please do not remove comments of special pins.
Keep unused IO-pins with Label Pxx.
Pins without connection between case and Die are omitted.
Unused CLK <==GND
IC30
EPC2
OFF = 1
EPC2LC20_VME
NSYSRES reconfigures from PROM.
reconfigure TIM chip
reconfigure VMEchip
VA_I_[19:1]
++required by symbol++
VME_INTERFACE_TIM
Configure Xilinx PROMs by VME controlled JTAG chain.
Configure Altera VME-PROMs by VME controlled JTAG chain.
TP189
TST_CLK_VME
CLK_VME

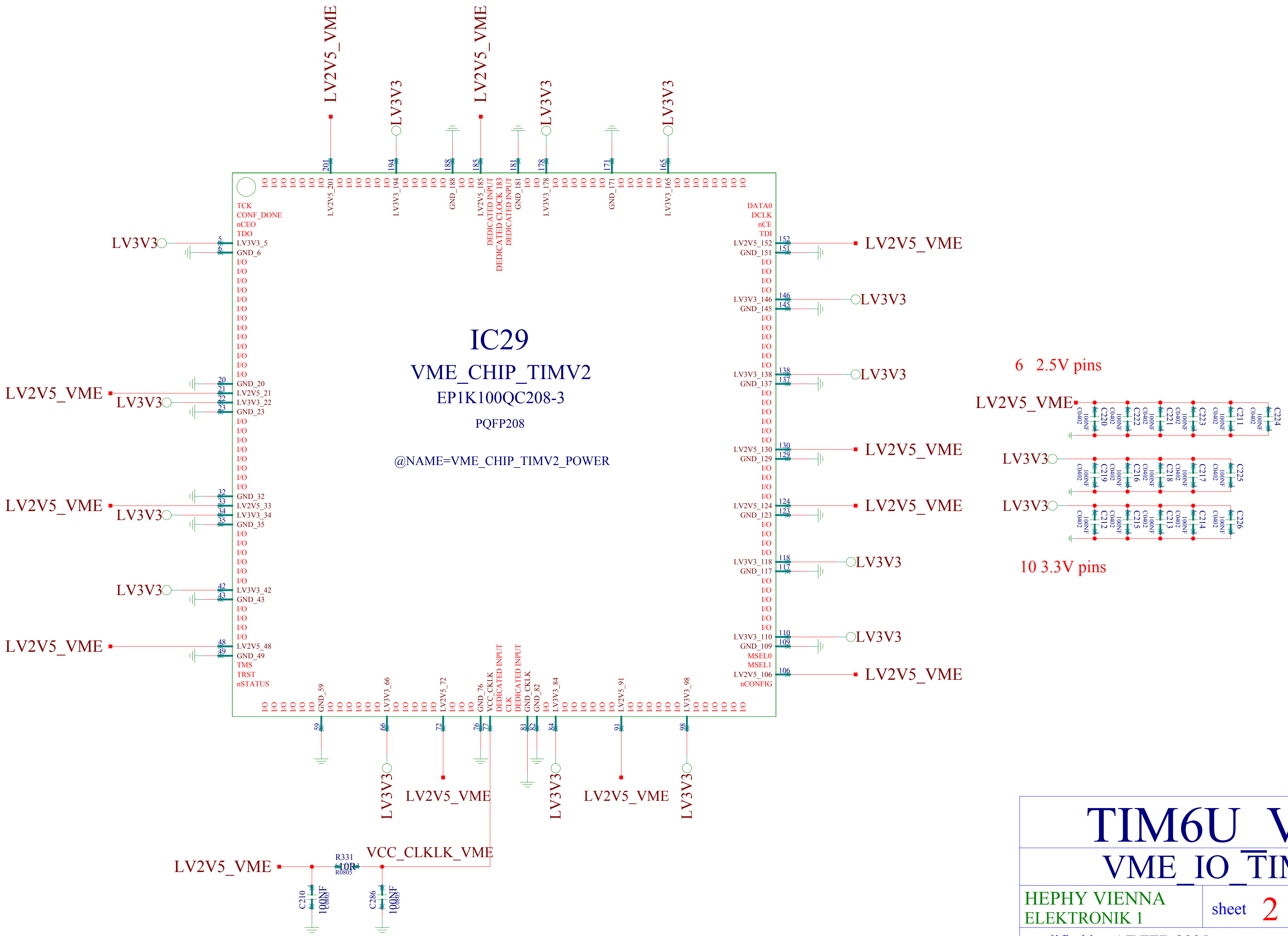
Address bits:
VA_I_31-24 : = base addr; see VME64
VA_I_23 - 20 select between chips
VA_I_19 - VA_I_1 go to ROP_daq and ROP_evm
VME chip uses VA_I_1..5

RESET Signale to TIM chip

RESET_DCM_xxx...resynchronizes chips to CLK
RESET_xxx...==> GSR to STARTUP...resets all registers
INACTIVE ==> GTS to STARTUP...releases IO pins
NSYSRES_xxx ==> reconfigures chips from PROM

VME_CHIP_TIMV2 is copy of GTFE and that of ACEX208 symbo

TIM6U_V2	
VME_IO_TIM	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 2
modified by: AT 8.2.2005	7-4-2005_17:56
checked by: HB	10-3-2005



TIM6U_V2

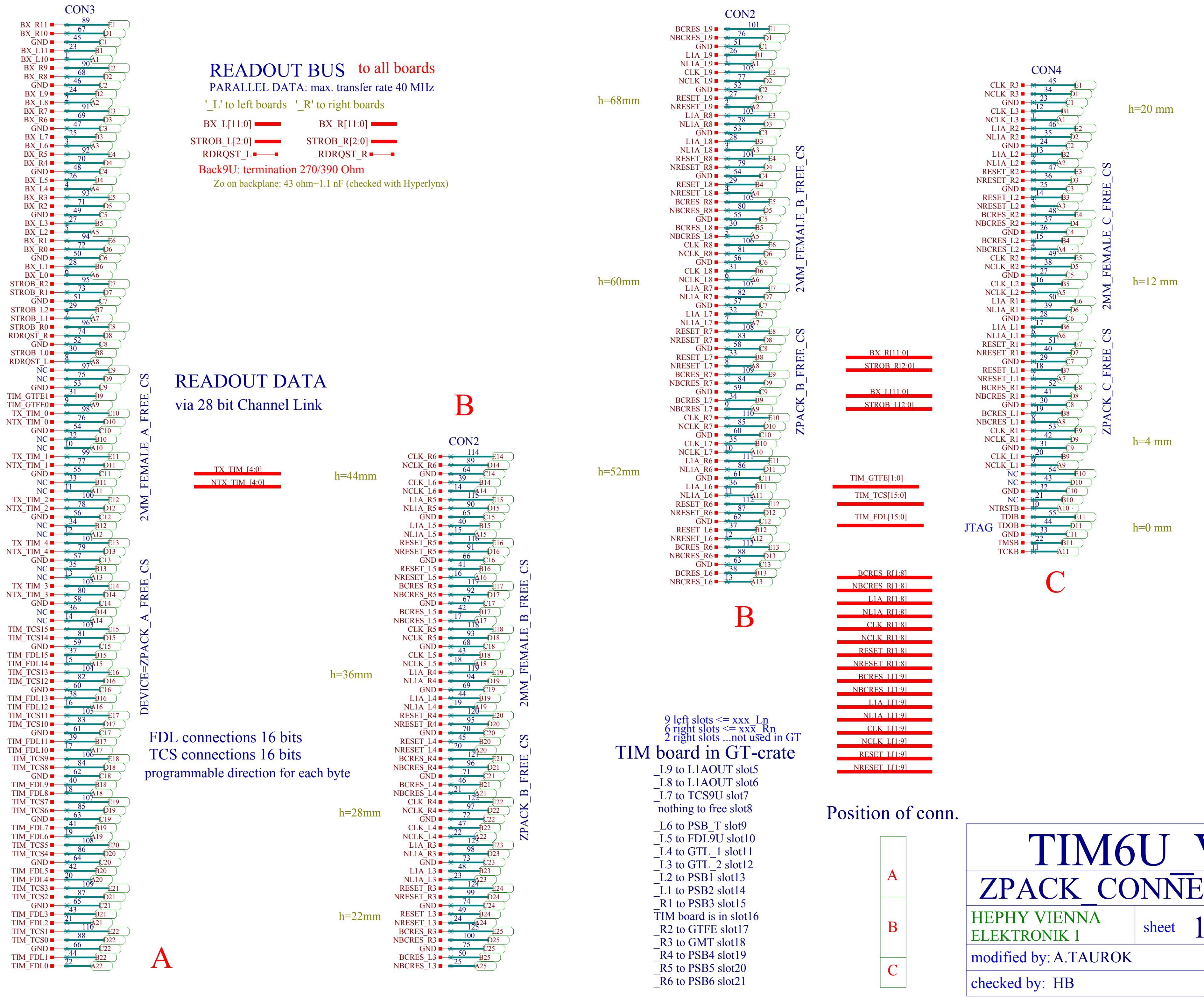
VME_IO_TIM

HEPHY VIENNA
ELEKTRONIK 1

sheet 2 of 2

modified by: AT FEB 2005 10-3-2005_9:20

checked by: HB 10-3-2005



9 left slots <= xxx_Ln
6 right slots <= xxx_Rn
2 right slots ...not used in GT

TIM board in GT-crate

_L9 to L1AOUT slot5
_L8 to L1AOUT slot6
_L7 to TCS9U slot7
nothing to free slot8
_L6 to PSB_T slot9
_L5 to FDL9U slot10
_L4 to GTL_1 slot11
_L3 to GTL_2 slot12
_L2 to PSB1 slot13
_L1 to PSB2 slot14
_R1 to PSB3 slot15
TIM board is in slot16
_R2 to GTFE slot17
_R3 to GMT slot18
_R4 to PSB4 slot19
_R5 to PSB5 slot20
_R6 to PSB6 slot21

TIM6U_V2

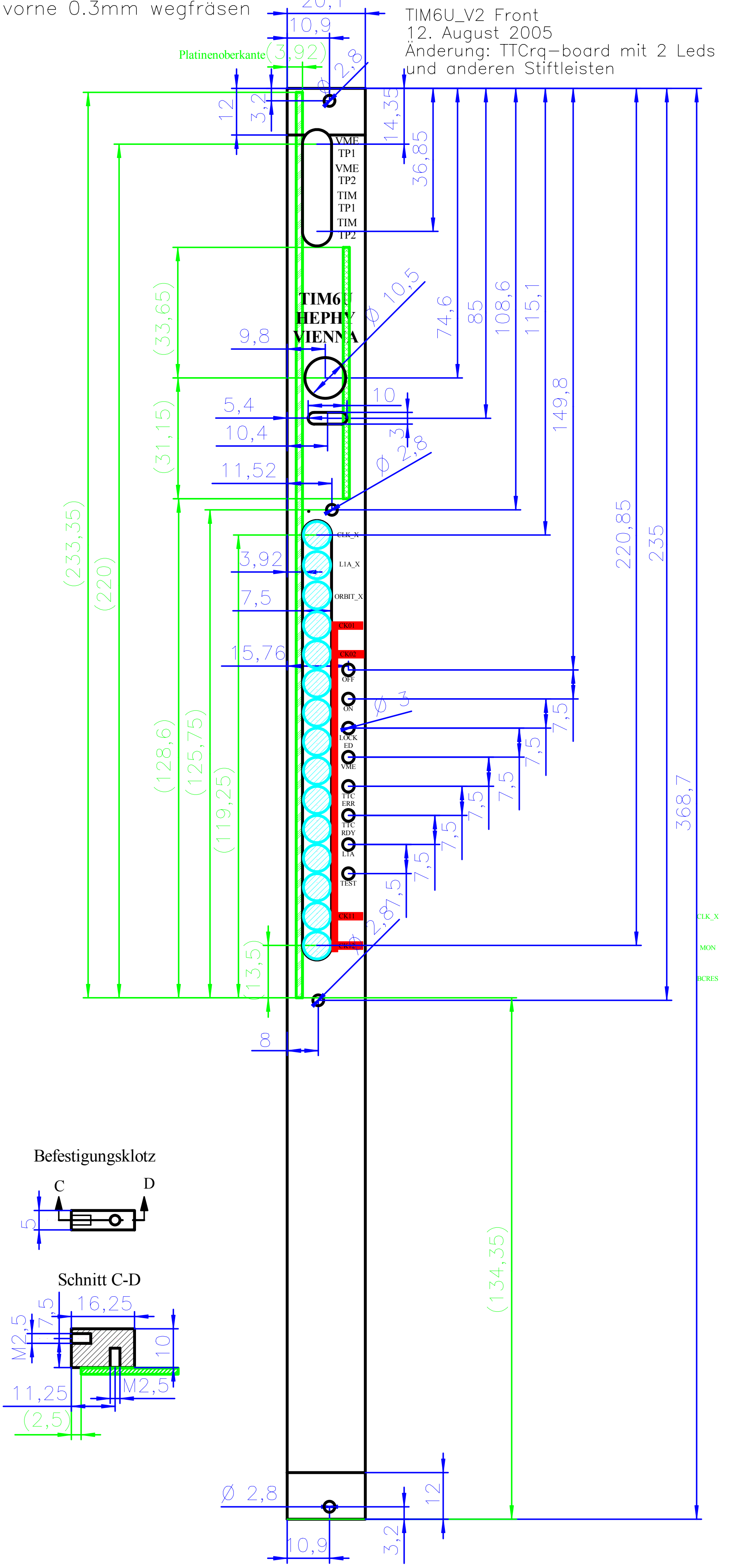
ZPACK_CONNECTION

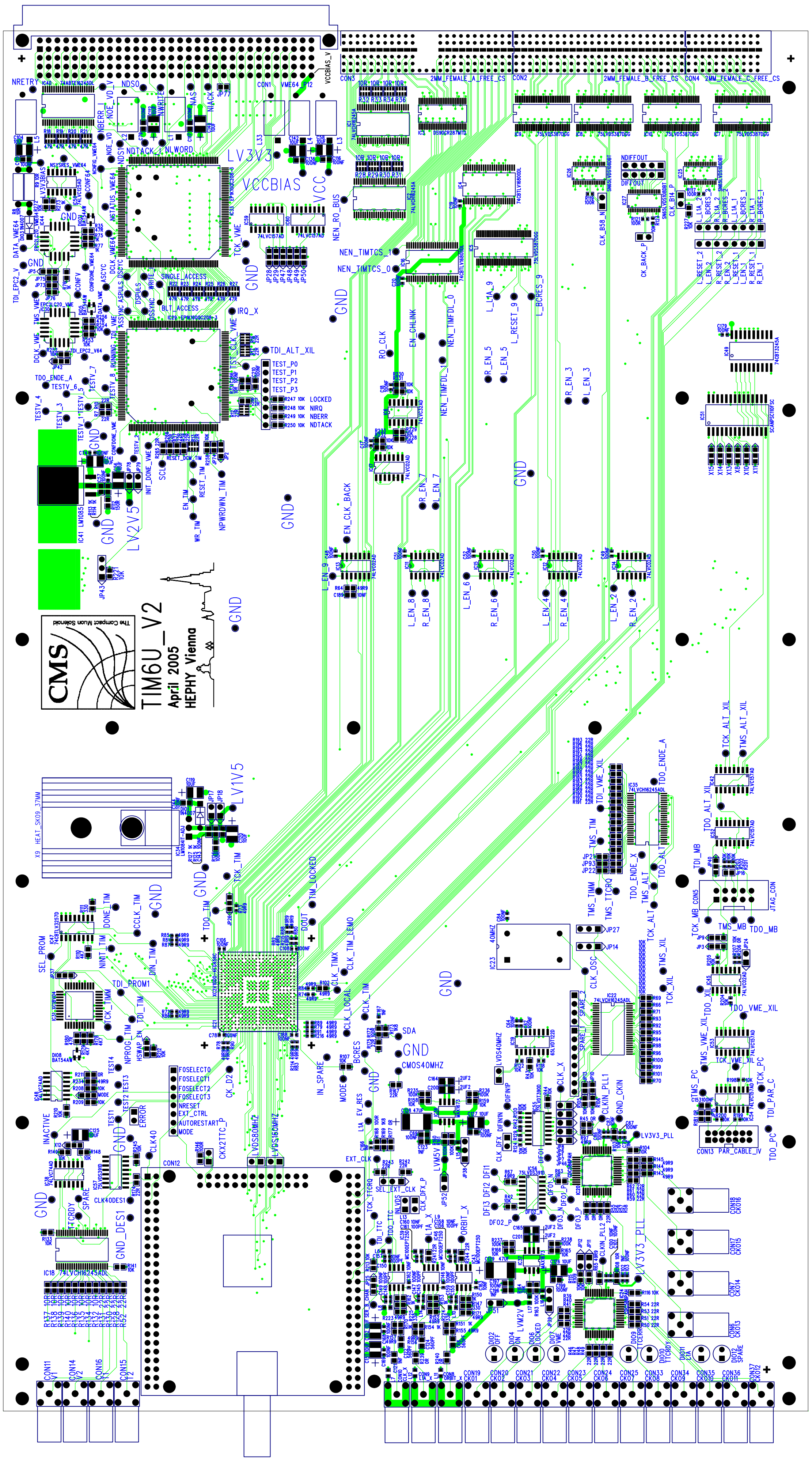
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by: A.TAUROK	10-3-2005_9:21
checked by: HB	10-3-2005

Material: Alu 3.0mm
oben und unten
vorne 0.3mm wegfräsen

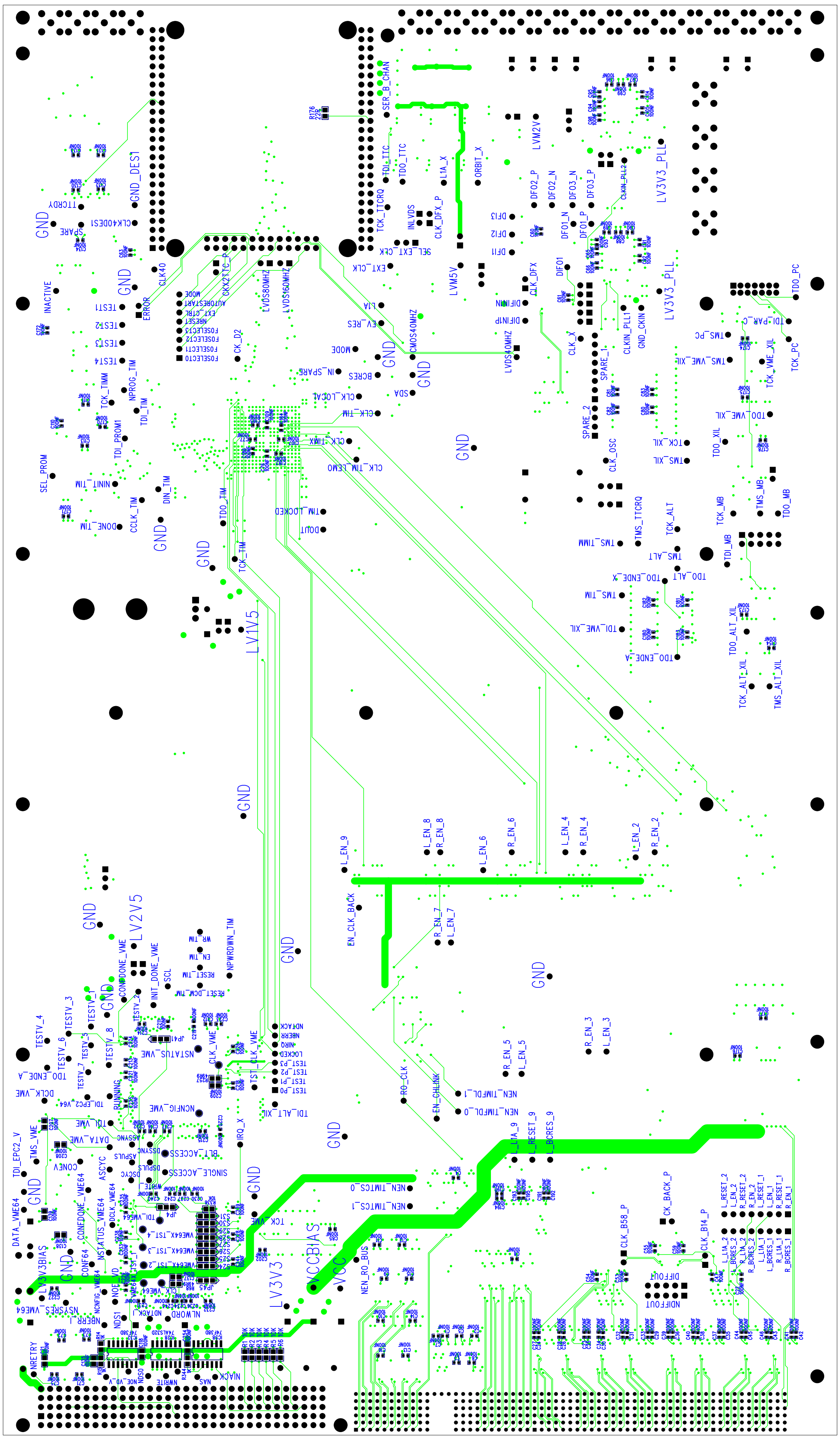
Datei:
TIM6U_V2_Front_Measures_120805.VLM

TIM6U_V2 Front
12. August 2005
Änderung: TTCrq–board mit 2 Leds
und anderen Stiftleisten





tim6u_v2_150405.pcb - Fri Apr 15 12:56:30 2005


GND
LV1V5
LV2V5
LV3V3
LV3V3_PLL
VCCBIAS
VCC
LVM5V
LVM2V
LV3V3BIAS

Jumper and switches on TIM6U_V2-card

R0805 with **0Ω:**
R45 (reserved termination R) **not** inserted.
R55, R56, R57, R63, R177, R181, R206, R211, R239 and **R240** inserted.

JP1, JP6, JP7 JP8 and JP46: selection of clock for PLL-chip (CLKIN_PLL1)
Only one jumper may be ON!
JP1 ON ➔ CK_DIFF2PLL selected.
JP6 ON ➔ clock from oscillator selected.
JP7 ON ➔ CK_DES2PLL selected.
JP8 ON ➔ external clock selected (CLK_X).
JP46 ON ➔ CK_DFX2PLL selected.

JP2 (SMD, top side)**:** TRST (JTAG) of VME-chip
1-2 (default)➔ solder R with 10K (LV3V3), TRST inactive.
2-3 ➔ do not solder.

JP3 (SMD, top side)**:** connect V_SEL_CABLES from VME
1-2 (default)➔ connected.
2-3 ➔ GND connection.

JP4 (SMD, bottom side)**:** VME64x-chip in JTAG-chain
1-2 ➔ VME64x-chip in JTAG-chain.
2-3 (default)➔ VME64x-chip **not** in JTAG-chain.

JP5 (SMD, top side)**:** PROM of VME64x-chip in JTAG-chain
1-2 (default)➔ PROM of VME64x-chip in JTAG-chain.
2-3 ➔ PROM of VME64x-chip **not** in JTAG-chain.

JP6 see at JP1

JP7 see at JP1

JP8 see at JP1

JP9 (SMD, top side)**:** connect V_SEL_BACKPL from VME
1-2 (default)➔ connected.
2-3 ➔ GND connection.

JP10 and JP13 (SMD, top side)**:** SEL-bits of IC20 (PLL)
(default)➔ JP10, JP13 = R with **0Ω**: 1 x CLKIN

JP11 and JP12 (top side)**:** SEL-bits of IC24 (PLL)
(default)➔ JP11, JP12 = **ON**: 1 x CLKIN

JP14 and JP27: selection of clock for VME-chips
1-2 (default)➔ interne clock.
2-3 ➔ externe clock.

JP15 (SMD, top side)**:** jumper for “VREF” of Parallel-Cable IV
(default)➔ not inserted.

JP16 (SMD, top side)**:** selection of VIO of masterblaster
OFF ➔ no voltage on VIO.
ON (default)➔ LV3V3 or VCC on VIO (see JP40).

JP17 and JP18: jumpers for LV1V5
(default)➔ solder-bridge after voltage-control.

JP19 (SMD, top side)**:** HSWAP_EN input of TIM-chip
1-2 (default)➔ HSWAP_EN=GND, enables pull-up-Rs of all I/O-pins in TIM-chip before configuration. In this position **configuration of TIM-chip via VME possible**.
2-3 ➔ HSWAP_EN=LV3V3, in this position **configuration of TIM-chip** via **VME not possible**.

JP20 [MODE] (SMD, top side)**:** NVME_CONF_TIM
(default)➔ nothing inserted.

JP21 (SMD, top side)**:** TMS_TTCRQ
OFF (default)➔ TTCRQ **not** in JTAG-chain.
ON ➔ TTCRQ in JTAG-chain.

JP22 (SMD, top side)**:** TMS_TIMM
OFF ➔ PROM of TIM-chip **not** in JTAG-chain.
ON (default)➔ PROM of TIM-chip in JTAG-chain.

JP23, JP30, JP31, JP32, JP33, JP34, JP35 and JP36 (SMD, bottom side)**:** S31-S24 for base address, **not used** in VME64x-systems!!!

JP24: jumper for “SEL_CABLE_JTAG”
OFF (default)➔ selection via VME.
ON ➔ MB and PC-IV selected for JTAG.

JP25 (SMD, top side)**:** TTCRQ in JTAG-chain
1-2 ➔ TTCRQ in JTAG-chain.
2-3 (default)➔ TTCRQ **not** in JTAG-chain.

JP26 (SMD, top side)**:** TIM-chip in JTAG-chain
1-2 ➔ TIM-chip in JTAG-chain.
2-3 (default)➔ TIM-chip **not** in JTAG-chain.

JP27 see at JP14

JP28, JP29, JP47, JP48, JP49 and JP50 (SMD, top side)**:** geographic addresses for DTTF, slot 12
JP50 ➔ 2-3: 0R inserted (NGAP_DTTF = GND)
JP49 ➔ 1-2: 0R inserted (NGA4_DTTF = open)
JP48 ➔ 2-3: 0R inserted (NGA3_DTTF = GND)
JP47 ➔ 2-3: 0R inserted (NGA2_DTTF = GND)
JP29 ➔ 1-2: 0R inserted (NGA1_DTTF = open)
JP28 ➔ 1-2: 0R inserted (NGA0_DTTF = open)

JP30 see at JP23

JP31 see at JP23

JP32 see at JP23

JP33 see at JP23

JP34 see at JP23

JP35 see at JP23

JP36 see at JP23

JP37 (SMD, top side)**:** PROM of TIM-chip in JTAG-chain
1-2 (default)➔ PROM of TIM-chip in JTAG-chain.
2-3 ➔ PROM of TIM-chip **not** in JTAG-chain.

JP38: voltage for LVM5-supply
1-2 (default)➔ VCC.
2-3 ➔ LV3V3.

JP39: voltage for LVM2-supply
1-2 (default)➔ VCC.
2-3 ➔ LV3V3.

JP40 (SMD, top side)**:** voltage selection for masterblaster
1-2 (default)➔ LV3V3.
2-3 ➔ VCC.

JP41 (SMD, bottom side)**:** VME-chip in JTAG-chain
1-2 ➔ VME-chip in JTAG-chain.
2-3 (default)➔ VME-chip **not** in JTAG-chain.

JP42 (SMD, top side)**:** PROM of VME-chip in JTAG-chain
1-2 (default)➔ PROM of VME-chip in JTAG-chain.
2-3 ➔ PROM of VME-chip **not** in JTAG-chain.

JP43: DTTF-mode
1-2 ➔ DTTF-mode.
2-3 ➔ GT-mode.

JP44 [SEL_EXT_CLK]: SEL_EXT_CLK
1-2 ➔ CLK_X, external clock to PLL-chip.
2-3 ➔ EXT_CLK2TTC, external clock to TTC_QPLL-chip.

JP45 (SMD, bottom side)**:** TRST (JTAG) of VME64x-chip
1-2 (default)➔ solder R with 10K (LV3V3), TRST inactive.
2-3 ➔ do not solder.

JP46 see at JP1

JP47 see at JP28

JP48 see at JP28

JP49 see at JP28

JP50 see at JP28

JP51 and JP52: jumpers for LVM2 and LVM5
(default)➔ solder-bridge after voltage-control.

JP53 - JP72 not in design

JP73, JP74, JP75 and JP76 (SMD, top sides)**:** jumper for "TMS-signals" for PROMs and VME-chips. These jumpers are set in the same way as JP4, JP5, JP41 and JP42.
OFF ➔ PROM **not** in JTAG-chain.
ON (default)➔ PROM in JTAG-chain.
JP4: VME64x-chip
JP5: PROM of VME64x-chip
JP41: VME-chip
JP42: PROM of VME-chip
JP73 (default)➔ inserted
JP74 (default)➔ not inserted
JP75 (default)➔ inserted
JP76 (default)➔ not inserted

JP77 (SMD, top side)**:** N_IACKIN/N_IACKOUT
ON ➔ always on, no interrupt.

JP78 and JP79: jumper for "LV2V5_VME"
(default)➔ solder-bridges after voltage-testing.

JP93 (SMD, top side)**:** TMS_TIM
OFF (default)➔ TIM-chip **not** in JTAG-chain.
ON ➔ TIM-chip in JTAG-chain.

MODE see at JP20

SEL_EXT_CLK see at JP44

X1 - X7: markers!!!

X8, X10, X11, X13 - X15 (SMD, top side)**:** jumper for JTAG-code from backplane for SCANPSC110
(default)➔ not used now.

X9: drill-hole!!!

X12 (SMD, top side)**:** INACTIVE/RUNNING after power-up
1-2 ➔ INACTIVE after power-up.
2-3 (default)➔ RUNNING after power-up.

VME64X-CHIP
(Version 0x100F)

of
TIM-6U_V2-card
(6U-Version)

H. Bergauer, K. Kastner, M. Padrta, A. Taurok

Dez-05

Version 0x100F

1 Introduction

The VME64x Interface for Global Trigger boards is made for a slave module without interrupt capabilities. It works in systems with backplanes supplying VME64x standard as well as in systems with VME/VME64 backplanes. The Interface will contain a VME64x_chip, a “board_access_chip”, transceivers for VME-data, logic for “live-insertion”, DTACK*- and BERR*-drivers and a special VME64x connector (P1/J1).

2 VME64x_chip

2.1 Versionshistory

- V1008: **do not use**, designed for EP1K30QC208-3.
- V1009: **do not use**, Testversion for V100A.
- V100A: **do not use, version does not work.** (HB120705)
- V100B: based on V1007 of other VME64x-chips, but DTTF_MODE signal of board is routed to the NSYSRES input of the VME64x-chip (designed for EP1K10QC208-3).
  Function 0 for GT-system, function 1 for DTTF-system. (HB130705)
- V100C: based on V100B of other VME64x-chips, but jumper S27-S24 used for CARD_NR to have only one configuration file for all card numbers. CARD_NR[3:0] is send to VME-chip on the lines ASCYC, ASSYNC, ASPULS and D08_E which are not used in previous versions.
  Version V100C has an own VIEWDRAW working directory and uses VIEWDRAW-library from Lab3Lib\Altera\Lab3_altera\vme_chips_lib.
  Function 0 for GT-system, function 1 for DTTF-system.
  Do not use, DTTF_MODE error. (HB090905)
- V100D: based on V100C of VME64x-chip, but NSYRES (=DTTF_MODE on board) pin implemented in VIEWDRAW. (HB090905)
- V100E: based on V100D of VME64x-chip, but AM=0x2F is combined with BASE_ADDR_CR_CSR to generate NVME_OE. (HB061205)
- **V100F:** complete new fully synchronous design implemented. DTACK_EXT and BERR_EXT from VME-CHIP-PSB are used as negative active signals now (because at power-up configuration of VME64X-CHIP is faster than configuration of VME-CHIP and therefore wrong DTACK and BERR signals are generated after configuration, which causes LEDs=”on” of CAEN-controller). INIT_DONE-feedback on pin 19 (S26 and pin 18 (S27) is implemented to have no wrong DTACK and BERR signals during init-phase after configuration-phase. Card-number is on S31-S28 (CARD_NR[3..0]) now. AM=0x2F is combined with BASE_ADDR_CR_CSR to generate correct NVME_OE. Therefore geo_addr_v2_0 and vme_d16_v1_6 are used. (HB161205)

2.2 Hardware

The VME64x-chip is an Altera EP1K10QC208-3.

2.3 Firmware

serial-nr.:	`TIM_V2`	
chip_id:	`0x0001Bn11`	`(n = CARD_NR from jumpers S31 – S28)`
version:	`0x0000100F`	

2.4 References

See VME64-specification and VME64x-specification for definitions.

2.5 Features of the VME64x-chip (V100F)

- **User Configuration ROM** (USER_CR: address range 0x01003..0x0101F, size is 8 bytes) for **"chip identifier" and "version"** of VME64x-chip (see **Error! Reference source not found.**).
- **"Card number"** is part of **"chip identifier"** and is fix soldered by jumpers on the lines S31-S28 (CARD_NR[3..0]).
- **"Serial number"** is **TIM_V2**.
- **User Command Status Register** (USER_CSR: address range 0x05003..0x0502F, size is 12 bytes) for the **"TEST_OUT-selection-registers"** is implemented (see USER_CSR space).
- **Function 0** (F0) for use in **GT**-system – **D16** only, base-address at **A31-A25**, AM=**0x0D** and **0x09** (only **single** transfer).
- **Function 1** (F1) for use in **DTTF**-system – **D16** only, base-address at **A23-A18**, AM=**0x3D** and **0x39** (only **single** transfer).

2.6 Address spaces overview

AM: **0x2F**, access: **D08_O**
A23-A19: Geographic address (=VME slot number) or '11110'=amnesia address

A18-A00	**=>**	**Register-name**
0x00003 - 0x007FF	=>	512x8 bit Configuration ROM (read)
0x01003	=>	chip-id_3 (read)
0x01007	=>	chip-id_2 (read)
0x0100B	=>	chip-id_1 (read)
0x0100F	=>	chip-id_0 (read)
0x01013	=>	version_3 (read)
0x01017	=>	version_2 (read)
0x0101B	=>	version_1 (read)
0x0101F	=>	version_0 (read)
0x01023 - 0x01037	=>	6 bytes Serial Number [TIM_V2] (read)
0x03003 - 0x037FF	=>	CRAM 512x8 bit RAM (not used!!) (read/write)
0x05003 - 0x05007	=>	TEST_OUT-selection in USER_CSR (read/write)
[0x7FC03 - 0x7FFFF	=>	Command/Status registers (read/write)]
0x7FF63	=>	ADER-F0_3 register (read/write)
0x7FF67	=>	ADER-F0_2 register (read/write)
0x7FF6B	=>	ADER-F0_1 register (read/write)
0x7FF6F	=>	ADER-F0_0 register (read/write)
0x7FF73	=>	ADER-F1_3 register (read/write)
0x7FF77	=>	ADER-F1_2 register (read/write)
0x7FF7B	=>	ADER-F1_1 register (read/write)
0x7FF7F	=>	ADER-F1_0 register (read/write)
0x7FFF7	=>	Bit Clear Register [BCR] (read/write)
0x7FFFB	=>	Bit Set Register [BSR] (read/write)
0x7FFFF	=>	BAR - Geographic address (read)

2.7 Parts of the VME64x-chip

2.7.1 Defined Configuration ROM (CR)

The definition of the CR is made in the VME64x-specification (10.2.1 The defined CR area, page 39 and Table 10-12, page 53).

- Checksum (0x03): see VME64-specification (Table 2-32, page 55)
  not calculated yet, to be done in cr.mif!!!
- Length of ROM (0x07..0x0F): see VME64-specification (Table 2-32, page 55)
  not calculated yet, to be done in cr.mif!!!
- Configuration ROM data access width (0x13): see VME64-specification (Table 2-32, page 55)
  0x81 => "Only use D08(O), every fourth byte".
- CSR data access width (0x17): see VME64-specification (Table 2-32, page 55)
  0x81 => "Only use D08(O), every fourth byte".
- CR/CSR space specification ID (0x1B): see VME64x-specification (Rule 10.3, page 39)
  0x02 => VME64x.
- Manufactor's ID (0x27..0x2F): see VME64-specification (Table 2-32, page 56)
  0x00.
- Board ID (0x33..0x3F): see VME64-specification (Table 2-32, page 56)
  not fixed yet, has to be defined for all boards of the GT-system!!
- Revision ID (0x43..0x4F): see VME64-specification (Table 2-32, page 56)
  not fixed yet, has to be defined for all boards of the GT-system!!
- Program ID (0x7F): see VME64-specification (Table 2-32, page 56)
  0x01 => "No program, ID ROM only".
- Offset to BEG_USER_CR (0x83..0x8B): see VME64x-specification (Table 10-12, page 53)
  0x01003 => used for chip_id- and version-register.
- Offset to END_USER_CR (0x8F..0x97): see VME64x-specification (Table 10-12, page 53)
  0x0101F => used for chip_id- and version-register.
- Offset to BEG_CRAM (0x9B..0xA3): see VME64x-specification (Table 10-12, page 53)
  0x03003 => used for future applications.
- Offset to END_CRAM (0xA7..0xAF): see VME64x-specification (Table 10-12, page 53)
  0x037FF => used for future applications.
- Offset to BEG_USER_CSR (0xB3..0xBB): see VME64x-specification (Table 10-12, page 53)
  0x05003 => used for TEST_OUT-selection register..
- Offset to END_USER_CSR (0xBF..0xC7): see VME64x-specification (Table 10-12, page 53)
  0x0502F.
- Offset to BEG_SN (0xCB..0xD3): see VME64x-specification (Table 10-12, page 53)
  0x01023 => part of USER_CR, contains the "serial number".
- Offset to END_SN (0xD7..0xDF): see VME64x-specification (Table 10-12, page 53)
  0x01033.
- Slave characteristics parameter (0xE3): see VME64x-specification (Table 10-1, page 40)
  0x00.
- Master characteristics parameter (0xEB): see VME64x-specification (Table 10-2, page 40)
  0x00.
- CRAM_ACCESS_WIDTH (0xFF): see VME64x-specification (Table 10-10, page 49)

0x81 => “Only use D08(O), every fourth byte“.

- Function 0 and 1 DAWPR (0x103..0x107): see VME64x-specification (Table 10-3, page 42)
  0x83 => “Accepts D16 or D08(EO) cycles“.
- Function 0 AMCAP (0x123..0x13F): see VME64x-specification (Table 10-5, page 44)
  0x0000 0000 0000 2200 => AM=0x0D and 0x09 „extended data access“ - single access.
- Function 1 AMCAP (0x143..0x15F): see VME64x-specification (Table 10-5, page 44)
  0x2200 0000 0000 0000 => AM=0x3D and 0x39 „standard data access“ - single access.
- Function 0 ADEM (0x623..0x62F): see VME64x-specification (Table 10-4, page 43)
  0xFE000000 => "mask bits 31-25=1" for GT-system.
- Function 1 ADEM (0x633..0x63F): see VME64x-specification (Table 10-4, page 43)
  0xFFFC0000 => "mask bits 31-18=1" for DTTF-system.

2.7.2 Defined Control/Status Register (CSR)

The definition of the CSR is made in the VME64x-specification (10.2.2 The defined CSR area, page 45 and Table 10-13, page 55).

- **Base Address Register** (BAR) (0x7FFFF): see VME64x-specification (Table 10-13, page 55), set with geographical address or amnesia address.
- **Bit Set Register** (BSR) (0x7FFFB): see VME64x-specification (Table 10-13, page 55), for settting see Table 10-6, page 45.
- **Bit Clear Register** (BCR) (0x7FFF7): see VME64x-specification (Table 10-13, page 55), for settting see Table 10-7, page 46.
  BCR, BSR bits:
    Bit 7: EN/DIS RESET_MODE
    Bit 4: EN/DIS MODULE
    Bit 3: EN/DIS BERR FLAG
- **Function 1 ADER** (0x7FF73..0x7FF7F): see VME64x-specification (Table 10-13, page 55), used for address relocation with Function 1 ADEM and Function 1 AMCAP (see Table 10-8, page 47).
- **Function 0 ADER** (0x7FF63..0x7FF6F): see VME64x-specification (Table 10-13, page 55), used for address relocation with Function 0 ADEM and Function 0 AMCAP (see Table 10-8, page 47).

2.7.3 Chip_ID and version ROM space

A user configuration ROM is implemented for the “chip_ID“ and “version” of the VME64x-chip of the board. It is located at the addresses 0x01003-0x0101F, size is 8 bytes, part of the USER_CR.

chip_id:	`0x0001B011`	`(CARD_NR comes from jumpers S31 – S28)`
version:	`0x0000100F`	

2.7.4 Serial Number ROM space

A user configuration ROM is implemented for the „Serial Number“ of the board. It is located at the addresses 0x01023-0x01037, size is 6 bytes, part of the USER_CR.

serial-nr.: `TIM_V2`

2.7.5 USER_CSR space

A “user command status register (USER_CSR)” is implemented for the „TEST_OUT-selection-registers”.

2.7.5.1 TEST_OUT registers

TEST_OUT-registers are used to select internal signals to a certain TEST_OUT-pin. There are four TEST_OUT-pins implemented, each pin can driven by 1 of 16 internal signals. So 4 bits are used for the code of the selection of internal signals. We have two registers, one for the selection of TST_OUT_1 and TST_OUT_2, the other for TST_OUT_3 and TST_OUT_4.

2.7.5.2 Test-signal-definition

Code for the selection of internal signals for TEST_OUT-pins:

sel_test_out_x[3:0]	➔	**testsignal-name**
0000	➔	TST_CLK_VME
0001	➔	D08_O_I
0010	➔	D16_EO_I
0011	➔	LD_CNT
0100	➔	CLT_CNT
0101	➔	CNT_EN
0110	➔	DTACK_CR_CSR
0111	➔	RD_CR
1000	➔	ASCYC_I
1001	➔	ASSYNC_I
1010	➔	ASPULS_I
1011	➔	DTACK_EXT_I
1100	➔	BERR_EXT_I
1101	➔	D32_EO_I
1110	➔	D08_E_I
1111	➔	MODULE_ENABLED

2.7.5.3 Registerdefinition

`0x05003` => **sel_test_out_12 (write/read)**

D7	D6	D5	D4	D3	D2	D1	D0
sel_test_out_2[3:0]				**sel_test_out_1[3:0]**			

`0x05007` => **sel_test_out_34 (write/read)**

D7	D6	D5	D4	D3	D2	D1	D0
sel_test_out_4[3:0]				**sel_test_out_3[3:0]**			

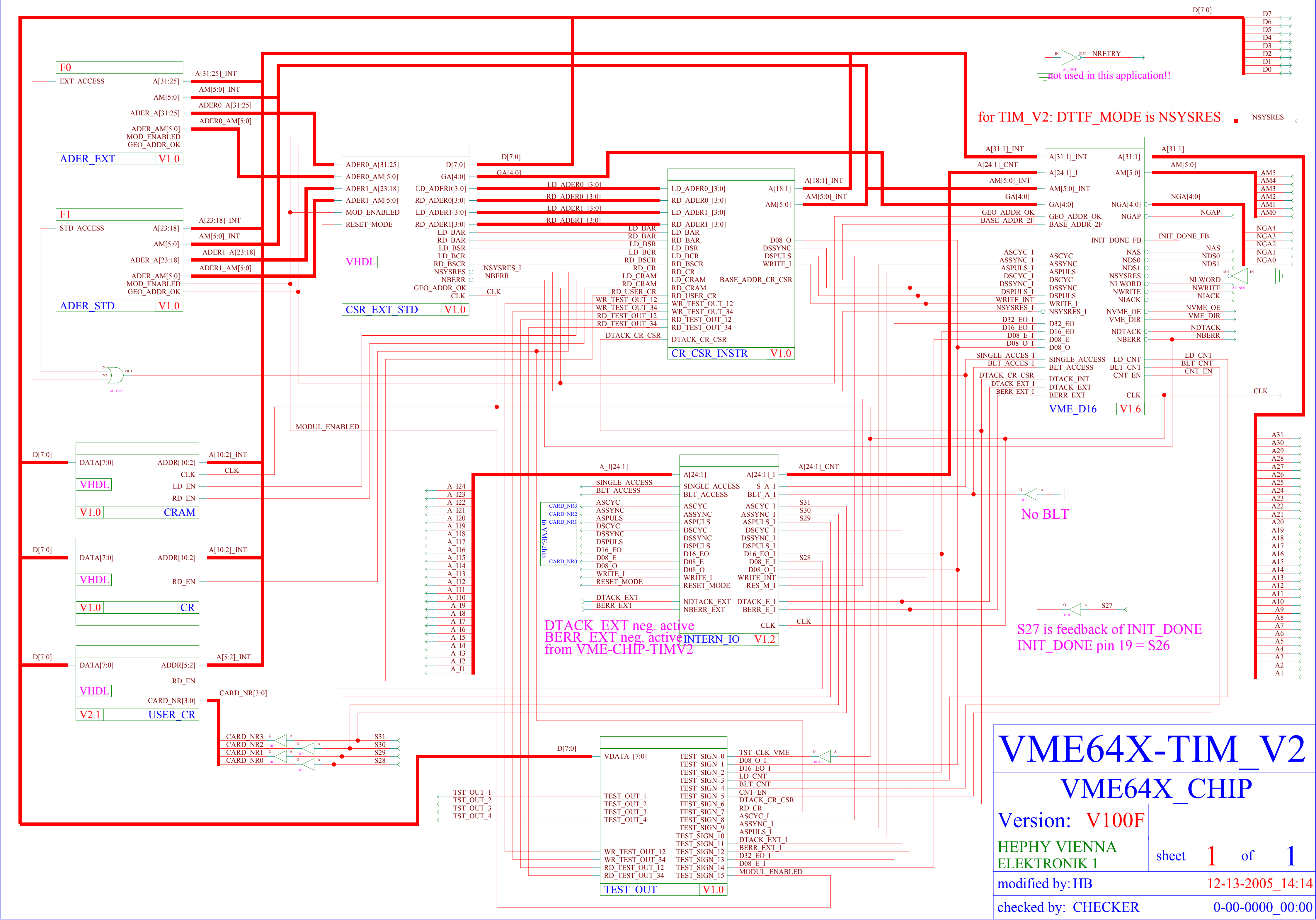
2.7.6 CRAM space

The configuration RAM (CRAM) is defined as a RAM for special purpose. The size is 512 bytes. The CRAM is located at addresses 0x03003..0x037FF.
The contains of the CRAM has to be defined!!!

3 Softwareguide for the VME64x Interface

3.1 Module enable

After power-up the module is disabled through the default value of the "ENABLE MODULE"-bit of the BitSet-register in the CommandStatusRegister (CSR) of VME64x.
To enable the module, one has to set bit 4 in the CSR, that means to write 0x10 to address 0x7FFFB with AM=0x2F.



VME64X-TIM_V2

VME64X_CHIP

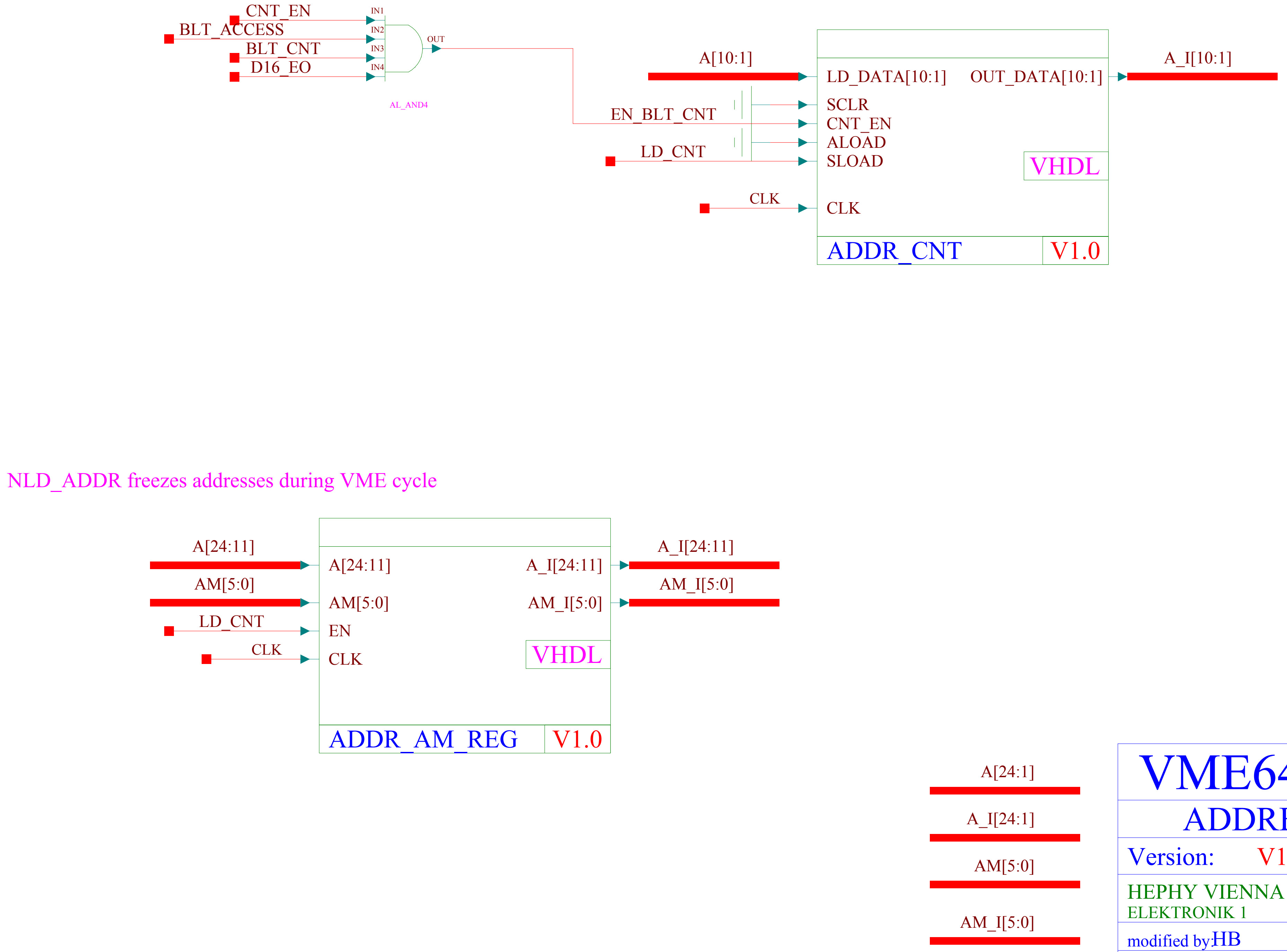
Version: V100F

HEPHY VIENNA
ELEKTRONIK 1

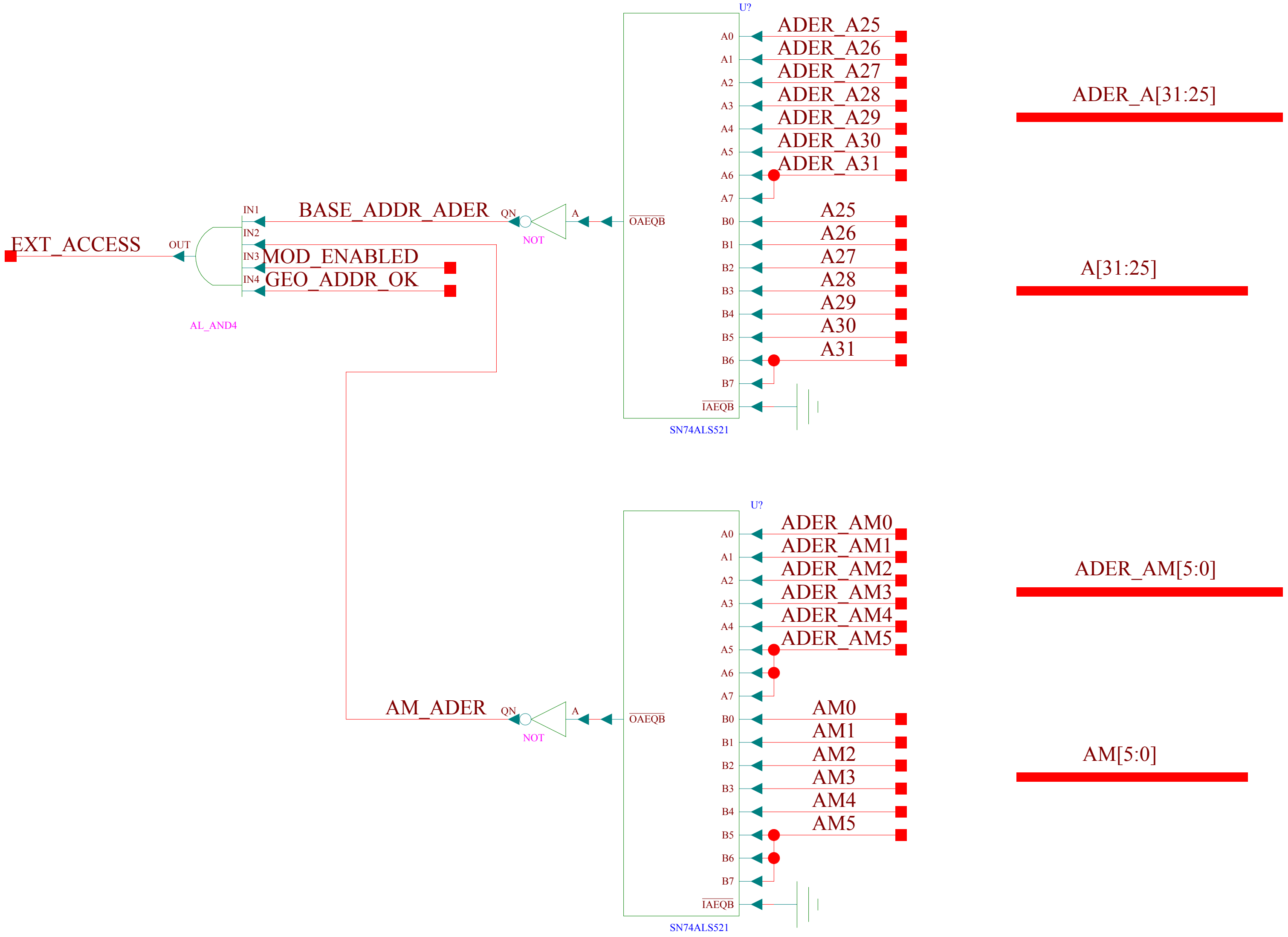
sheet 1 of 1

modified by: HB 12-13-2005_14:14

checked by: CHECKER 0-00-0000_00:00



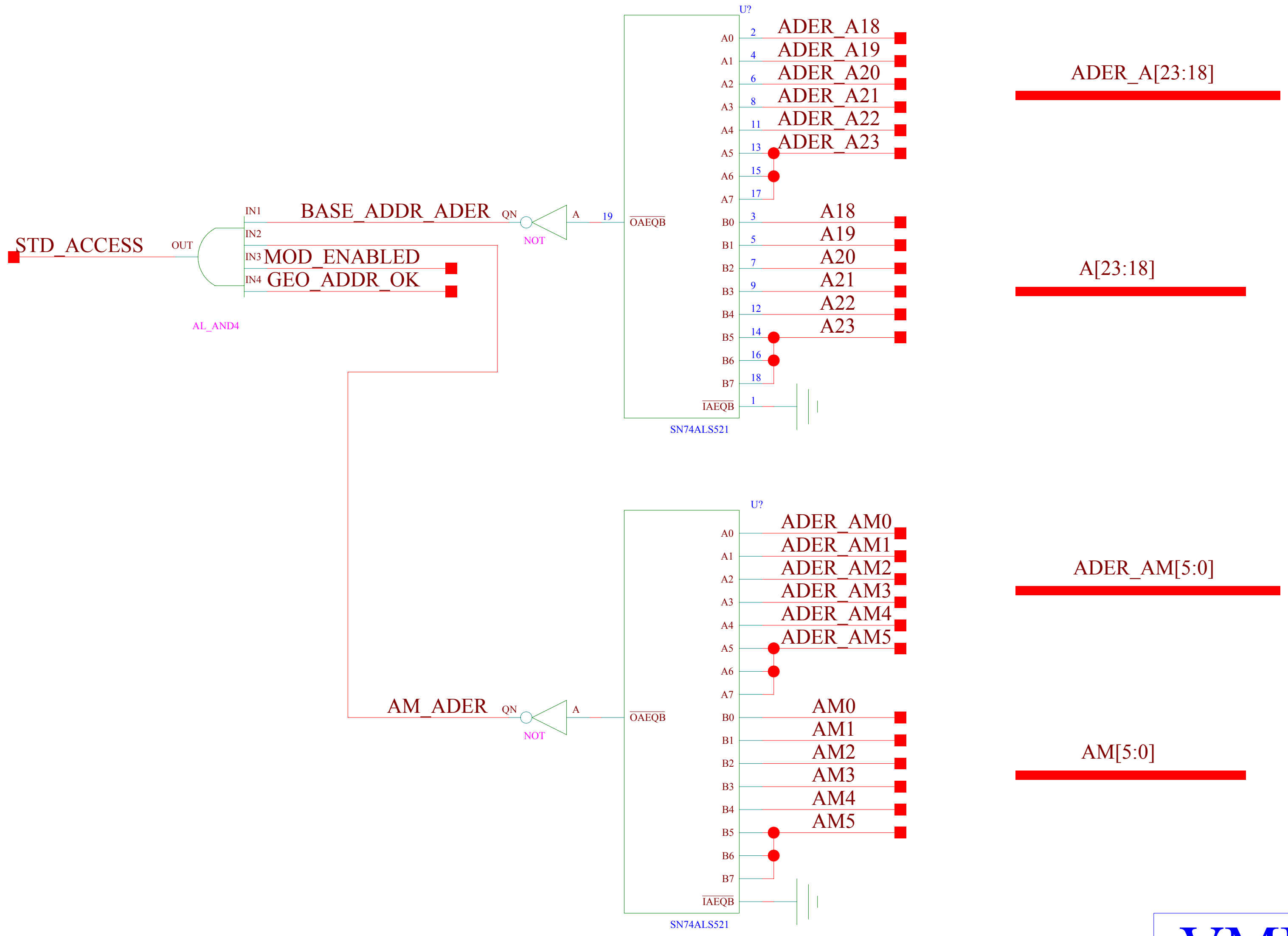
VME64X-CHIP	
ADDRESSES_AM	
Version: V1.1	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by: HB	9-23-2005_10:45
checked by: CHECKER	0-00-0000_00:00



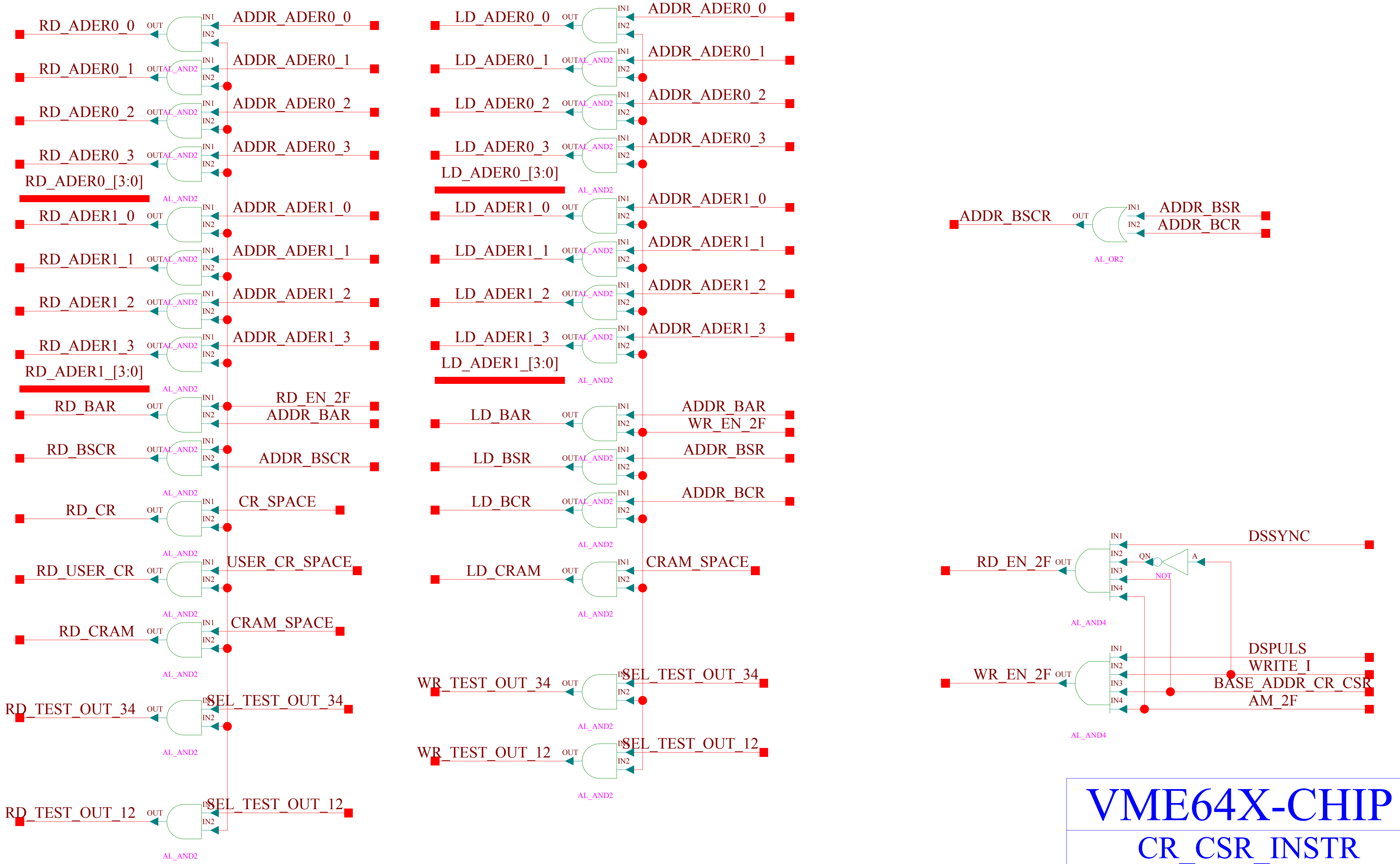
VME64X-CHIP

ADER_EXT

Version: V1.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by: HB	8-29-2005_9:32
checked by: CHECKER	0-00-0000_00:00



VME64X-CHIP	
ADER_STD	
Version: V1.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by HB	8-29-2005_9:33
checked by: CHECKER	0-00-0000_00:00



VME64X-CHIP

CR_CSR_INSTR

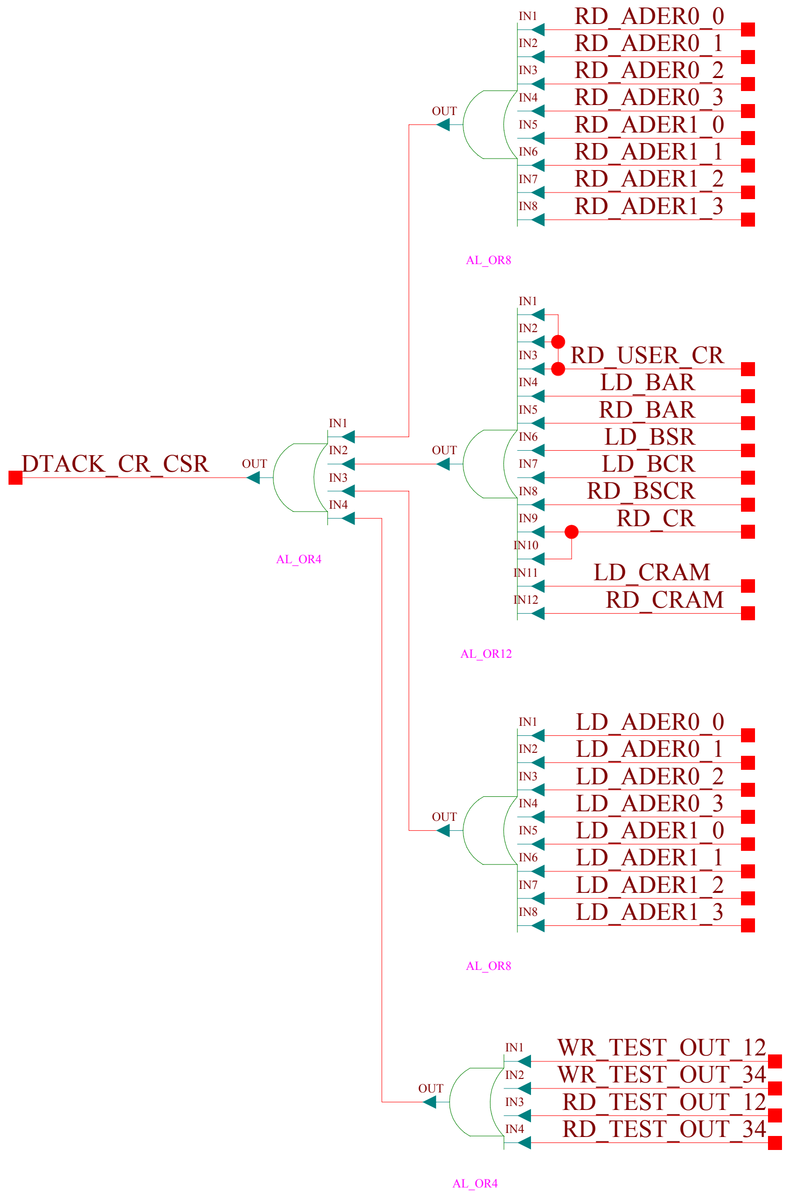
Version: V1.0

HEPHY VIENNA
ELEKTRONIK 1

sheet 1 of 5

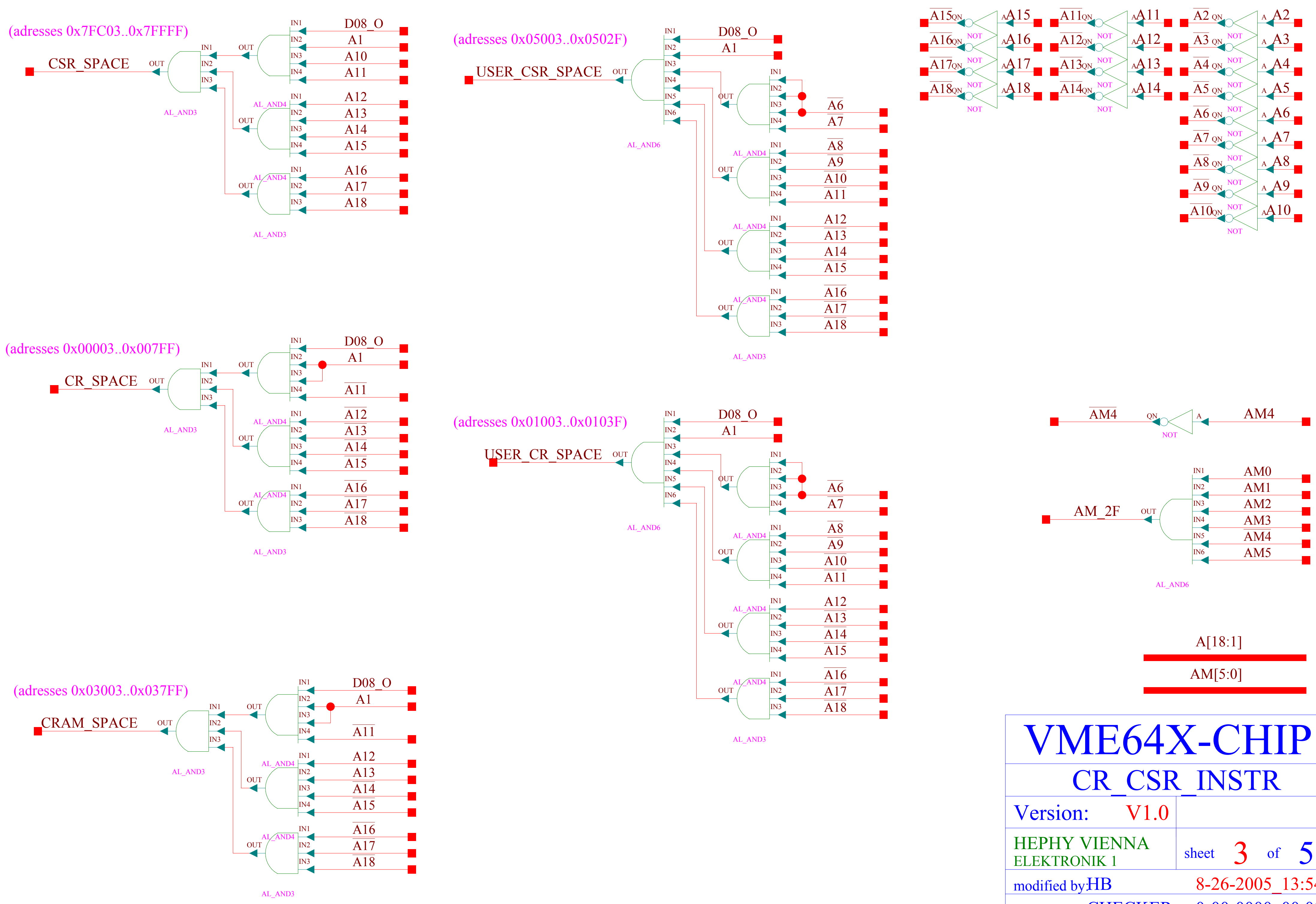
modified by HB 8-26-2005_13:54

checked by: CHECKER 0-00-0000_00:00



no BERR for CR/CSR access!!

VME64X-CHIP	
CR_CSR_INSTR	
Version: V1.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 2 of 5
modified by:HB	8-26-2005_13:54
checked by: CHECKER	0-00-0000_00:00



VME64X-CHIP

CR_CSR_INSTR

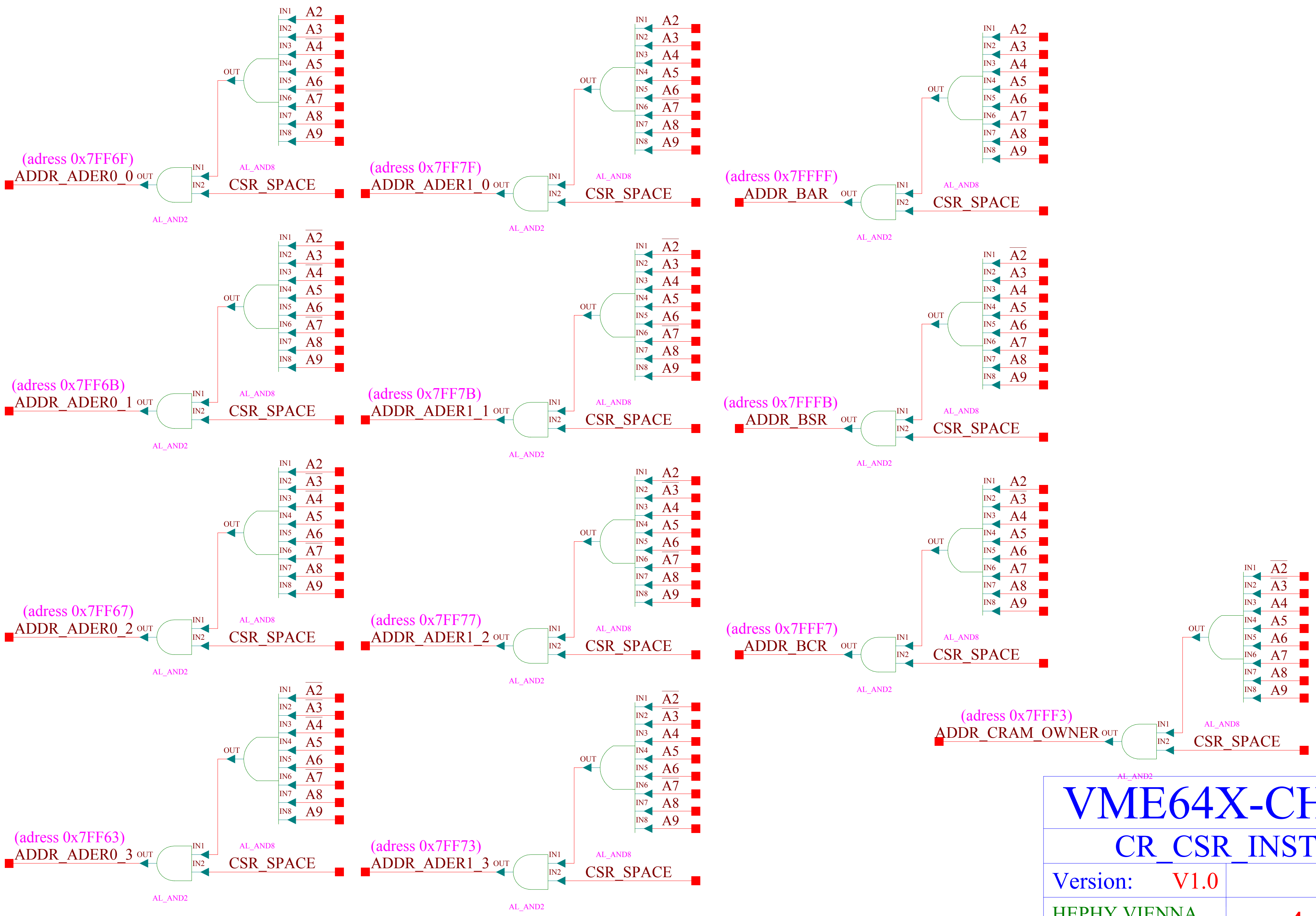
Version: V1.0

HEPHY VIENNA
ELEKTRONIK 1

sheet 3 of 5

modified by HB 8-26-2005_13:54

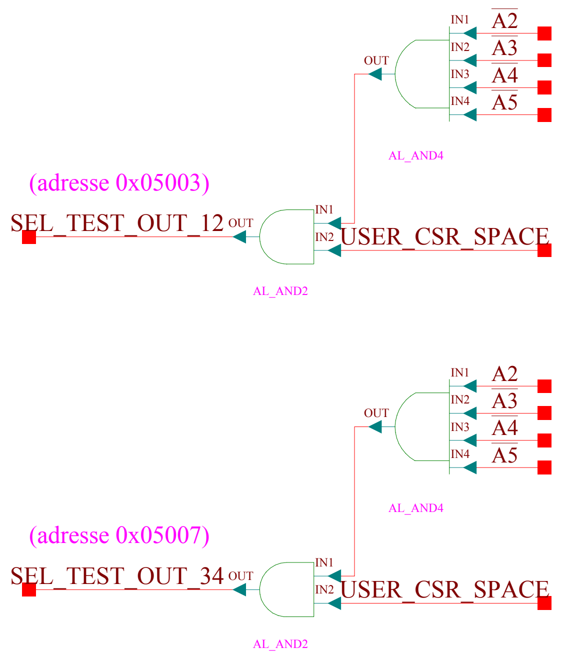
checked by: CHECKER 0-00-0000_00:00



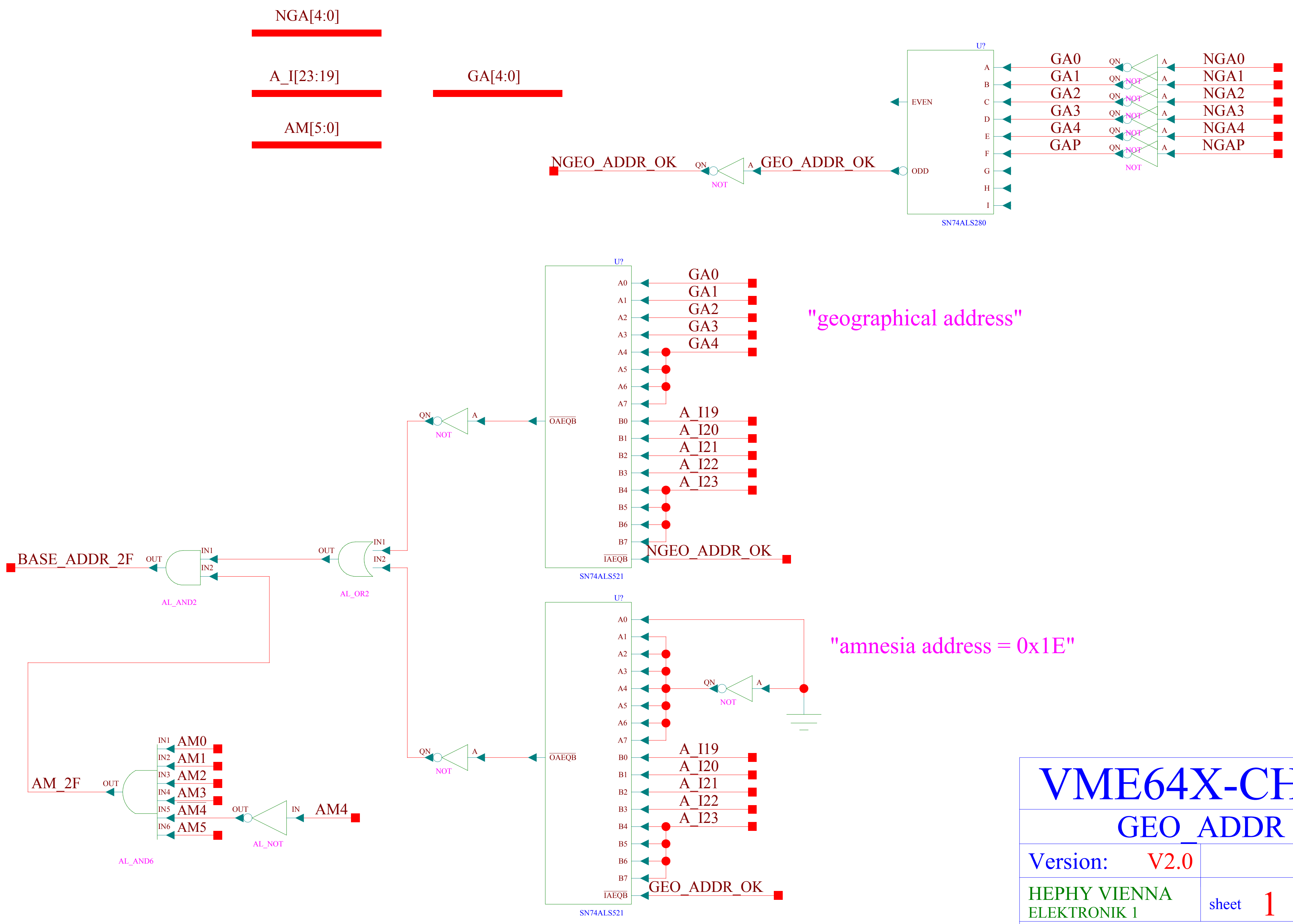
VME64X-CHIP

CR_CSR_INSTR

Version: V1.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 4 of 5
modified by:HB	8-26-2005_13:54
checked by: CHECKER	0-00-0000_00:00



VME64X-CHIP	
CR_CSR_INSTR	
Version: V1.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 5 of 5
modified by: HB	8-26-2005_13:54
checked by: CHECKER	0-00-0000_00:00



VME64X-CHIP

GEO_ADDR

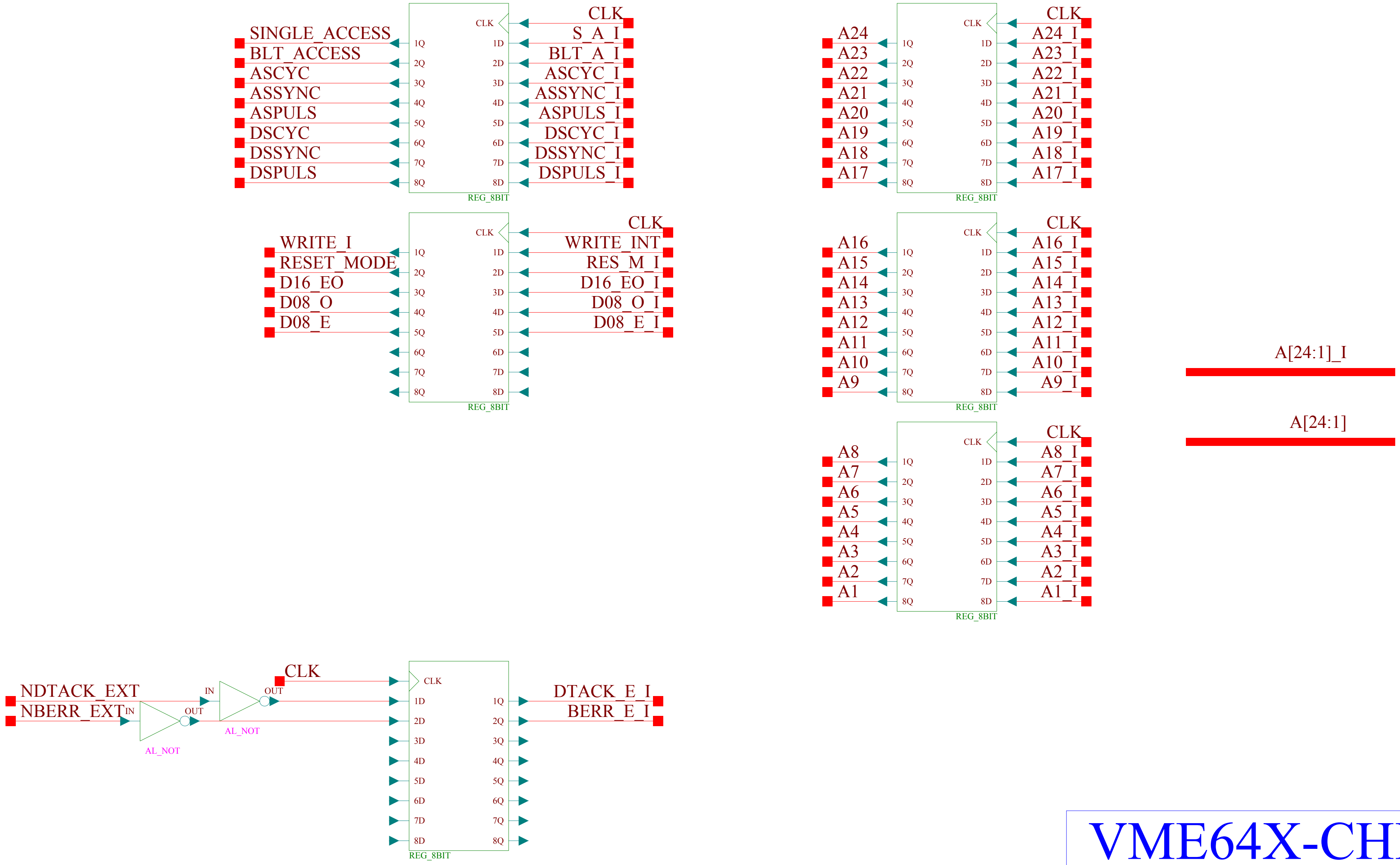
Version: V2.0

HEPHY VIENNA
ELEKTRONIK 1

sheet 1 of 1

modified by: HB 12-7-2005_14:36

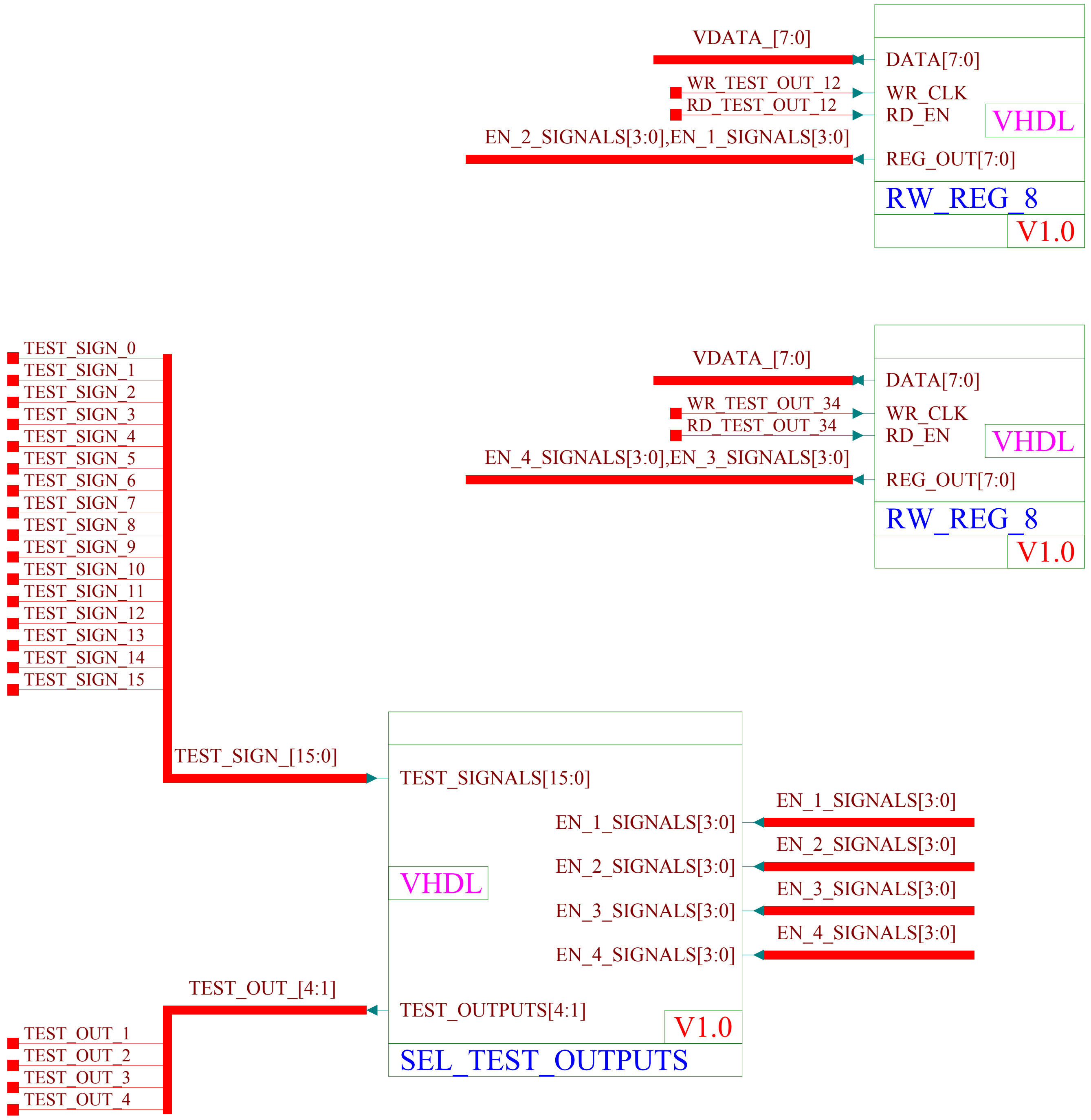
checked by: CHECKER 0-00-0000_00:00



VME64X-CHIP

INTERN_IO

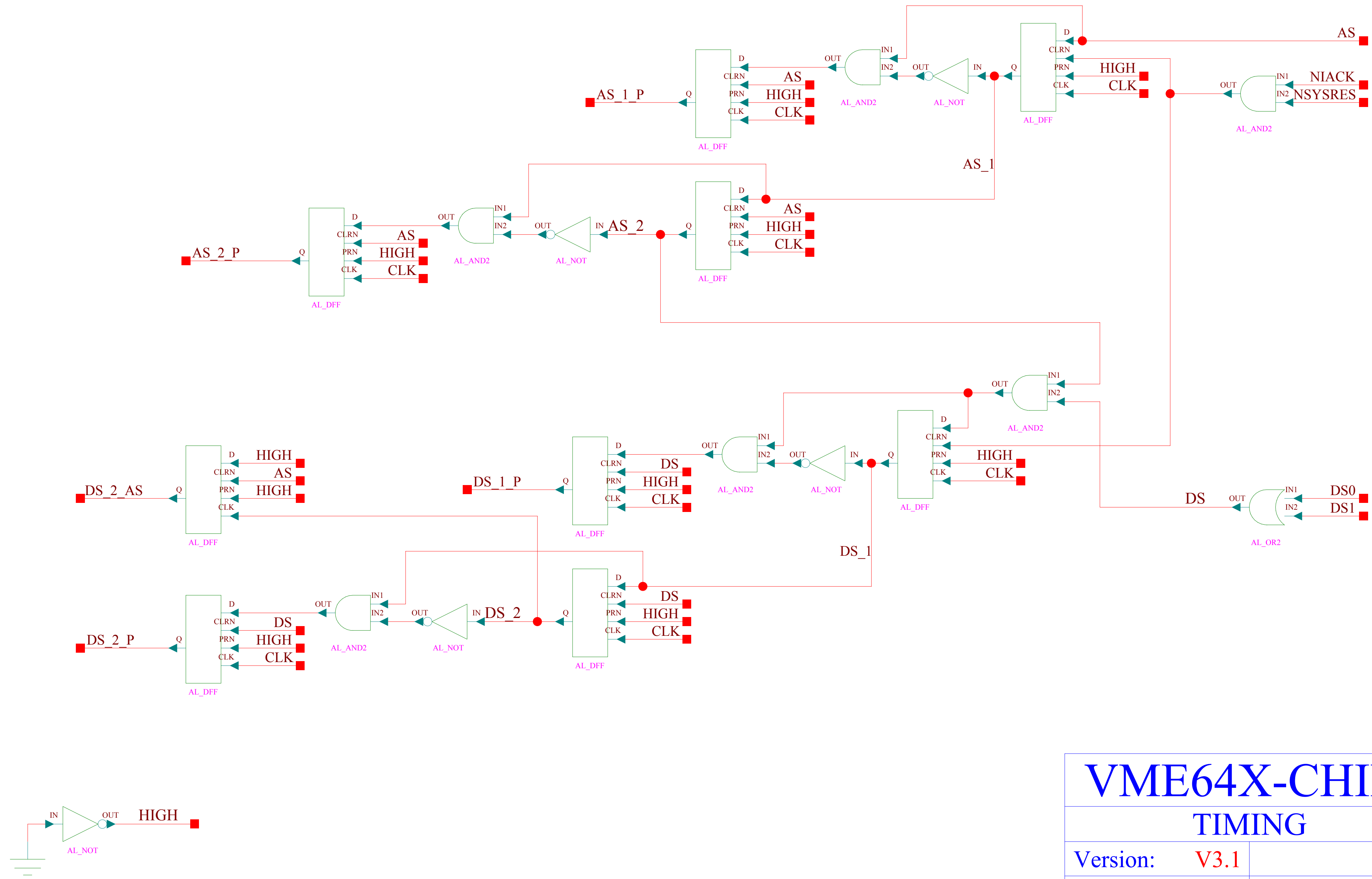
Version: V1.2	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by HB	10-20-2005_9:17
checked by: CHECKER	0-00-0000_00:00



VME64X-CHIP

TEST_OUT

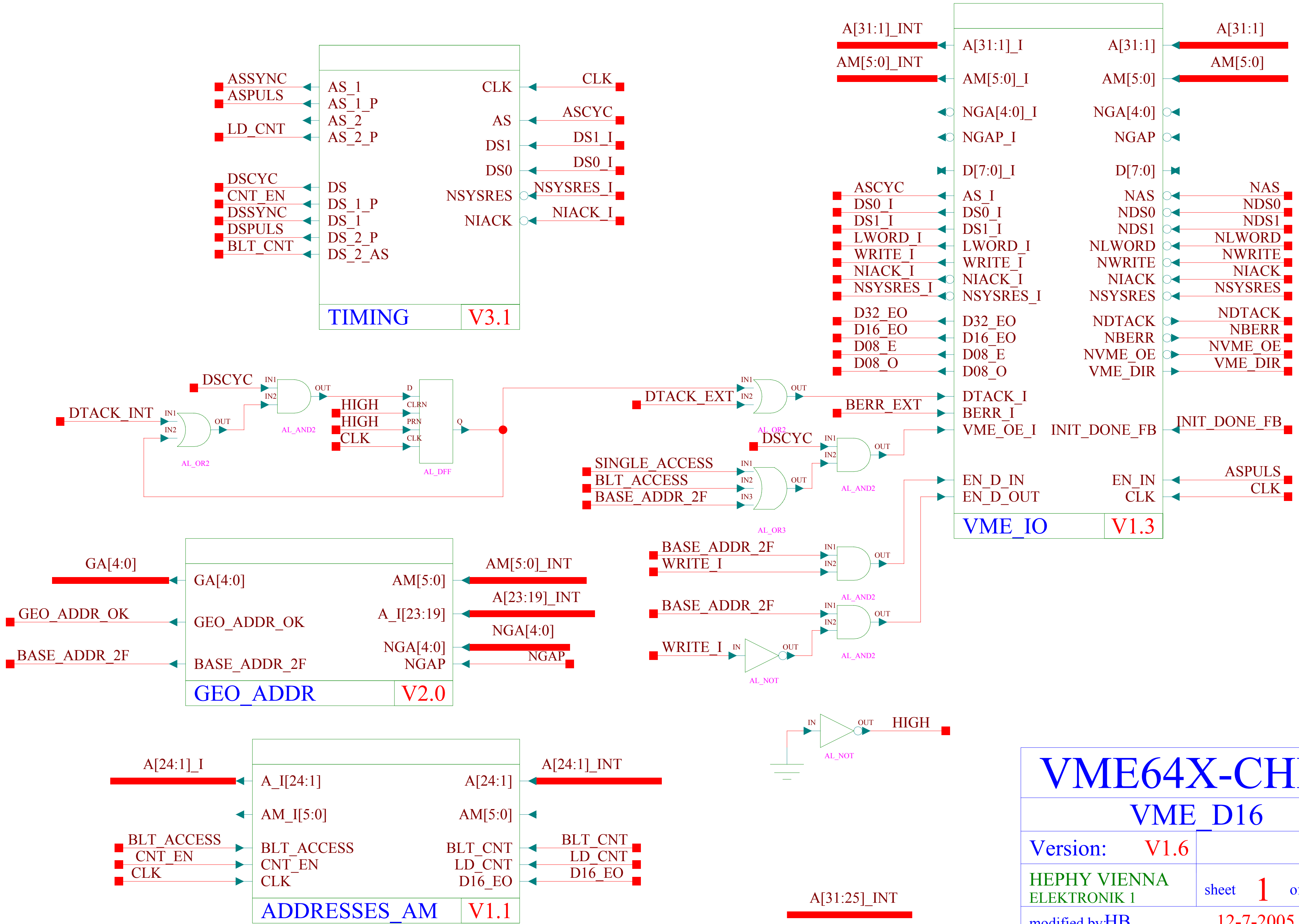
Version: V1.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by:HB	8-29-2005_11:07
checked by: CHECKER	0-00-0000_00:00



VME64X-CHIP

TIMING

Version: V3.1	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by:HB	10-7-2005_14:18
checked by: CHECKER	0-00-0000_00:00



VME64X-CHIP

VME_D16

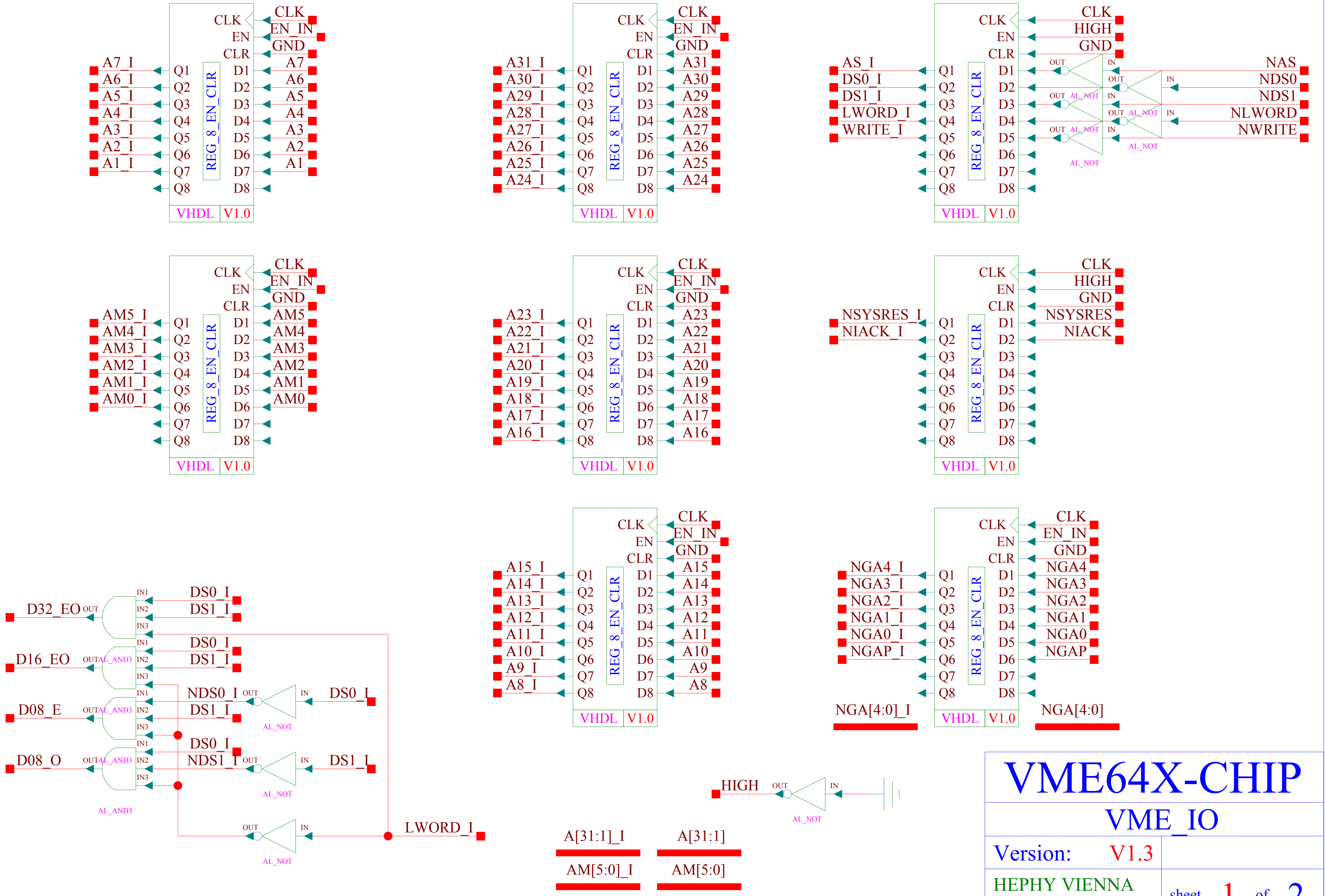
Version: V1.6

HEPHY VIENNA
ELEKTRONIK 1

sheet 1 of 1

modified by HB 12-7-2005_14:57

checked by: CHECKER 0-00-0000_00:00


REG_8_EN_CLR
CLK
EN
CLR
Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8
D1 D2 D3 D4 D5 D6 D7 D8
VHDL
V1.0
EN_IN
GND
HIGH
A7_I A6_I A5_I A4_I A3_I A2_I A1_I
A7 A6 A5 A4 A3 A2 A1
A31_I A30_I A29_I A28_I A27_I A26_I A25_I A24_I
A31 A30 A29 A28 A27 A26 A25 A24
AS_I DS0_I DS1_I LWORD_I WRITE_I
NAS NDS0 NDS1 NLWORD NWRITE
AM5_I AM4_I AM3_I AM2_I AM1_I AM0_I
AM5 AM4 AM3 AM2 AM1 AM0
A23_I A22_I A21_I A20_I A19_I A18_I A17_I A16_I
A23 A22 A21 A20 A19 A18 A17 A16
NSYSRES_I NIACK_I
NSYSRES NIACK
A15_I A14_I A13_I A12_I A11_I A10_I A9_I A8_I
A15 A14 A13 A12 A11 A10 A9 A8
NGA4_I NGA3_I NGA2_I NGA1_I NGA0_I NGAP_I
NGA4 NGA3 NGA2 NGA1 NGA0 NGAP
NGA[4:0]_I
NGA[4:0]
AL_NOT
AL_AND3
IN
OUT
IN1 IN2 IN3
D32_EO
D16_EO
D08_E
D08_O
DS0_I
DS1_I
NDS0_I
NDS1_I
DS0_I
DS1_I
LWORD_I
HIGH
A[31:1]_I
A[31:1]
AM[5:0]_I
AM[5:0]

VME64X-CHIP

VME_IO

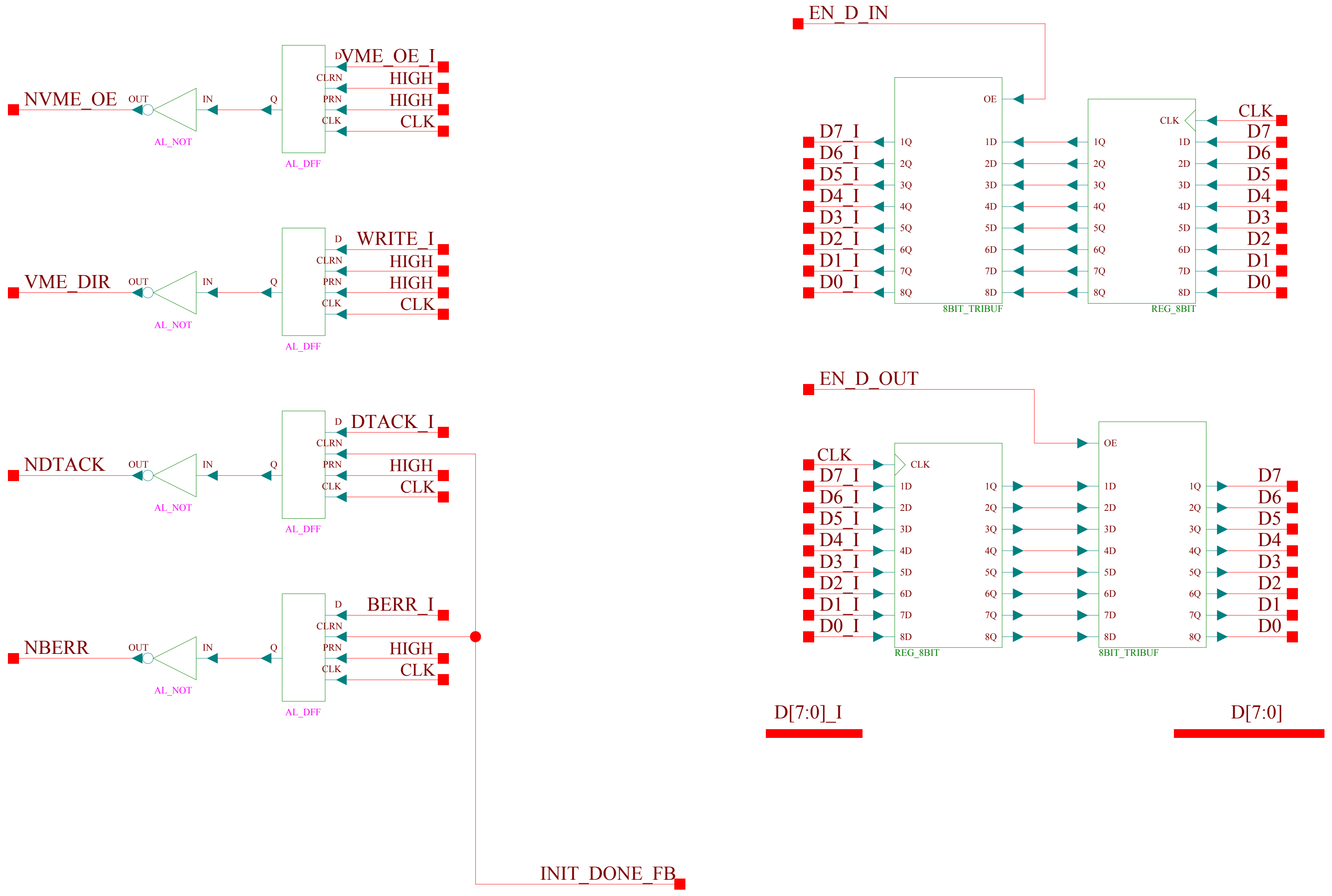
Version: V1.3

HEPHY VIENNA
ELEKTRONIK 1

sheet 1 of 2

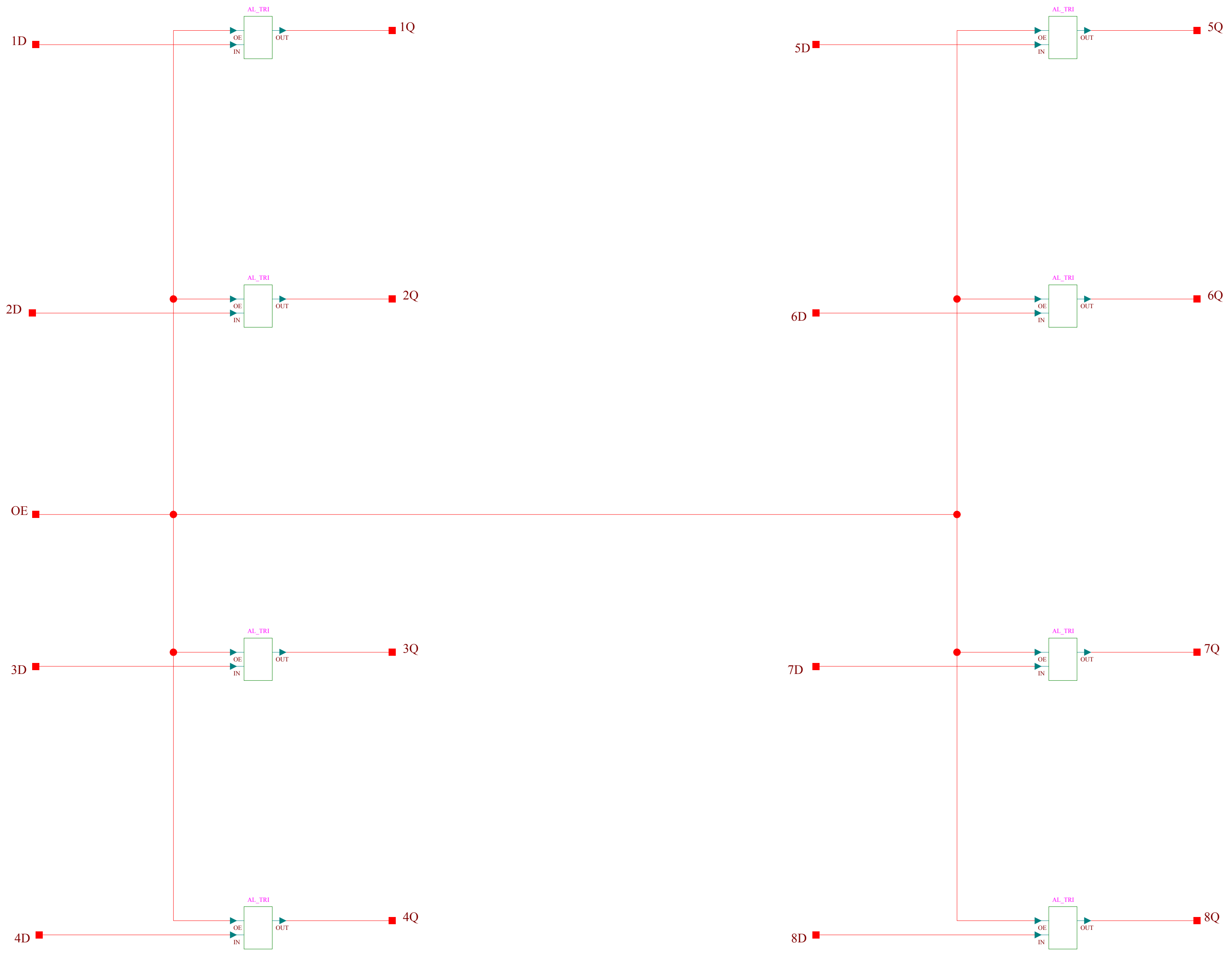
modified by HB 10-21-2005_14:41

checked by: CHECKER 0-00-0000_00:00



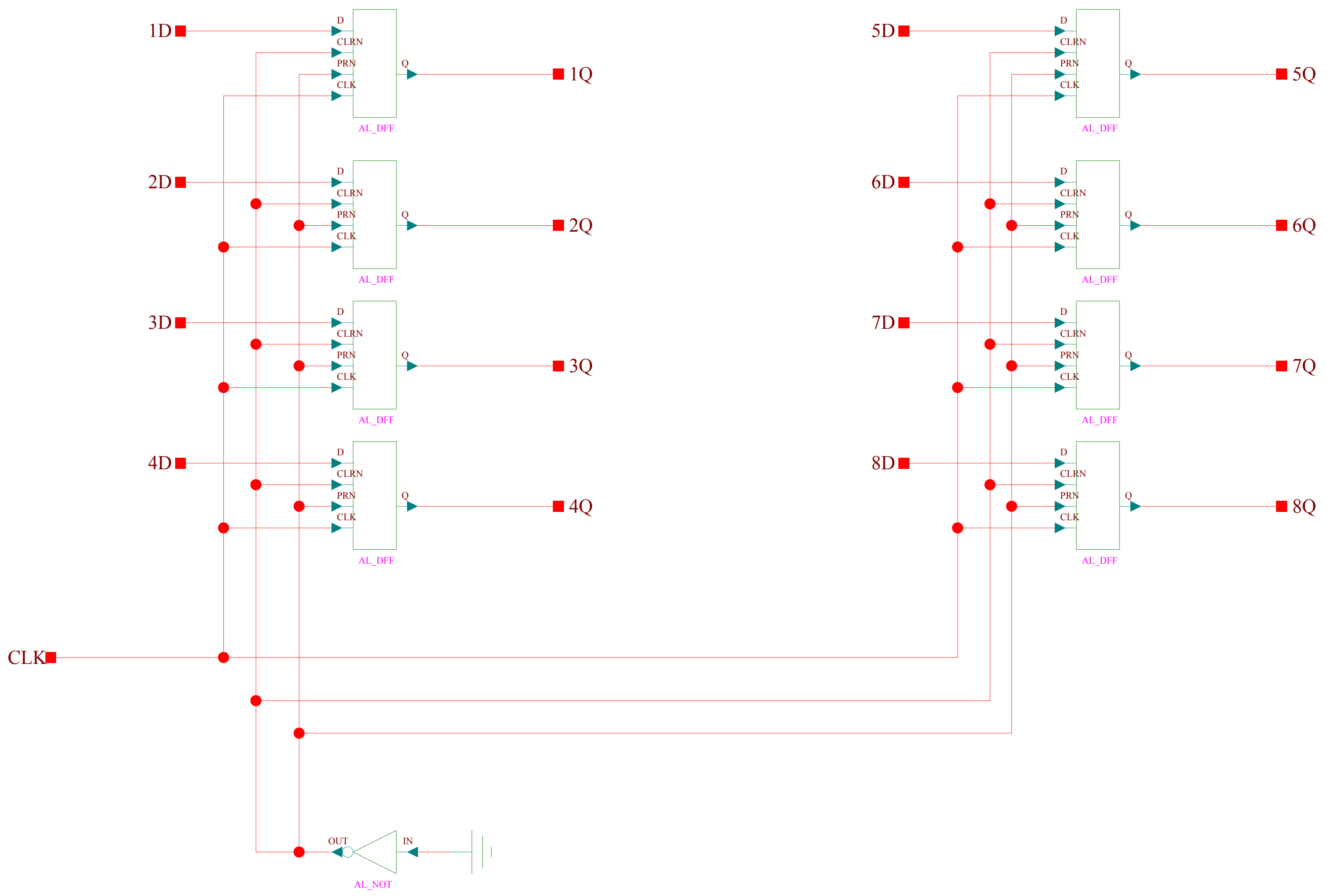
VME64X-CHIP	
VME_IO	
Version: V1.3	
HEPHY VIENNA ELEKTRONIK 1	sheet 2 of 2
modified by HB	10-21-2005_14:41
checked by: CHECKER	0-00-0000_00:00

8bit_tribuf



reg_8bit

8-bit register generated by LAB3



```
------------------------------------------------------------
--                                                        --
-- LOGIC CORE: GTL-module vme64x interface chip logic     --
-- MODULE NAME: cr                                        --
-- INSTITUTION: Hephy Vienna                              --
-- DESIGNER: H. Bergauer                                  --
--                                                        --
-- VERSION: V1.0                                          --
-- DATE: 08 2005                                          --
--                                                        --
-- FUNCTIONAL DESCRIPTION:                                --
-- configuration ROM for VME64x                           --
-- range: 0x03 - 0x7FF                                    --
--                                                        --
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY lpm;
USE lpm.lpm_components.ALL;

ENTITY cr IS
    PORT(
        addr    : IN    STD_LOGIC_VECTOR(10 DOWNTO 2);
        rd_en   : IN    STD_LOGIC;
        data    : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END cr;

ARCHITECTURE rtl OF cr IS
BEGIN

-- Configuration ROM für VME64x mit 512x8 bits
    cr_rom:lpm_rom
  GENERIC MAP   (LPM_WIDTH => 8,
   LPM_WIDTHAD => 9,
            LPM_OUTDATA => "UNREGISTERED",
            LPM_ADDRESS_CONTROL => "UNREGISTERED",
   LPM_FILE => "cr.mif")
  PORT MAP  (address => addr,
   memenab => rd_en,
   q => data);

END ARCHITECTURE rtl;
```

```
------------------------------------------------------------
--                                                        --
-- LOGIC CORE: GTL-module vme64x interface chip logic     --
-- MODULE NAME: cram                                      --
-- INSTITUTION: Hephy Vienna                              --
-- DESIGNER: H. Bergauer                                  --
--                                                        --
-- VERSION: V1.0                                          --
-- DATE: 08 2005                                          --
--                                                        --
-- FUNCTIONAL DESCRIPTION:                                --
-- configuration RAM 512x8 for VME64x                     --
-- range: 0x03003 - 0x037FF                               --
--                                                        --
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY lpm;
USE lpm.lpm_components.ALL;

ENTITY cram IS
    PORT(
        addr    : IN    STD_LOGIC_VECTOR(10 DOWNTO 2);
        clk : IN    STD_LOGIC;
        ld_en   : IN    STD_LOGIC;
        rd_en   : IN    STD_LOGIC;
        data    : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END cram;

ARCHITECTURE rtl OF cram IS
BEGIN

-- Configuration RAM für VME64x mit 512x8 bits
    config_ram: lpm_ram_io
  GENERIC MAP   (LPM_WIDTH => 8,
   LPM_WIDTHAD => 9,
            LPM_INDATA => "REGISTERED",
            LPM_OUTDATA => "UNREGISTERED",
            LPM_ADDRESS_CONTROL => "REGISTERED")
  PORT MAP  (address => addr,
   inclock => clk,
   we => ld_en,
   outenab => rd_en,
   dio => data);

END ARCHITECTURE rtl;
```

```
-------------------------------------------------------------
--                                                         --
-- LOGIC CORE: GTL-module vme64x interface chip logic      --
-- MODULE NAME: csr                                        --
-- INSTITUTION: Hephy Vienna                               --
-- DESIGNER: H. Bergauer                                   --
--                                                         --
-- VERSION: V100B                                          --
-- DATE: 07 2005                                           --
--                                                         --
-- FUNCTIONAL DESCRIPTION:                                 --
-- control/status register                                 --
-- range: 0x7FC00 - 0x7FFFF                                --
-- V100B for TIM_V2 of GT- and DTTF-system                 --
--                                                         --
-------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY altera;
USE altera.maxplus2.ALL;
LIBRARY lpm;
USE lpm.lpm_components.ALL;

ENTITY csr IS
    PORT(
  clk : IN  STD_LOGIC;
  d : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0);
  ga : IN   STD_LOGIC_VECTOR(4 DOWNTO 0);
  nsysres : IN  STD_LOGIC;
  nberr : IN    STD_LOGIC;
  geo_addr_ok : IN  STD_LOGIC;
  ld_ader0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  ld_ader1 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  rd_ader0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  rd_ader1 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  ld_bar : IN   STD_LOGIC;
  rd_bar : IN   STD_LOGIC;
  ld_bsr : IN   STD_LOGIC;
  ld_bcr : IN   STD_LOGIC;
  rd_bscr : IN  STD_LOGIC;
  reset_mode : INOUT    STD_LOGIC;
  mod_enabled : INOUT   STD_LOGIC;
        ader0_a : OUT   STD_LOGIC_VECTOR(31 DOWNTO 25);
        ader1_a : OUT   STD_LOGIC_VECTOR(23 DOWNTO 18);
        ader0_am    : OUT   STD_LOGIC_VECTOR(5 DOWNTO 0);
        ader1_am    : OUT   STD_LOGIC_VECTOR(5 DOWNTO 0));
END csr;

ARCHITECTURE rtl OF csr IS
 CONSTANT amnesia_addr: STD_LOGIC_VECTOR(4 DOWNTO 0) := "11110";
    SIGNAL aclr: STD_LOGIC;
    SIGNAL ader0_3_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader0_2_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader0_1_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader0_0_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader1_3_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader1_2_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader1_1_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader1_0_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL bsr_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL bcr_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL bscr_in: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL bar_in: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL set_res_mode: STD_LOGIC;
    SIGNAL dis_res_mode: STD_LOGIC;
    SIGNAL en_module: STD_LOGIC;
    SIGNAL dis_module: STD_LOGIC;
    SIGNAL set_berr_flag: STD_LOGIC;
    SIGNAL clr_berr_flag: STD_LOGIC;
    SIGNAL berr_flag: STD_LOGIC;
BEGIN
 aclr <= NOT nsysres;
    
-- GT-system base-address A31-A25
 ader0_a <= ader0_3_out(7 DOWNTO 1);
 ader0_am <= ader0_0_out(7 DOWNTO 2);

-- DTTF-system base-address A23-A18
 ader1_a <= ader1_2_out(7 DOWNTO 2);
 ader1_am <= ader1_0_out(7 DOWNTO 2);

-- ****************************
-- load ader0_3 register
    ader0_3_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader0(3),
            aclr => aclr,
   q => ader0_3_out);

-- load ader0_2 register
    ader0_2_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader0(2),
   aclr => aclr,
   q => ader0_2_out);

-- load ader0_1 register
    ader0_1_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader0(1),
   aclr => aclr,
   q => ader0_1_out);

-- load ader0_0 register
    ader0_0_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader0(0),
   aclr => aclr,
   q => ader0_0_out);

-- read ader0_3 register
    ader0_3_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader0_3: tri
   PORT MAP(ader0_3_out(i),
      rd_ader0(3),
      d(i));
 END GENERATE ader0_3_read;

-- read ader0_2 register
    ader0_2_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader0_2: tri
   PORT MAP(ader0_2_out(i),
      rd_ader0(2),
      d(i));
 END GENERATE ader0_2_read;

-- read ader0_1 register
    ader0_1_read:
```

```
 FOR i IN 0 TO 7 GENERATE
    tri_ader0_1: tri
   PORT MAP(ader0_1_out(i),
      rd_ader0(1),
      d(i));
 END GENERATE ader0_1_read;

-- read ader0_0 register
    ader0_0_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader0_0: tri
   PORT MAP(ader0_0_out(i),
      rd_ader0(0),
      d(i));
 END GENERATE ader0_0_read;

-- ***************************
-- load ader1_3 register
    ader1_3_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader1(3),
   aclr => aclr,
   q => ader1_3_out);

-- load ader1_2 register
    ader1_2_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader1(2),
   aclr => aclr,
   q => ader1_2_out);

-- load ader1_1 register
    ader1_1_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader1(1),
   aclr => aclr,
   q => ader1_1_out);

-- load ader1_0 register
    ader1_0_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader1(0),
   aclr => aclr,
   q => ader1_0_out);

-- read ader1_3 register
    ader1_3_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader1_3: tri
   PORT MAP(ader1_3_out(i),
      rd_ader1(3),
      d(i));
 END GENERATE ader1_3_read;

-- read ader1_2 register
    ader1_2_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader1_2: tri
   PORT MAP(ader1_2_out(i),
      rd_ader1(2),
      d(i));
 END GENERATE ader1_2_read;

-- read ader1_1 register
    ader1_1_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader1_1: tri
   PORT MAP(ader1_1_out(i),
      rd_ader1(1),
      d(i));
 END GENERATE ader1_1_read;

-- read ader1_0 register
    ader1_0_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader1_0: tri
   PORT MAP(ader1_0_out(i),
      rd_ader1(0),
      d(i));
 END GENERATE ader1_0_read;

-- ***************************
-- load bit set register
    bsr_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_bsr,
   aclr => aclr,
   q => bsr_out);

 set_res_mode <= bsr_out(7);
 en_module <= bsr_out(4);
 set_berr_flag <= bsr_out(3);

-- load bit clear register
    bcr_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_bcr,
   aclr => aclr,
   q => bcr_out);

 dis_res_mode <= bcr_out(7);
 dis_module <= bcr_out(4);
 clr_berr_flag <= bcr_out(3);

-- ***************************
-- setting and clearing bits of BSR and BCR

PROCESS (set_res_mode, dis_res_mode, en_module, dis_module, set_berr_flag, clr_berr_flag,
BEGIN
    IF set_res_mode='1'THEN
            reset_mode <= '1';
    ELSIF (set_res_mode='0' AND dis_res_mode='1') OR nsysres='0' THEN
            reset_mode <= '0';
    END IF;
--  IF en_module='1' OR nsysres='0' THEN
--          mod_enabled <= '1';
--  ELSIF (en_module='0' AND dis_module='1' AND nsysres='1') THEN
--          mod_enabled <= '0';
--  END IF;
    IF en_module='1' THEN
            mod_enabled <= '1';
    ELSIF (en_module='0' AND dis_module='1') OR nsysres='0' THEN
            mod_enabled <= '0';
    END IF;

-- set berr_flag if a BERR is generated on board or set_berr_flag='1'??
-- clear berr_flag if a SYSRES is generated or clr_berr_flag='1' (and set_berr_flag='0')
-- see VME64x-specification Rule 10.16

    IF set_berr_flag='1' OR nberr='0' THEN
            berr_flag <= '1';
    ELSIF (set_berr_flag='0' AND clr_berr_flag='1') OR nsysres='0' THEN
            berr_flag <= '0';
    END IF;
END PROCESS;

-- ****************************
 bscr_in <= reset_mode & '0' & '0' & mod_enabled & berr_flag & '0' & '0' & '0'; 

-- read bit set/clear register
    bscr_read:
 FOR i IN 0 TO 7 GENERATE
    tri_bscr: tri
   PORT MAP(bscr_in(i),
      rd_bscr,
      d(i));
 END GENERATE bscr_read;

-- ****************************
-- setting BAR with GA or amnesia address

PROCESS (geo_addr_ok, ga)
BEGIN
 IF geo_addr_ok = '1' THEN
   bar_in(7 DOWNTO 3) <= ga(4 DOWNTO 0); 
   bar_in(2 DOWNTO 0) <= "000"; 
 ELSE
   bar_in <= amnesia_addr & '0' & '0' & '0'; 
 END IF;
END PROCESS;

-- read base address register
    bar_read:
 FOR i IN 0 TO 7 GENERATE
    tri_bar: tri
   PORT MAP(bar_in(i),
      rd_bar,
      d(i));
 END GENERATE bar_read;

END ARCHITECTURE rtl;
```

```
------------------------------------------------------------
--                                                        --
-- LOGIC CORE: GTL-module vme64x interface chip logic     --
-- MODULE NAME: csr_ext_std                               --
-- INSTITUTION: Hephy Vienna                              --
-- DESIGNER: H. Bergauer                                  --
--                                                        --
-- VERSION: V1.0                                          --
-- DATE: 08 2005                                          --
--                                                        --
-- FUNCTIONAL DESCRIPTION:                                --
-- control/status register                                --
-- range: 0x7FC00 - 0x7FFFF                               --
-- F0 => extended access A31-A25                          --
-- F1 => standard access A23-A18                          --
--                                                        --
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY altera;
USE altera.maxplus2.ALL;
LIBRARY lpm;
USE lpm.lpm_components.ALL;

ENTITY csr_ext_std IS
    PORT(
  clk : IN  STD_LOGIC;
  d : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0);
  ga : IN   STD_LOGIC_VECTOR(4 DOWNTO 0);
  nsysres : IN  STD_LOGIC;
  nberr : IN    STD_LOGIC;
  geo_addr_ok : IN  STD_LOGIC;
  ld_ader0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  ld_ader1 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  rd_ader0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  rd_ader1 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  ld_bar : IN   STD_LOGIC;
  rd_bar : IN   STD_LOGIC;
  ld_bsr : IN   STD_LOGIC;
  ld_bcr : IN   STD_LOGIC;
  rd_bscr : IN  STD_LOGIC;
  reset_mode : INOUT    STD_LOGIC;
  mod_enabled : INOUT   STD_LOGIC;
        ader0_a : OUT   STD_LOGIC_VECTOR(31 DOWNTO 25);
        ader1_a : OUT   STD_LOGIC_VECTOR(23 DOWNTO 18);
        ader0_am    : OUT   STD_LOGIC_VECTOR(5 DOWNTO 0);
        ader1_am    : OUT   STD_LOGIC_VECTOR(5 DOWNTO 0));
END csr_ext_std;

ARCHITECTURE rtl OF csr_ext_std IS
 CONSTANT amnesia_addr: STD_LOGIC_VECTOR(4 DOWNTO 0) := "11110";
    SIGNAL aclr: STD_LOGIC;
    SIGNAL ader0_3_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader0_2_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader0_1_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader0_0_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader1_3_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader1_2_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader1_1_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL ader1_0_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL bsr_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL bcr_out: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL bscr_in: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL bar_in: STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL set_res_mode: STD_LOGIC;
    SIGNAL dis_res_mode: STD_LOGIC;
    SIGNAL en_module: STD_LOGIC;
    SIGNAL dis_module: STD_LOGIC;
    SIGNAL set_berr_flag: STD_LOGIC;
    SIGNAL clr_berr_flag: STD_LOGIC;
    SIGNAL berr_flag: STD_LOGIC;
BEGIN
 aclr <= NOT nsysres;
    
-- GT-system base-address A31-A25
 ader0_a <= ader0_3_out(7 DOWNTO 1);
 ader0_am <= ader0_0_out(7 DOWNTO 2);

-- DTTF-system base-address A23-A18
 ader1_a <= ader1_2_out(7 DOWNTO 2);
 ader1_am <= ader1_0_out(7 DOWNTO 2);

-- ****************************
-- load ader0_3 register
    ader0_3_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader0(3),
            aclr => aclr,
   q => ader0_3_out);

-- load ader0_2 register
    ader0_2_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader0(2),
   aclr => aclr,
   q => ader0_2_out);

-- load ader0_1 register
    ader0_1_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader0(1),
   aclr => aclr,
   q => ader0_1_out);

-- load ader0_0 register
    ader0_0_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader0(0),
   aclr => aclr,
   q => ader0_0_out);

-- read ader0_3 register
    ader0_3_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader0_3: tri
   PORT MAP(ader0_3_out(i),
      rd_ader0(3),
      d(i));
 END GENERATE ader0_3_read;

-- read ader0_2 register
    ader0_2_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader0_2: tri
   PORT MAP(ader0_2_out(i),
      rd_ader0(2),
      d(i));
 END GENERATE ader0_2_read;

-- read ader0_1 register
```

```
    ader0_1_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader0_1: tri
   PORT MAP(ader0_1_out(i),
      rd_ader0(1),
      d(i));
 END GENERATE ader0_1_read;

-- read ader0_0 register
    ader0_0_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader0_0: tri
   PORT MAP(ader0_0_out(i),
      rd_ader0(0),
      d(i));
 END GENERATE ader0_0_read;

-- ***************************
-- load ader1_3 register
    ader1_3_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader1(3),
   aclr => aclr,
   q => ader1_3_out);

-- load ader1_2 register
    ader1_2_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader1(2),
   aclr => aclr,
   q => ader1_2_out);

-- load ader1_1 register
    ader1_1_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader1(1),
   aclr => aclr,
   q => ader1_1_out);

-- load ader1_0 register
    ader1_0_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_ader1(0),
   aclr => aclr,
   q => ader1_0_out);

-- read ader1_3 register
    ader1_3_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader1_3: tri
   PORT MAP(ader1_3_out(i),
      rd_ader1(3),
      d(i));
 END GENERATE ader1_3_read;

-- read ader1_2 register
    ader1_2_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader1_2: tri
   PORT MAP(ader1_2_out(i),
      rd_ader1(2),
      d(i));
 END GENERATE ader1_2_read;

-- read ader1_1 register
    ader1_1_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader1_1: tri
   PORT MAP(ader1_1_out(i),
      rd_ader1(1),
      d(i));
 END GENERATE ader1_1_read;

-- read ader1_0 register
    ader1_0_read:
 FOR i IN 0 TO 7 GENERATE
    tri_ader1_0: tri
   PORT MAP(ader1_0_out(i),
      rd_ader1(0),
      d(i));
 END GENERATE ader1_0_read;

-- ***************************
-- load bit set register
    bsr_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_bsr,
   aclr => aclr,
   q => bsr_out);

 set_res_mode <= bsr_out(7);
 en_module <= bsr_out(4);
 set_berr_flag <= bsr_out(3);

-- load bit clear register
    bcr_load: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 8)
  PORT MAP  (data => d, 
   clock => clk,
   enable => ld_bcr,
   aclr => aclr,
   q => bcr_out);

 dis_res_mode <= bcr_out(7);
 dis_module <= bcr_out(4);
 clr_berr_flag <= bcr_out(3);

-- ***************************
-- setting and clearing bits of BSR and BCR

PROCESS (set_res_mode, dis_res_mode, en_module, dis_module, set_berr_flag, clr_berr_flag,
BEGIN
    IF set_res_mode='1'THEN
            reset_mode <= '1';
    ELSIF (set_res_mode='0' AND dis_res_mode='1') OR nsysres='0' THEN
            reset_mode <= '0';
    END IF;
--  IF en_module='1' OR nsysres='0' THEN
--          mod_enabled <= '1';
--  ELSIF (en_module='0' AND dis_module='1' AND nsysres='1') THEN
--          mod_enabled <= '0';
--  END IF;
    IF en_module='1' THEN
            mod_enabled <= '1';
    ELSIF (en_module='0' AND dis_module='1') OR nsysres='0' THEN
            mod_enabled <= '0';
    END IF;

```

```
-- set berr_flag if a BERR is generated on board or set_berr_flag='1'??
-- clear berr_flag if a SYSRES is generated or clr_berr_flag='1' (and set_berr_flag='0')
-- see VME64x-specification Rule 10.16

    IF set_berr_flag='1' OR nberr='0' THEN
            berr_flag <= '1';
    ELSIF (set_berr_flag='0' AND clr_berr_flag='1') OR nsysres='0' THEN
            berr_flag <= '0';
    END IF;
END PROCESS;

-- ***************************
 bscr_in <= reset_mode & '0' & '0' & mod_enabled & berr_flag & '0' & '0' & '0'; 

-- read bit set/clear register
    bscr_read:
 FOR i IN 0 TO 7 GENERATE
    tri_bscr: tri
   PORT MAP(bscr_in(i),
      rd_bscr,
      d(i));
 END GENERATE bscr_read;

-- ***************************
-- setting BAR with GA or amnesia address

PROCESS (geo_addr_ok, ga)
BEGIN
 IF geo_addr_ok = '1' THEN
   bar_in(7 DOWNTO 3) <= ga(4 DOWNTO 0); 
   bar_in(2 DOWNTO 0) <= "000"; 
 ELSE
   bar_in <= amnesia_addr & '0' & '0' & '0'; 
 END IF;
END PROCESS;

-- read base address register
    bar_read:
 FOR i IN 0 TO 7 GENERATE
    tri_bar: tri
   PORT MAP(bar_in(i),
      rd_bar,
      d(i));
 END GENERATE bar_read;

END ARCHITECTURE rtl;
```

```
------------------------------------------------------------
--                                                        --
-- LOGIC CORE: GTL-module vme64x interface chip logic     --
-- MODULE NAME: user_cr                                   --
-- INSTITUTION: Hephy Vienna                              --
-- DESIGNER: H. Bergauer                                  --
--                                                        --
-- VERSION: V2.1                                          --
-- DATE: 08 2005                                          --
--                                                        --
-- FUNCTIONAL DESCRIPTION:                                --
-- ROM for chip_id and version (each 4 bytes)             --
-- and serial number (VME64x)                             --
-- range: 0x01003 - 0x0103F                               --
--                                                        --
-- REVISION:                                              --
-- V2.1: CARD_NR from S24-S27 jumpers                     --
--                                                        --
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY lpm;
USE lpm.lpm_components.ALL;
LIBRARY altera;
USE altera.maxplus2.ALL;

ENTITY user_cr IS
    PORT(
        addr    : IN    STD_LOGIC_VECTOR(5 DOWNTO 2);
        card_nr : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        rd_en   : IN    STD_LOGIC;
        data    : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END user_cr;

ARCHITECTURE rtl OF user_cr IS
    SIGNAL addr_card_nr : std_logic;
    SIGNAL data_mem : STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL data_int : STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL data_tri : STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN

addr_card_nr <= NOT addr(5) AND NOT addr(4) AND addr(3) AND NOT addr(2) AND rd_en;

-- USER_CR for chip_id, version and SN with 16x8 bits
call_user_cr: lpm_rom
  GENERIC MAP   (LPM_WIDTH => 8,
   LPM_WIDTHAD => 4,
   LPM_OUTDATA => "UNREGISTERED",
   LPM_ADDRESS_CONTROL => "UNREGISTERED",
   LPM_FILE => "user_cr.mif")
  PORT MAP  (address => addr,
   memenab => rd_en,
   q => data_mem);

-- card_nr
data_int(7) <= data_mem(7);
data_int(6) <= data_mem(6);
data_int(5) <= data_mem(5);
data_int(4) <= data_mem(4);
data_int(3) <= card_nr(3);
data_int(2) <= card_nr(2);
data_int(1) <= card_nr(1);
data_int(0) <= card_nr(0);

-- mux for card_nr
call_mux: busmux
  GENERIC MAP   (WIDTH => 8)
  PORT MAP  (dataa => data_mem,
   datab => data_int,
   sel => addr_card_nr,
   result => data_tri);

tri_data:
FOR i IN 0 TO 7 GENERATE
 call_data_mem: tri
  PORT MAP(data_tri(i),
      rd_en,
      data(i));
END GENERATE tri_data;

END ARCHITECTURE rtl;
```

```
-------------------------------------------------------------
--                                                         --
-- LOGIC CORE: GTL-module vme64x interface chip logic     --
-- MODULE NAME: addr_am_reg                                --
-- INSTITUTION: Hephy Vienna                               --
-- DESIGNER: H. Bergauer                                   --
--                                                         --
-- VERSION: V1.0                                           --
-- DATE: 08 2005                                           --
--                                                         --
-- FUNCTIONAL DESCRIPTION:                                 --
-- input register for addresses and address modifier      --
--                                                         --
-------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY lpm;
USE lpm.lpm_components.ALL;

ENTITY addr_am_reg IS
    PORT(
        clk : IN    STD_LOGIC;
        en  : IN    STD_LOGIC;
        a   : IN    STD_LOGIC_VECTOR(24 DOWNTO 11);
        am  : IN    STD_LOGIC_VECTOR(5 DOWNTO 0);
        a_i : OUT   STD_LOGIC_VECTOR(24 DOWNTO 11);
        am_i    : OUT   STD_LOGIC_VECTOR(5 DOWNTO 0));
END addr_am_reg;

ARCHITECTURE rtl OF addr_am_reg IS
BEGIN

-- input register for addresses
    addr_reg: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 14)
  PORT MAP  (data => a,
   clock => clk,
   enable => en,
   q => a_i);

-- input register for address modifier
    am_reg: lpm_ff
  GENERIC MAP   (LPM_WIDTH => 6)
  PORT MAP  (data => am,
   clock => clk,
   enable => en,
   q => am_i);

END ARCHITECTURE rtl;
```

```
------------------------------------------------------------
--                                                        --
-- LOGIC CORE: GTL-module vme64x interface chip logic     --
-- MODULE NAME: addr_cnt                                  --
-- INSTITUTION: Hephy Vienna                              --
-- DESIGNER: H. Bergauer                                  --
--                                                        --
-- VERSION: V1.0                                          --
-- DATE: 08 2005                                          --
--                                                        --
-- FUNCTIONAL DESCRIPTION:                                --
-- addresses counter for BLT                              --
--                                                        --
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY lpm;
USE lpm.lpm_components.ALL;

ENTITY addr_cnt IS
    PORT(
        clk : IN    STD_LOGIC;
        cnt_en  : IN    STD_LOGIC;
        sclr    : IN    STD_LOGIC;
        aload   : IN    STD_LOGIC;
        sload   : IN    STD_LOGIC;
        ld_data : IN    STD_LOGIC_VECTOR(10 DOWNTO 1);
        out_data    : OUT   STD_LOGIC_VECTOR(10 DOWNTO 1));
END addr_cnt;

ARCHITECTURE rtl OF addr_cnt IS
BEGIN
-- default of updown is UP
-- this is an up-counter

inst_cnt: lpm_counter
   GENERIC MAP(LPM_WIDTH => 10,
      LPM_TYPE => "LPM_COUNTER")
   PORT MAP(data => ld_data,
        clock => clk,
        cnt_en => cnt_en,
        sclr => sclr,
        aload => aload,
        sload => sload,
        q => out_data);

END ARCHITECTURE rtl;
```

```
------------------------------------------------------------
--                                                        --
-- LOGIC CORE: GT-logic                                   --
-- MODULE NAME: rw_reg_8                                  --
-- INSTITUTION: Hephy Vienna                              --
-- DESIGNER: H. Bergauer                                  --
--                                                        --
-- VERSION: V1.0                                          --
-- DATE: 08 2005                                          --
--                                                        --
-- FUNCTIONAL DESCRIPTION:                                --
-- read/write register (8 bit)                            --
--                                                        --
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY lpm;
USE lpm.lpm_components.ALL;
LIBRARY altera;
USE altera.maxplus2.ALL;

ENTITY rw_reg_8 IS
    PORT(
        data    : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        wr_clk  : IN    STD_LOGIC;
        rd_en   : IN    STD_LOGIC;
        reg_out : OUT   STD_LOGIC_VECTOR(7 DOWNTO 0));
END rw_reg_8;

ARCHITECTURE rtl OF rw_reg_8 IS
    SIGNAL data_int : STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN

-- write register

write_reg: lpm_ff
  GENERIC MAP(LPM_WIDTH => 8)
    PORT MAP(data, wr_clk, q => data_int);

reg_out <= data_int;

-- read register (tri-state outputs to VME)
tri_stat_out:
FOR i IN 0 TO 7 GENERATE
 call_tri: tri
   PORT MAP(data_int(i), rd_en, data(i));
END GENERATE tri_stat_out;

END ARCHITECTURE rtl;
```

```
------------------------------------------------------------
--                                                        --
-- LOGIC CORE: GT logic                                   --
-- MODULE NAME: sel_test_outputs                          --
-- INSTITUTION: Hephy Vienna                              --
-- DESIGNER: H. Bergauer                                  --
--                                                        --
-- VERSION: V1.0                                          --
-- DATE: 08 2005                                          --
--                                                        --
-- FUNCTIONAL DESCRIPTION:                                --
-- selection of 1 of 16 signals for                       --
-- 4 test_out-pins (scope)                                --
--                                                        --
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY sel_test_outputs IS
    PORT(
--      test_signals_0      : IN    STD_LOGIC;
--      test_signals_1      : IN    STD_LOGIC;
--      test_signals_2      : IN    STD_LOGIC;
--      test_signals_3      : IN    STD_LOGIC;
--      test_signals_4      : IN    STD_LOGIC;
--      test_signals_5      : IN    STD_LOGIC;
--      test_signals_6      : IN    STD_LOGIC;
--      test_signals_7      : IN    STD_LOGIC;
--      test_signals_8      : IN    STD_LOGIC;
--      test_signals_9      : IN    STD_LOGIC;
--      test_signals_10     : IN    STD_LOGIC;
--      test_signals_11     : IN    STD_LOGIC;
--      test_signals_12     : IN    STD_LOGIC;
--      test_signals_13     : IN    STD_LOGIC;
--      test_signals_14     : IN    STD_LOGIC;
--      test_signals_15     : IN    STD_LOGIC;
        test_signals        : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
        en_1_signals        : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        en_2_signals        : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        en_3_signals        : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        en_4_signals        : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        test_outputs        : OUT   STD_LOGIC_VECTOR(4 DOWNTO 1)
        );
END sel_test_outputs;

ARCHITECTURE rtl OF sel_test_outputs IS
--COMPONENT test_out_coded IS
--  PORT(
--      en_signals          : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
--      test_signals        : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
--      test_out            : OUT   STD_LOGIC
--      );
--END COMPONENT test_out_coded;

--  SIGNAL test_signals_vec : STD_LOGIC_VECTOR(15 DOWNTO 0);

    TYPE en_signals_type IS ARRAY (4 DOWNTO 1)
        OF STD_LOGIC_VECTOR(3 DOWNTO 0);

    SIGNAL en_signals_arr : en_signals_type;
BEGIN
-- *****************************************************************
-- ERKLÄRUNG:
-- en_signals sind codiert, 4 bits aus VME-registers (2x8 bits für
-- 4 test_outputs mit je 16 test-signals), die angeben,
-- welches interne signal auf test-output gelegt wird.
-- test_signals ist der vector, der die internen signals enthält.
-- Korrespondierend mit dem value der en_signals wird das jeweilige
-- interne signal auf den test-output gelegt. Definition des internen
-- signals in vector notwendig!!!
-- ****************************************************************

en_signals_arr(4) <= en_4_signals;
en_signals_arr(3) <= en_3_signals;
en_signals_arr(2) <= en_2_signals;
en_signals_arr(1) <= en_1_signals;

--test_signals_vec <=
--  test_signals_15 & test_signals_14 & test_signals_13 & test_signals_12 &
--  test_signals_11 & test_signals_10 & test_signals_9 & test_signals_8 &
--  test_signals_7 & test_signals_6 & test_signals_5 & test_signals_4 &
--  test_signals_3 & test_signals_2 & test_signals_1 & test_signals_0;

loop_test_outputs:
    for i in 1 to 4 generate
        test_outputs(i) <= test_signals(CONV_INTEGER(en_signals_arr(i)));
    end generate loop_test_outputs;

--loop_test_outputs:
--for i in 1 to 4 generate
--  call_test_out: test_out_coded
--      PORT MAP(en_signals_arr(i), test_signals_vec, test_outputs(i));
--end generate loop_test_outputs;

END rtl;
```

```
-- ****************************************************************************
-- Specify values for addresses of Configuration ROM for VME64x
-- Only every forth address of CR table is used. D08_O = 1 and A01 = 1.

-- V100F:
-- function 1 and 3 from V1008 not implemented to reduce ressources
-- only functions for single transfer implemented!!
-- to use EP1K10 !!! HB130705

-- CR/CSR space definition:

-- VME64x_CR range: 0x03-0xFFF (0x03-0x7FF in this mif-file defined)
-- USER_CR range for chip_id, version and SN: 0x001003-0x00103F (see user_cr.mif)
-- CRAM range for future applications (not defined yet): 0x003003-0x0037FF
-- USER_CSR range for TEST_OUT-selection register: 0x005003-0x00502F
-- VME64x_CSR range: 0x7FC00-0x7FFFF

-- FUNCTION definition:

-- Function 0: D16 only, A31-A25, AM=0x0D and 0x09 (single transfer) for GT-system
-- Function 1: D16 only, A23-A18, AM=0x3D and 0x39 (single transfer) for DTTF-system
-- ****************************************************************************

DEPTH = 512;    % Memory depth and width are required   %
WIDTH = 8;  % Enter a decimal number    %

ADDRESS_RADIX = HEX;    % Address and value radixes are required    %
DATA_RADIX = HEX;   % Enter BIN, DEC, HEX, OCT, or UNS; unless  %
            % otherwise specified, radixes = HEX    %

-- used address-bits
-- A10 | A09 A08 A07 A06 | A05 A04 A03 A02

CONTENT
BEGIN
000          : 00; -- checksum [CR address = 0x03] -- to be defined!!
001          : 00; -- length of ROM (MSB, byte 2), not defined [CR address = 0x07] -- to be defined!!
002          : 00; -- length of ROM (byte 1), not defined [CR address = 0x0B]
003          : 00; -- length of ROM (LSB, byte 0), not defined [CR address = 0x0F]

004          : 81; -- CR data access width (0x81=D08(O)) [CR address = 0x13]
005          : 81; -- CSR data access width (0x81=D08(O)) [CR address = 0x17]
006          : 02; -- CR/CSR space specification ID (0x02=VME64x) [CR address = 0x13]
007          : 43; -- 0x43 (ASCII "C") [CR address = 0x1F]

008          : 52; -- 0x52 (ASCII "R") [CR address = 0x23]
```

```
009          : 00; -- Manufactor's ID (MSB, byte 2) (0x00=no IEEE code) [CR address = 0x27]
00A          : 00; -- Manufactor's ID (byte 1) (0x00=no IEEE code) [CR address = 0x2B]
00B          : 00; -- Manufactor's ID (LSB, byte 0) (0x00=no IEEE code) [CR address = 0x2F]

00C          : A0; -- Board ID (MSB, byte 3) (0xA0=example) [CR address = 0x33]
00D          : 12; -- Board ID (byte 2) (0x12=example) [CR address = 0x37]
00E          : 34; -- Board ID (byte 1) (0x34=example) [CR address = 0x3B]
00F          : 56; -- Board ID (LSB, byte 0) (0x56=example) [CR address = 0x3F]

010          : B9; -- Revision ID (MSB, byte 3) (0xB9=example) [CR address = 0x43]
011          : 87; -- Revision ID (byte 2) (0x87=example) [CR address = 0x47]
012          : 65; -- Revision ID (byte 1) (0x65=example) [CR address = 0x4B]
013          : 43; -- Revision ID (LSB, byte 0) (0x43=example) [CR address = 0x4F]

[014..01E]   : 00; -- not used! ("Pointer to null ..." and "RESERVED") [CR address = 0x53..0x7B]

01F          : 01; -- Programm ID (0x01=no program, ID code only) [CR address = 0x7F]

020          : 00; -- Offset to BEG_USER_CR (MSB, byte 2), (0x00) [CR address = 0x83] -- BEG_USER_CR = 0x001003
021          : 10; -- Offset to BEG_USER_CR (byte 1), (0x10) [CR address = 0x87]
022          : 03; -- Offset to BEG_USER_CR (LSB, byte 0), (0x03) [CR address = 0x8B]
023          : 00; -- Offset to END_USER_CR (MSB, byte 2), (0x00) [CR address = 0x8F] -- END_USER_CR = 0x00103F

024          : 10; -- Offset to END_USER_CR (byte 1), (0x10) [CR address = 0x93]
025          : 3F; -- Offset to END_USER_CR (LSB, byte 0), (0x1F) [CR address = 0x97]
026          : 00; -- Offset to BEG_CRAM (MSB, byte 2), (0x00) [CR address = 0x9B] -- BEG_CRAM = 0x003003
027          : 30; -- Offset to BEG_CRAM (byte 1), (0x30) [CR address = 0x9F]

028          : 03; -- Offset to BEG_CRAM (LSB, byte 0), (0x03) [CR address = 0xA3]
029          : 00; -- Offset to END_CRAM (MSB, byte 2), (0x00) [CR address = 0xA7] -- END_CRAM = 0x0037FF
02A          : 37; -- Offset to END_CRAM (byte 1), (0x37) [CR address = 0xAB]
02B          : FF; -- Offset to END_CRAM (LSB, byte 0), (0xFF) [CR address = 0xAF]

02C          : 00; -- Offset to BEG_USER_CSR (MSB, byte 2), (0x00) [CR address = 0xB3] -- BEG_USER_CSR = 0x005003
02D          : 50; -- Offset to BEG_USER_CSR (byte 1), (0x50) [CR address = 0xB7]
02E          : 03; -- Offset to BEG_USER_CSR (LSB, byte 0), (0x03) [CR address = 0xBB]
02F          : 00; -- Offset to END_USER_CSR (MSB, byte 2), (0x00) [CR address = 0xBF] -- END_USER_CSR = 0x00502F

030          : 50; -- Offset to END_USER_CSR (byte 1), (0x50) [CR address = 0xC3]
031          : 2F; -- Offset to END_USER_CSR (LSB, byte 0), (0x2F) [CR address = 0xC7]
032          : 00; -- Offset to BEG_SN (MSB, byte 2), (0x00) [CR address = 0xCB] -- BEG_SN = 0x001023
033          : 10; -- Offset to BEG_SN (byte 1), (0x10) [CR address = 0xCF]

034          : 23; -- Offset to BEG_SN (LSB, byte 0), (0x23) [CR address = 0xD3]
035          : 00; -- Offset to END_SN (MSB, byte 2), (0x00) [CR address = 0xD7] -- BEG_SN = 0x001037
036          : 10; -- Offset to END_SN (byte 1), (0x10) [CR address = 0xDB]
```

```
037           : 37; -- Offset to END_SN (LSB, byte 0), (0x2F) [CR address = 0xDF]

038           : 00; -- Slave characteristics parameter (0x00, see Table 10-1 VME64x spec.) [CR address = 0xE3]
039           : 00; -- User defined slave characteristics, not used! [CR address = 0xE7]
03A           : 00; -- Master characteristics, not used! [CR address = 0xEB]
03B           : 00; -- User defined master characteristics, not used! [CR address = 0xEF]

03C           : 00; -- Interrupt handler capabilities, not used! [CR address = 0xF3]
03D           : 00; -- Interrupter capabilities, not used! [CR address = 0xF7]
03E           : 00; -- Reserved, not used! [CR address = 0xFB]
03F           : 00; -- CRAM_ACCESS_WIDTH (0x81=D08(O)) [CR address = 0xFF]

040           : 83; -- Function 0 DAWPR  (0x83, "D16 only (C. Schwick!)", see Table 10-3 VME64x spec.) [CR address = 0x103]
041           : 83; -- Function 1 DAWPR  (0x83, "D16 only (C. Schwick!)", see Table 10-3 VME64x spec.)[CR address = 0x107]
042           : 00; -- Function 2 DAWPR  (0x00, "feature not implemented", see Table 10-3 VME64x spec.) [CR address = 0x10B]
043           : 00; -- Function 3 DAWPR  (0x00, "feature not implemented", see Table 10-3 VME64x spec.)[CR address = 0x10F]

044           : 00; -- Function 4 DAWPR  (0x00, "feature not implemented", see Table 10-3 VME64x spec.) [CR address = 0x113]
045           : 00; -- Function 5 DAWPR  (0x00, "feature not implemented", see Table 10-3 VME64x spec.) [CR address = 0x117]
046           : 00; -- Function 6 DAWPR  (0x00, "feature not implemented", see Table 10-3 VME64x spec.) [CR address = 0x11B]
047           : 00; -- Function 7 DAWPR  (0x00, "feature not implemented", see Table 10-3 VME64x spec.) [CR address = 0x11F]

048           : 00; -- Function 0 AMCAP  (MSB, byte 7) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x123]
049           : 00; -- Function 0 AMCAP  (byte 6) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x127]
04A           : 00; -- Function 0 AMCAP  (byte 5) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x12B]
04B           : 00; -- Function 0 AMCAP  (byte 4) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x12F]

04C           : 00; -- Function 0 AMCAP  (byte 3) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x133]
04D           : 00; -- Function 0 AMCAP  (byte 2) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x137]
04E           : 22; -- Function 0 AMCAP  (byte 1) (0x22, AM=0x0D and 0x09, see 10.2.1.4.2 VME64x spec.) [CR address = 0x13B]
04F           : 00; -- Function 0 AMCAP  (LSB, byte 0) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x13F]

050           : 44; -- Function 1 AMCAP  (MSB, byte 7) (0x22, AM=0x3D and 0x39, see 10.2.1.4.2 VME64x spec.) [CR address = 0x143]
051           : 00; -- Function 1 AMCAP  (byte 6) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x147]
052           : 00; -- Function 1 AMCAP  (byte 5) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x14B]
053           : 00; -- Function 1 AMCAP  (byte 4) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x14F]

054           : 00; -- Function 1 AMCAP  (byte 3) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x153]
055           : 00; -- Function 1 AMCAP  (byte 2) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x157]
056           : 00; -- Function 1 AMCAP  (byte 1) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x15B]
057           : 00; -- Function 1 AMCAP  (LSB, byte 0) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x15F]

058           : 00; -- Function 2 AMCAP  (MSB, byte 7) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x163]
059           : 00; -- Function 2 AMCAP  (byte 6) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x167]
05A           : 00; -- Function 2 AMCAP  (byte 5) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x16B]
05B           : 00; -- Function 2 AMCAP  (byte 4) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x16F]
```

```

05C           : 00; -- Function 2 AMCAP  (byte 3) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x173]
05D           : 00; -- Function 2 AMCAP  (byte 2) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x177]
05E           : 00; -- Function 2 AMCAP  (byte 1) (0x22, see 10.2.1.4.2 VME64x spec.) [CR address = 0x17B]
05F           : 00; -- Function 2 AMCAP  (LSB, byte 0) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x17F]

060           : 00; -- Function 3 AMCAP  (MSB, byte 7) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x183]
061           : 00; -- Function 3 AMCAP  (byte 6) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x187]
062           : 00; -- Function 3 AMCAP  (byte 5) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x18B]
063           : 00; -- Function 3 AMCAP  (byte 4) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x18F]

064           : 00; -- Function 3 AMCAP  (byte 3) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x193]
065           : 00; -- Function 3 AMCAP  (byte 2) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x197]
066           : 00; -- Function 3 AMCAP  (byte 1) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x19B]
067           : 00; -- Function 3 AMCAP  (LSB, byte 0) (0x00, see 10.2.1.4.2 VME64x spec.) [CR address = 0x19F]

[068..187]    : 00;  -- not used! ("Function 4 - 7 AMCAP" and "Function 0 - 7 XAMCAP") [CR address = 0x1A3..0x61F]

188           : FE; -- Function 0 ADEM  (MSB, byte 3) (0xFE, "mask bits 31-25=1, mask bits 24=0", see Table 10-4 VME64x spec.) [CR a
189           : 00; -- Function 0 ADEM  (byte 2) (0x00, "mask bits 23-16=0", see Table 10-4 VME64x spec.) [CR address = 0x627]
18A           : 00; -- Function 0 ADEM  (byte 1) (0x00, "mask bits 15-8=0", see Table 10-4 VME64x spec.) [CR address = 0x62B]
18B           : 00; -- Function 0 ADEM  (LSB, byte 0) (0x00, "special bits=0", see Table 10-4 VME64x spec.) [CR address = 0x62F]

18C           : FF; -- Function 1 ADEM  (MSB, byte 3) (0xFF, "mask bits 31-24=1", see Table 10-4 VME64x spec.) [CR address = 0x633]
18D           : FC; -- Function 1 ADEM  (byte 2) (0xFC, "mask bits 23-18=1, mask bits 17-16=0", see Table 10-4 VME64x spec.) [CR add
18E           : 00; -- Function 1 ADEM  (byte 1) (0x00, "mask bits 15-8=0", see Table 10-4 VME64x spec.) [CR address = 0x63B]
18F           : 00; -- Function 1 ADEM  (LSB, byte 0) (0x00, "special bits=0", see Table 10-4 VME64x spec.) [CR address = 0x63F]

190           : 00; -- Function 2 ADEM  (MSB, byte 3) (0x00, "mask bits 31-24=0", see Table 10-4 VME64x spec.) [CR address = 0x623]
191           : 00; -- Function 2 ADEM  (byte 2) (0x00, "mask bits 23-16=0", see Table 10-4 VME64x spec.) [CR address = 0x627]
192           : 00; -- Function 2 ADEM  (byte 1) (0x00, "mask bits 15-8=0", see Table 10-4 VME64x spec.) [CR address = 0x62B]
193           : 00; -- Function 2 ADEM  (LSB, byte 0) (0x00, "special bits=0", see Table 10-4 VME64x spec.) [CR address = 0x62F]

194           : 00; -- Function 3 ADEM  (MSB, byte 3) (0x00, "mask bits 31-24=0", see Table 10-4 VME64x spec.) [CR address = 0x633]
195           : 00; -- Function 3 ADEM  (byte 2) (0x00, "mask bits 23-16=0", see Table 10-4 VME64x spec.) [CR address = 0x637]
196           : 00; -- Function 3 ADEM  (byte 1) (0x00, "mask bits 15-8=0", see Table 10-4 VME64x spec.) [CR address = 0x63B]
197           : 00; -- Function 3 ADEM  (LSB, byte 0) (0x00, "special bits=0", see Table 10-4 VME64x spec.) [CR address = 0x63F]

[198..1AA]    : 00; -- not used! ("Function 4 - 7 ADEM" and "reserved, read as zero") [CR address = 0x643..0x6AB]

1AB           : 00; -- Master data access width (0x00, "feature not implemented", see Table 10-3 VME64x spec.) [CR address = 0x6AF]

[1AC..1D3]    : 00; -- not used! ("Master AMCAP" and "Master XAMCAP") [CR address = 0x6B3..0x74F]
[1D4..1FF]    : 00; -- not used! ("RESERVED") [CR address = 0x753..0x7FF]

END;
```

□

```
-- ***************************************************************************
-- Specify values for addresses of User Configuration ROM for VME64x of TIM-6U_V2-card
-- Only every forth address is used. D08_O = 1 and A01 = 1.
-- USER_CR range for chip_id, version and SN (VME64x): 0x001003-0x001037
-- chip_id: 0x0001B011 => 0 for card_nr, card_nr comes in hardware from S31-S28!!!!
-- see user_cr.vhd !!!
-- version: 0x0000100F
-- ***************************************************************************

DEPTH = 16; % Memory depth and width are required   %
WIDTH = 8;  % Enter a decimal number    %

ADDRESS_RADIX = HEX;    % Address and value radixes are required    %
DATA_RADIX = HEX;   % Enter BIN, DEC, HEX, OCT, or UNS; unless  %
            % otherwise specified, radixes = HEX    %

-- used address-bits
-- A05 A04 A03 A02

CONTENT
BEGIN
0           : 00; -- chip_id-register 3 (MSB) [USER_CR address = 0x001003] 
1           : 01; -- chip_id-register 2 [USER_CR address = 0x001007] 
2           : B0; -- chip_id-register 1 [USER_CR address = 0x00100B] 
3           : 11; -- chip_id-register 0 [USER_CR address = 0x00100F] 

4           : 00; -- version-register 3 (MSB) [USER_CR address = 0x001013] 
5           : 00; -- version-register 2 [USER_CR address = 0x001017] 
6           : 10; -- version-register 1 [USER_CR address = 0x00101B] 
7           : 0F; -- version-register 0 [USER_CR address = 0x00101F] 

8           : 54; -- serial number byte 7 (MSB), ASCII "T" [SN address = 0x001023] 
9           : 49; -- serial number byte 6, ASCII "I" [SN address = 0x001027]
A           : 4D; -- serial number byte 5, ASCII "M" [SN address = 0x00102B]
B           : 5F; -- serial number byte 4, ASCII "_" [SN address = 0x00102F]

C           : 56; -- serial number byte 3, ASCII "V" [SN address = 0x001033] 
D           : 32; -- serial number byte 2, ASCII "2" [SN address = 0x001037]

[E..F]      : 00; -- not used! [CR address = 0x00103B-0x00103F]

END;
```

VME-CHIP
(Version 0x1009)

of

TIM-6U_V2-card
(6U-Version)

H. Bergauer, K. Kastner, M. Padrta, A. Taurok

Jan-06

Version 0x1009

1 Introduction

The VME chip TIM_V2 works with the VME64x chip as controller for the VME-bus of the TIM-6U_V2-card. There are VME-registers on it as the Registers for Programmable-chips-configuration, General registers, Chip ID / version registers and JTAG registers (for details see “VME chip register”). The VME-accesses to the TIM-chip are made via this chip too.

2 VME chip of TIM-6U_V2-card

2.1 Versionshistory

- V1000: first design, A19 and A18 are not used in the decoder, but error in general-command-register. **DO NOT USE!!** (HB050805).
- V1001: A19 and A18 are used in the decoder, causes problems with baseaddress for DTTF-system. **DO NOT USE!!** (HB050805).
- V1002: based on V1000, but with new design for general-command-register. Error at RESET_MODE. **DO NOT USE!!** (HB110805).
- V1003: based on V1002, but RESET_MODE error corrected. **DO NOT USE!!** (HB110805).
- V1004: based on V1002, but LOCKED_LED only by TIM_LOCKED. **DO NOT USE!!** (HB110805).
- V1005: based on V1004, but write/read for general-register and configuration-register implemented. **DO NOT USE!!** (HB170805).
- V1006: based on V1005, but DSCYC for LED_DELAY used, to prevent VME-access indication of address-spikes on the end of VME transfer. [VIEWDRAW: new styled symbols from P:\Lab3Lib\..\vme_chip_lib copied to local working directory] (HB240805).
- V1007: based on V1006, but lines ASCYC, ASSYNC, ASPULS and D08_E from VME64x-chip represent the CARD_NR[3:0] – from jumpers S27-S24 (HB310805).
- V1008: based on V1007, but DSSYNC (with one clock delay) for read/write of TIM-chip [VIEWDRAW: library P:\Lab3Lib\..\vme_chip_lib used] (HB140905).
- **V1009:** based on V1008, but DTACK_EXT and BERR_EXT are used as negative active signals to VME64x-chip (because at power-up configuration of VME64X-CHIP is faster than configuration of VME-CHIP and therefore wrong DTACK and BERR signals are generated after configuration, which causes LEDs=”on” of CAEN-controller). JTAG_CTRL V1.5 implemented, because of Quartus 5.1 and EN_JTAG is delayed to have proper setup-time for addresses, because of syncronous version of VME64x-chip. Lines ASCYC, ASSYNC, ASPULS and D08_E from VME64x-chip represent the CARD_NR[3:0] – from jumpers S31-S28. (HB050106).

2.2 Hardware

The VME64x-chip is an Altera EP1K100QC208-3.

2.3 Firmware

chip_id:	`0x0001Bn21`	`(n = CARD_NR from jumpers S31 – S28)`
version:	`0x00001009`	

2.4 VME access

2.4.1 Base address

Base address of all GT-slaves is encoded on A31-A25 (A24 not used), because of address space of GTL-6U-card. See definition in VME64x-chip for TIM_V2. The DTTF-system works with base address on A23-A18.

2.4.2 AM and datatransfer

GT-system:
AM=0x0D and 0x09 „extended data access“ - for single access.
D16 „word access” - for all accesses.
A19 and A18 are 0, set by software during access to onboard-registers.
DTTF-system:
AM=0x3D and 0x39 „standard data access“ - for single access.
D16 „word access” - for all accesses.
See definitions in VME64x-chip for TIM_V2.

2.5 Chip selection on TIM-6U_V2-card

With the VME addresses A17-A16 the chip selection is done on the TIM-6U_V2-card.

A17	A16	Chip-name
0	0	VME chip
0	1	TIM-chip

GT-system:

A31-A24	=>	Base address – extended access modes
A23-A20	=>	not used
A19-A18	=>	not used (set to 0 by software during access to onboard-registers)
A17-A00	=>	Register-addresses

DTTF-system:

A31-A24	=>	not used – standard access modes
A23-A18	=>	Base address
A17-A00	=>	Register-addresses

2.6 VME chip register

Register for Programmable-chips-configuration:			
A31..A24/A23..A18	A17..A16	A15..A06	A05..A01,(00)
8/6 bits		10bits	5bits
Base address	00	0000000000	Registers

2.6.1 VME chip address-table

The address-table lists the address-offset which has to be combined with the base-address of the card. The **address-offset** is **valid** for the **GT-system** only, for the DTTF-system the address-offset is different, because of the address-lines of the base address.

A23-A00	**=>**	**Register-name**
Register for Programmable-chips-configuration:		
0x000000	=>	CMD_ENPROG-register (write/read)
0x000002	=>	CMD_NPROG-register (write/read)
0x000004	=>	CMD_INIT-register (write/read)
0x000006	=>	STAT_INIT-register (read)
0x000008	=>	STAT_DONE-register (read)
0x00000A	=>	Configuration register TIM-chip (write)
General pulse registers:		
0x000010	=>	Command pulse register (write)
0x000012	=>	Status pulse register (read)
General registers:		
0x000014	=>	Command register (write/read)
0x000016	=>	Status register (read)
TTC-control registers:		
0x000018	=>	TTC control register (write/read)
TEST-OUT registers:		
0x00001A	=>	TEST-OUT-10 register (write/read)
0x00001C	=>	TEST-OUT-32 register (write/read)
Chip ID and version registers:		
0x000020	=>	chip_id_register_3 (read)
0x000022	=>	chip_id_register_2 (read)
0x000024	=>	chip_id_register_1 (read)
0x000026	=>	chip_id_register_0 (read)
0x000028	=>	version_register_3 (read)
0x00002A	=>	version_register_2 (read)
0x00002C	=>	version_register_1 (read)
0x00002E	=>	version_register_0 (read)
JTAG registers:		
0x000030	=>	tdo_register (write/read)
0x000032	=>	tdi_register (write/read)
0x000034	=>	tms0_register (write/read)
0x000036	=>	tms1_register (write/read)
0x000038	=>	cnt32_register (write/read)
0x00003A	=>	mode0_register (write/read)
0x00003C	=>	mode1_register (write/read)
0x00003E	=>	mode2_register (write/read)
Access to/from TIM-chip (in GT-system) :		
0x01XXXX	=>	see TIM-chip

2.6.2 Register for Programmable-chips-configuration

The TIM-chip (Virtex-II) is configurable by configuration device and by VMEbus instructions. The selection is made by jumpers.

Register names	D7..D1	D0
CMD_ENPROG	-	ENPROG_TIM
CMD_NPROG	-	NPROG_TIM
CMD_INIT	-	INIT_TIM
STAT_INIT	-	INIT_TIM
STAT_DONE	-	DONE_TIM
CONF_TIM	-	configuration data

2.6.2.1 CMD_ENPROG-register

0x000000 => CMD_ENPROG-register (write/read)
Bit 0 of the CMD_ENPROG-register allows sending the configuration bits via VME-bus to the TIM-chip.

2.6.2.2 CMD_NPROG-register

0x000002 => CMD_NPROG-register (write/read)
Data-bit 0 = 1 of this register set the NPROG-signal of TIM-chip active. Then it should be reset to 0. Then the TIM-chip enters into the configuration procedure. The FPGA either waits for configuration data (slave mode) sent via VME or starts to read configuration bits from a serial PROM (master mode).

2.6.2.3 CMD_INIT-register

0x000004 => CMD_INIT-register (write/read)
Data-bit 0 = 1 of this register set the NINIT-signal of TIM-chip active.

2.6.2.4 STAT_INIT-register

0x000006 => STAT_INIT-register (read)
Read the status of the NINIT-signal of TIM-chip (data-bit 0)

2.6.2.5 STAT_DONE-register

0x000008 => STAT_DONE-register (read)
Read the status of the DONE-signal of TIM-chip (data-bit 0). After a successful configuration the TIM-chip sets DONE =1.

2.6.2.6 Configuration register TIM-chip

0x00000A => Configuration register TIM-chip (write)
The register is used to load the configuration bits into the TIM-chip (Virtex-II).
A write access to this register generates a CCLK and sends the data-bit 0 as DIN-signal to the TIM-chip, if the CMD_ENPROG-register bit has been set before. The VME accesses are repeated until the last bit has been loaded into the TIM-chip.

2.6.3 General pulse registers

Register names	**D3**	**D2**	**D1**	**D0**
Command_Pulse _Reg	SET_RUNNING (pulse)	RES_TIMM (pulse*)	RES_DCM_ TIMM (pulse)	PWRDWN_ TIM (pulse)
Status_Pulse_Reg	RUNNING	LOCKED_ LED	TIM_LOCKED	not used

*) also generated by RESET_MODE

Register names	**D7**	**D6**	**D5**	**D4**
Command_Pulse _Reg	not used	not used	not used	not used
Status_Pulse_Reg	not used	not used	TTC_ERROR	TTC_LOCKED

2.6.3.1 Command pulse register

`0x000010` => **Command-pulse-register (write)**

D0: **PWRDWN_TIM = 1** sends a low active pulse to the TIM-chip setting it into power down mode. NPWRDWN_B is sent as an open drain signal from the VME-TIM_V2-chip to the TIM-chip.
Remark from data sheet:
The power-down sequence enables a designer to set the device into a low-power, inactive state. The sequence is initiated by pulling the PWRDWN_B pin Low. To monitor power-down status, observe the PWRDWN_B pin. When asserted, power-down has completed. After a successful wake-up, the status pin de-asserts. While powered down, the only active pins are the PWRDWN_B and DONE. All inputs are off and all outputs are 3-stated. While in the POWERDOWN state, the Power On Reset (POR) circuit is still active, but it does not reset the device if V_{CCINT}, V_{CCO}, or V_{CCAUX} falls below its minimum value. The POR circuit waits until the PWRDWN_B pin is released before resetting the device. Also, the PROG_B pin is not sampled while the device is in the POWERDOWN state. The PROG_B pin becomes active when the PWRDWN_B pin is released. Therefore, the device cannot be reset while in the POWERDOWN state. The wake-up sequence is the reverse of the power-down sequence.

D1: **RES_DCM_TIMM = 1** sends a high active pulse to the TIM-chip, to forces the DCM module to lock.

D2: **RES_TIMM = 1** sends a high active pulse to the TIM-chip, to reset logic.

D3: **SET_RUNNING = 1** sends a high active pulse to set board in RUNNING mode.

D7-D4: not used.

2.6.3.2 Status pulse register

`0x000012` => **Status-pulse-register (read)**

D0: not used.

D1: **TIM_LOCKED = 1** indicates, that the DCM module of the TIM chip is locked to the 40 MHz clock.
This status bit has to be checked immediately after the configuration of the TIM chip and before any other actions.

D2: **LOCKED_LED** is the status of TIM_LOCKED-signals (TTC_LOCKED is not included in LOCKED_LED).

D3: **RUNNING = 1** board is active.
If RUNNING = 0, send a SET_RUNNING command via VME.

D4: **TTC_LOCKED = 1** QPLL of TTCrq-module has locked.

D5: **TTC_ERROR = 1** TTCrq-module has an error.
D7-D6: not used.

2.6.4 General registers

Register names	**D3**	**D2**	**D1**	**D0**
Command_Reg	SEL_ CABLES	VME_CONF	not used	CROSS_SWITCH
Status_Reg	not used	STATUS_SEL_ VME	not used	not used

Register names	**D7**	**D6**	**D5**	**D4**
Command_Reg	not used	not used	not used	SEL_ BACKPL
Status_Reg	not used	DTTF_MODE	JTAG_JUMPER	not used

2.6.4.1 Command register

`0x000014` => **Command-register (write/read)**

D0: **CROSS_SWITCH = 1** selects differential TTC-clock to PLL and PLL-clock to backplane.
CROSS_SWITCH = 0 selects differential TTC-clock to PLL and to backplane
D1: not used.
D2: **VME_CONF = 1** enables configuration of TIM-chip via VME and switches external mux from PROM to VME.
D3: **SEL_CABLES = 1** switches JTAG-chains to cables (MasterBlaster and Parallel-Cable-IV).
D4: **SEL_BACKPL = 1** switches JTAG-chains to backplane connection via SCANPSC110 (if SEL_CABLES = 0).

Truthtable for D4 and D3:

D4	D3	
0	0	JTAG-chains via VME
X	1	JTAG-chains via cables
1	0	JTAG-chains via backplane

2.6.4.2 Status register

`0x000016` => **Status-register (read)**

D0: not used.
D1: not used.
D2: **STATUS_SEL_VME = 1** indicates, that configuration of TIM-chip via VME is selected.
For configuration of TIM-chip via VME, set VME_CONF = 1 in the Command-register.
D3: not used.
D4: not used.
D5: **JTAG_JUMPER = 1** indicates, that SEL_CABLE_JTAG-jumper (JP50) is inserted. Therefore JTAG-chains are connected to cables (MasterBlaster and Parallel-Cable-IV).
For changing the sources of JTAG-chains, remove the jumper and make the selection with SEL_CABLES and SEL_BACKPL in the Command-register.
D6: **DTTF_MODE = 1** operation in DTTF-system is selected by jumper.

(DTTF_MODE = 0 operation in GT-system is selected).

2.6.5 TTC-control register

The TTC-control register is used to setup the QPLL of the TTCrq-module.

Register names	D3	D2	D1	D0
TTC_control	F0SELECT3	F0SELECT2	F0SELECT1	F0SELECT0

Register names	D7	D6	D5	D4
TTC_control	EXT_CTRL	MODE	NRESET (pulse) / F0SELECT5	AUTORESTART / F0SELECT4

2.6.5.1 TTC control register

`0x000018` => **TTC-control-register (write/read)**

D3-D0: **F0SELECT[3:0]** these signals (including F0SELECT[5:4]) control the VCXO free running oscillation frequency when the signal "externalControl" (EXT_CTRL) is set to "1". If "externalControl" is set to "0" these signals have no influence on the operation of the QPLL.

D4: **AUTORESTART / F0SELECT4**
If "externalControl" = 0, **AUTORESTART = 0** automatic restart of the PLL is disbled, a frequency calibration will only occur after a reset.
If "externalControl" = 0, **AUTORESTART = 1** automatic restart of the PLL is enabled, a frequency calibration will only occur each time the PLL is detected to be unlocked or after a reset.
If "externalControl" = 1, **AUTORESTART** becomes **F0SELECT4**.

D5: **NRESET / F0SELECT5**
If "externalControl" = 0, active low reset signal. It initiates a frequency calibration cycle and lock acquisition.
If "externalControl" = 1, **NRESET** becomes **F0SELECT5**.

D6: **MODE = 0** 120 MHz frequency multiplication mode (120 MHz crystal required).
MODE = 1 160 MHz frequency multiplication mode (160 MHz crystal required).

D7: **EXT_CTRL** ("externalControl") **= 0** the VCXO centre frequency is set by automatic frequency calibration procedure.
EXT_CTRL = 1 the VCXO free running frequency is set by the signals F0SELECT[5:0].

Examples for combination of D7 and D5:

- Reading from TTC-control-register after power-up gives 0x20, because D5 (NRESET is inactive) is '1' and D7 is '0'.
- Writing 0x20 to TTC-control-register causes a negative pulse on D5 (NRESET pulse).
- Writing 0x80 to TTC-control-register enables F0SELECT5 on D5.

2.6.6 TEST-OUT registers

The TEST-OUT registers select the signals for the four testoutputs (TEST_P[3:0]). In the register there is to set the 4-bit code for 16 selectable internal signals for each testoutput. See the following table:

TEST_Px_[3:0]	selected signal
0000	V_A1

0001	V_A2
0010	V_A3
0011	V_A4
0100	V_A5
0101	V_A16
0110	V_A17
0111	CMD_PULSE_GEN
1000	STAT_PULSE_GEN
1001	WR_CMD_GEN
1010	RD_STAT_GEN
1011	WR_TTC_CTRL
1100	WR_TEST_OUT_10
1101	WR_TEST_OUT_32
1110	EN_JTAG
1111	DTACK_JTAG

2.6.6.1 TEST-OUT-10 register

Register names	**D3**	**D2**	**D1**	**D0**
TEST-OUT-10	TEST_P0_3	TEST_P0_2	TEST_P0_1	TEST_P0_0

Register names	**D7**	**D6**	**D5**	**D4**
TEST-OUT-10	TEST_P1_3	TEST_P1_2	TEST_P1_1	TEST_P1_0

0x00001A =>　TEST-OUT-10 register (write/read)

2.6.6.2 TEST-OUT-32 register

Register names	**D3**	**D2**	**D1**	**D0**
TEST-OUT-32	TEST_P2_3	TEST_P2_2	TEST_P2_1	TEST_P2_0

Register names	**D7**	**D6**	**D5**	**D4**
TEST-OUT-32	TEST_P3_3	TEST_P3_2	TEST_P3_1	TEST_P3_0

0x00001C =>　TEST-OUT-32 register (write/read)

2.6.7 Chip ID and version registers

2.6.7.1 Definitions

Chip_id_register and version_register have fixed values in the hardware. These registers have read access only.

The versions `0x00000000 - 0x00000FFF` are used for tests.
The versions `0x00001000 - 0xFFFFFFFF` are used for runs in CMS.

2.6.7.2 Settings

TIM-6U_V2-card:

chip_id:	`0x0001B021 (CARD_NR comes from jumpers S27-S24)`
version:	`0x00001008`

2.6.7.3 Chip ID and version registers addresses

A23-A00 => Register-name

`0x000020` => chip_id_register_3 (read)

D7	D6	D5	D4	D3	D2	D1	D0
chip_ID [31..24]							

`0x000022` => chip_id_register_2 (read)

D7	D6	D5	D4	D3	D2	D1	D0
chip_ID [23..16]							

`0x000024` => chip_id_register_1 (read)

D7	D6	D5	D4	D3	D2	D1	D0
chip_ID [15..08]							

`0x000026` => chip_id_register_0 (read)

D7	D6	D5	D4	D3	D2	D1	D0
chip_ID [07..00]							

`0x000028` => version_register_3 (read)

D7	D6	D5	D4	D3	D2	D1	D0
version [31..24]							

`0x00002A` => version_register_2 (read)

D7	D6	D5	D4	D3	D2	D1	D0
version [23..16]							

`0x00002C` => version_register_1 (read)

D7	D6	D5	D4	D3	D2	D1	D0
version [15..08]							

`0x00002E` => version_register_0 (read)

D7	D6	D5	D4	D3	D2	D1	D0
version [07..00]							

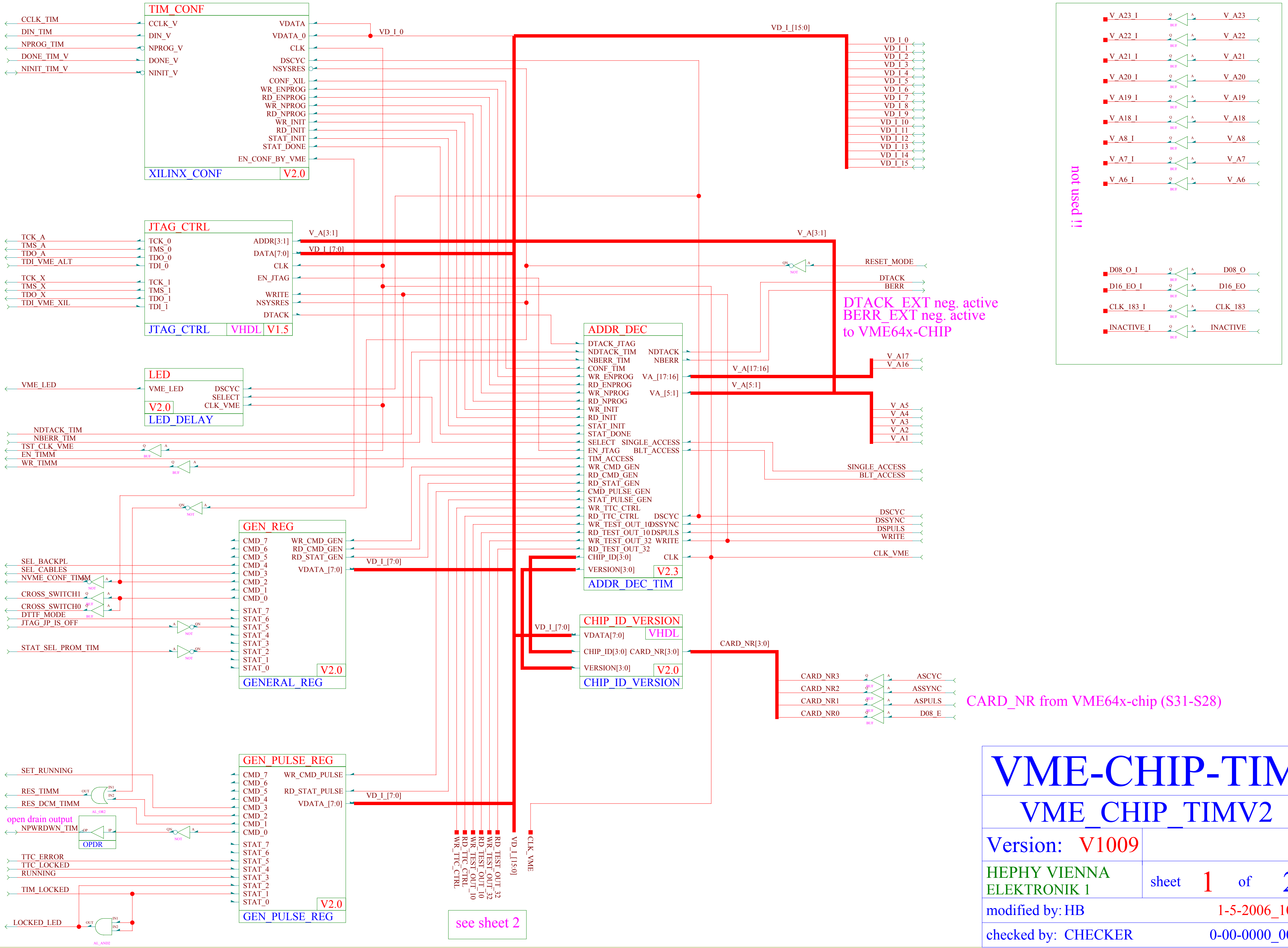
2.6.8 JTAG-registers

2.6.8.1 Definitions

JTAG registers are used to control JTAG-chains via VME-bus.
For details see JTAGController.vhd from Hannes Sakulin.

2.7 DTACK/BERR-generation

Writing to writeable registers and reading from readable registers generates a DTACK signal.
Access to/from TIM-chip generates a DTACK signal.
A BERR signal is generated only from the TIM-chip!



VME-CHIP-TIM

VME_CHIP_TIMV2

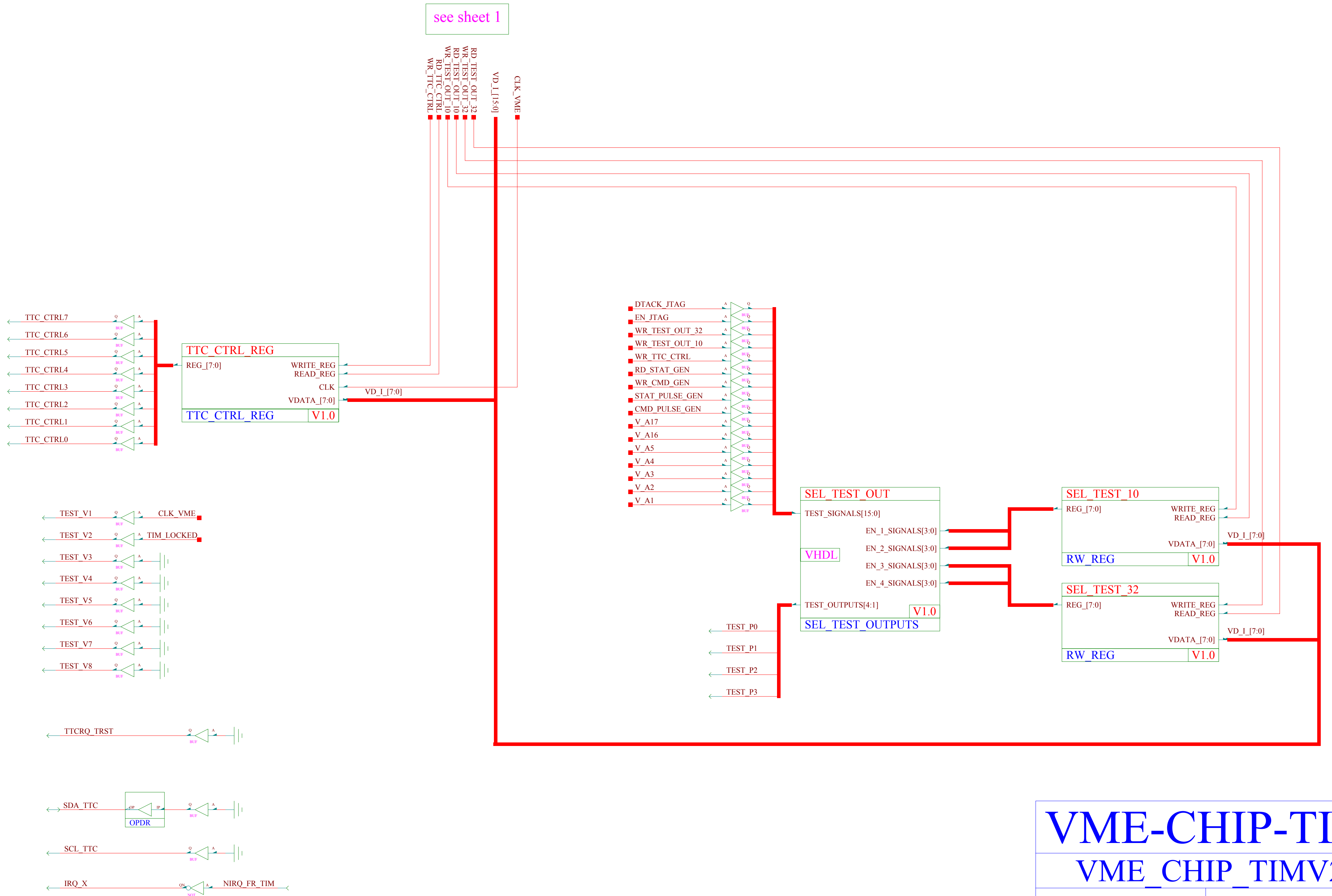
Version: V1009

HEPHY VIENNA
ELEKTRONIK 1

sheet 1 of 2

modified by: HB 1-5-2006_10:02

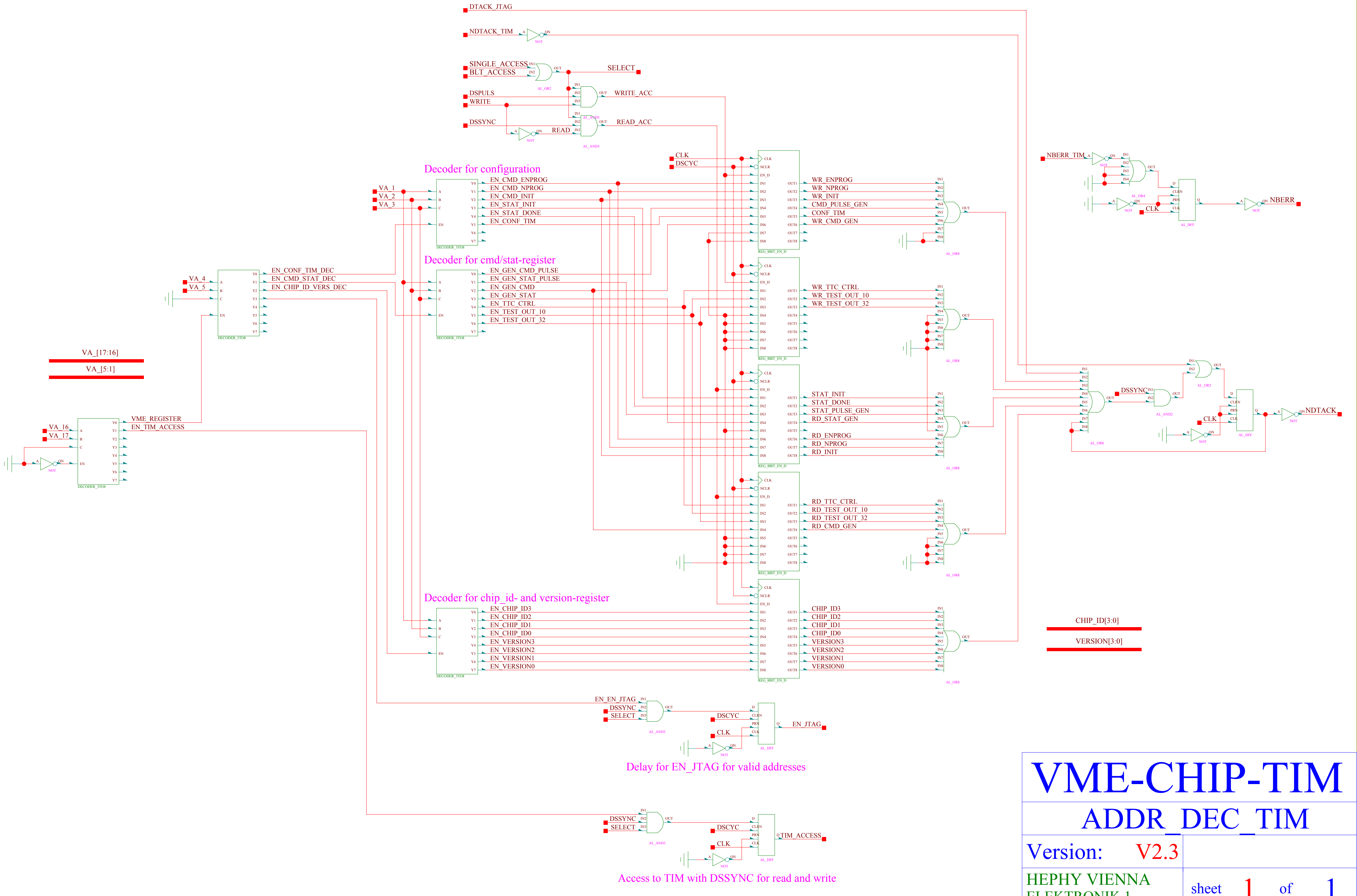
checked by: CHECKER 0-00-0000_00:00



VME-CHIP-TIM

VME_CHIP_TIMV2

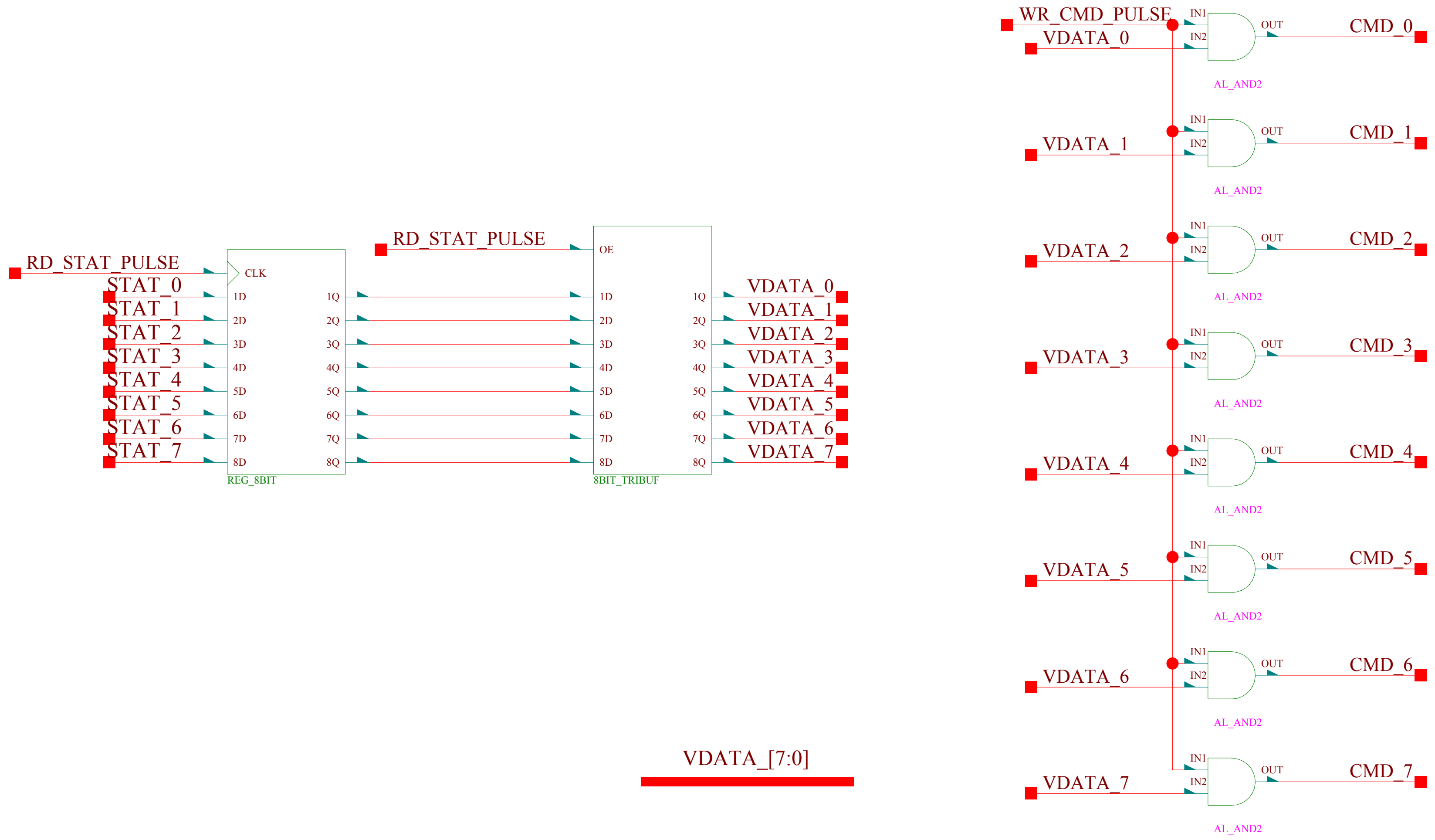
Version: V1009	
HEPHY VIENNA ELEKTRONIK 1	sheet 2 of 2
modified by: HB	12-19-2005_11:53
checked by: CHECKER	0-00-0000_00:00



VME-CHIP-TIM

ADDR_DEC_TIM

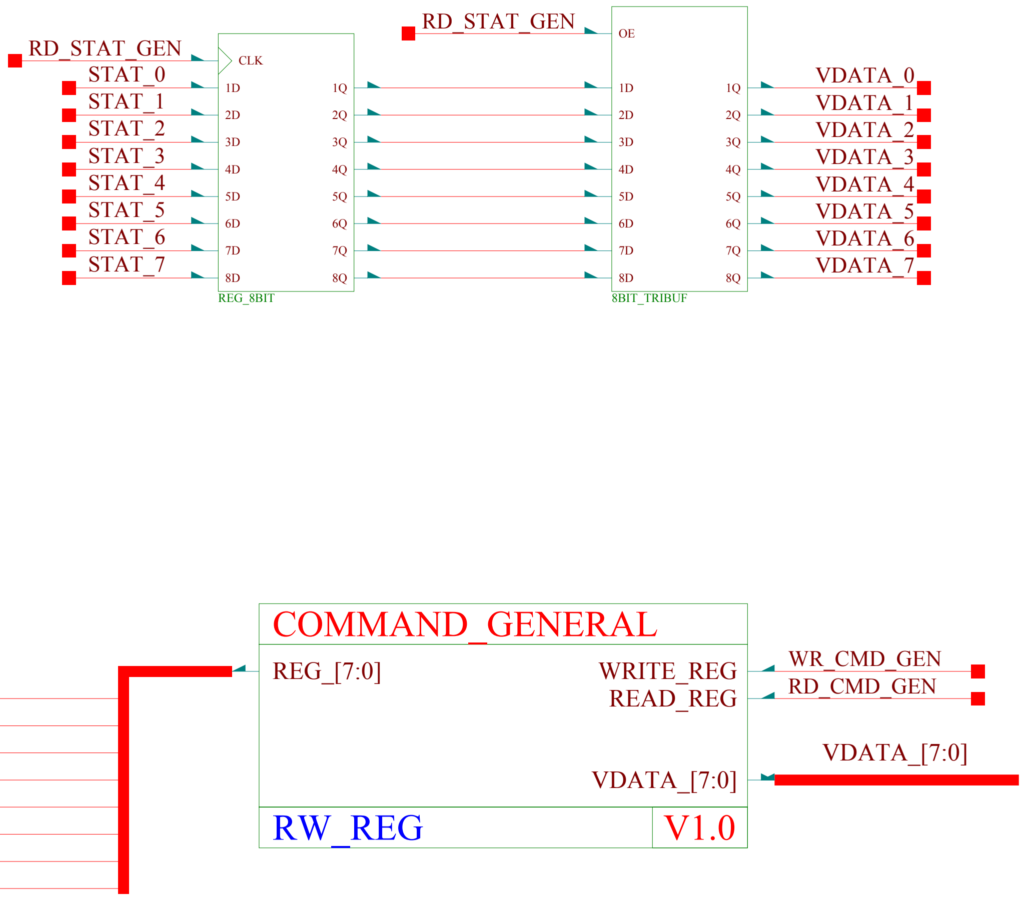
Version: V2.3	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by: HB	1-5-2006_10:02
checked by: CHECKER	0-00-0000_00:00



VME-CHIP

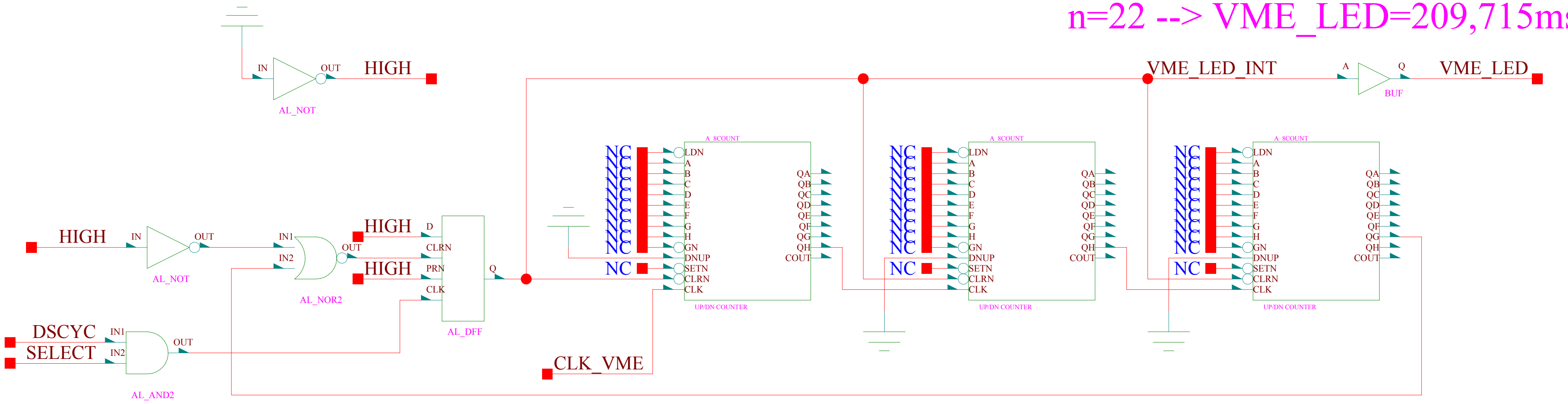
GEN_PULSE_REG

Version: V2.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by HB	9-1-2005_10:29
checked by: CHECKER	0-00-0000_00:00



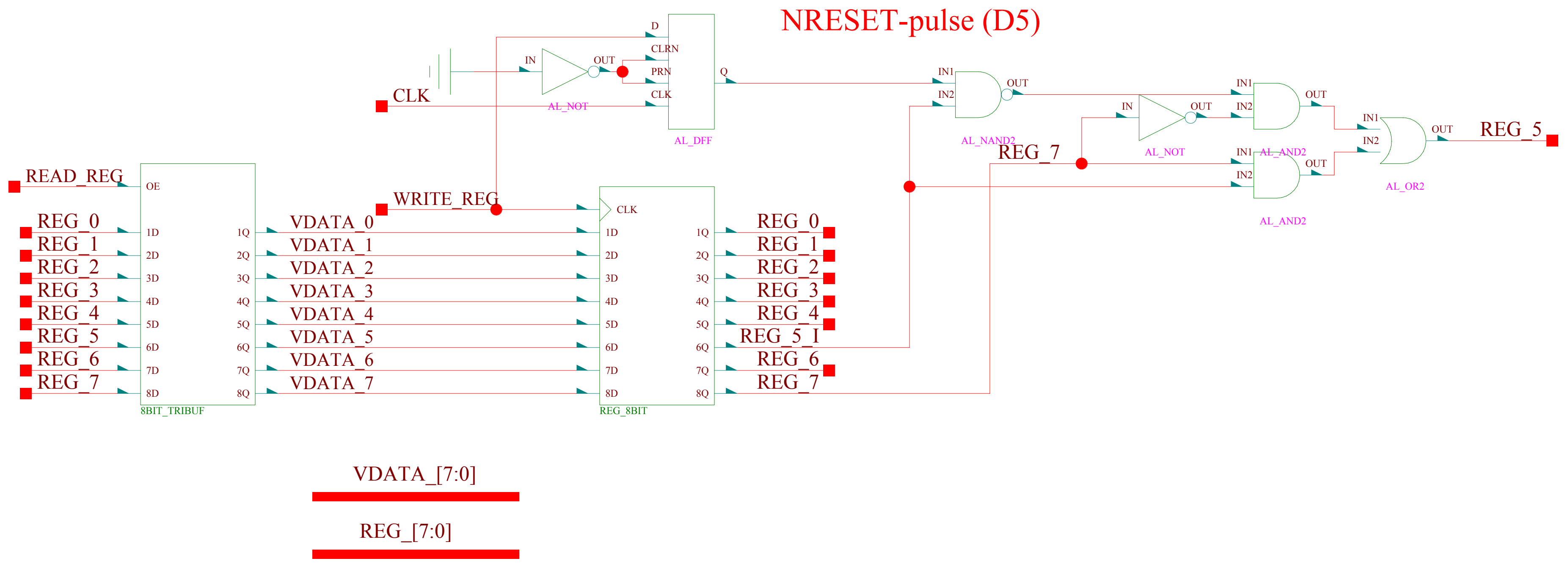
VME-CHIP	
GENERAL_REG	
Version: V2.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by: HB	9-1-2005_10:29
checked by: CHECKER	0-00-0000_00:00

n=22 --> VME_LED=209,715ms



Formel: (2**(n+1)-1)*25ns (25ns=CLK_VME)

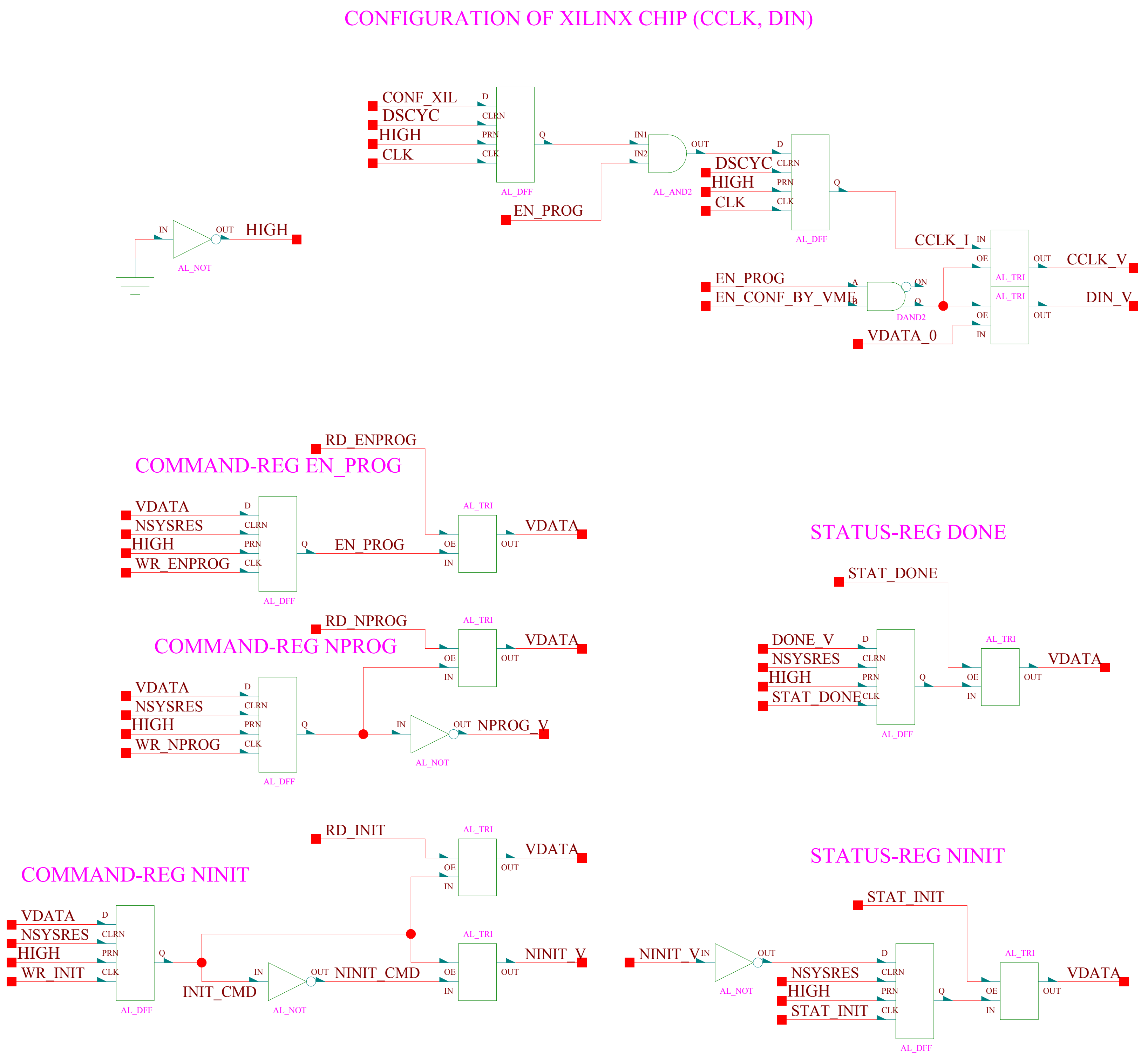
VME-CHIP	
LED_DELAY	
Version: V2.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by:HB	9-1-2005_10:32
checked by: CHECKER	0-00-0000_00:00



VME-CHIP

TTC_CTRL_REG

Version: V1.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by:HB	8-23-2005_14:00
checked by: CHECKER	0-00-0000_00:00

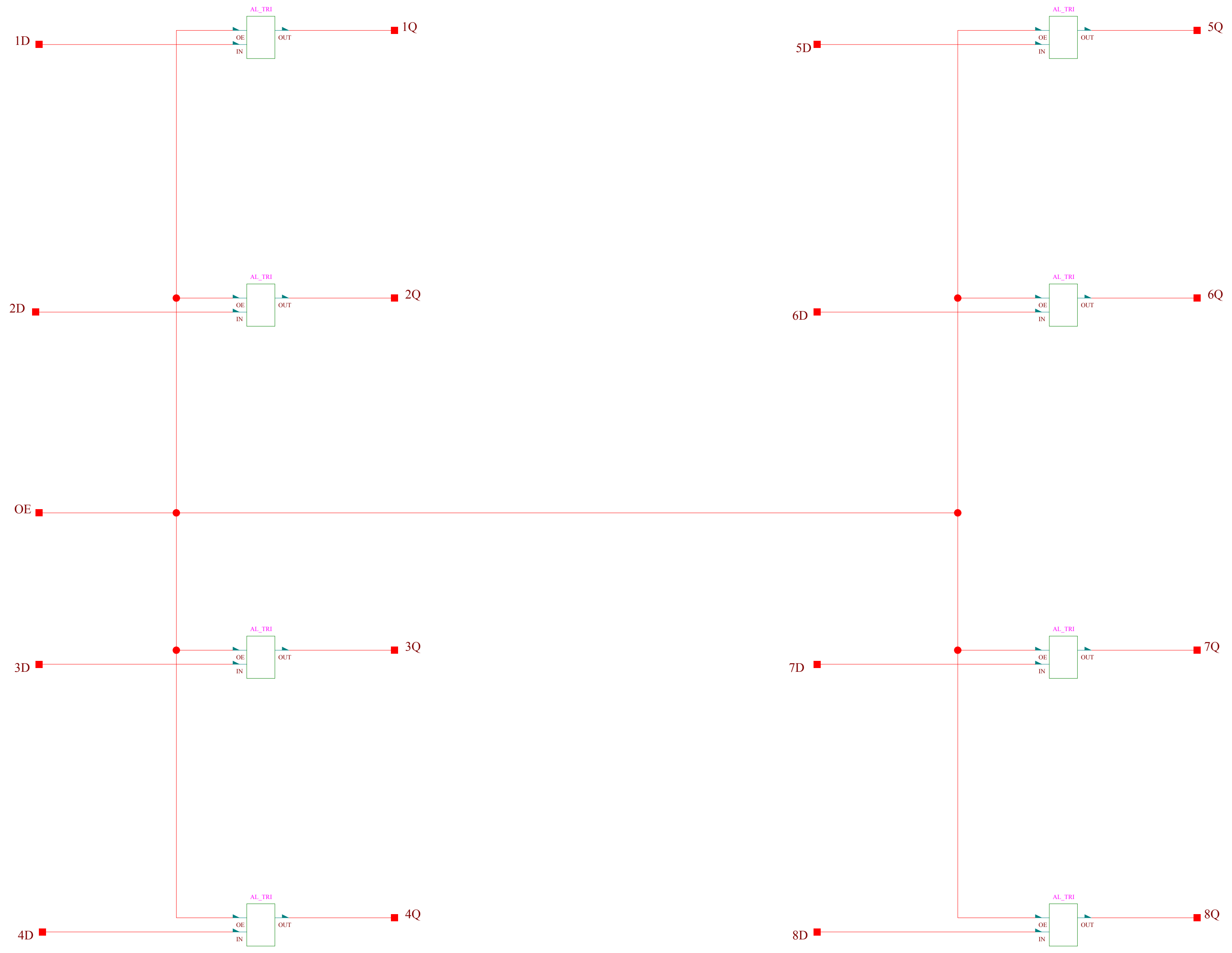


VME-CHIP

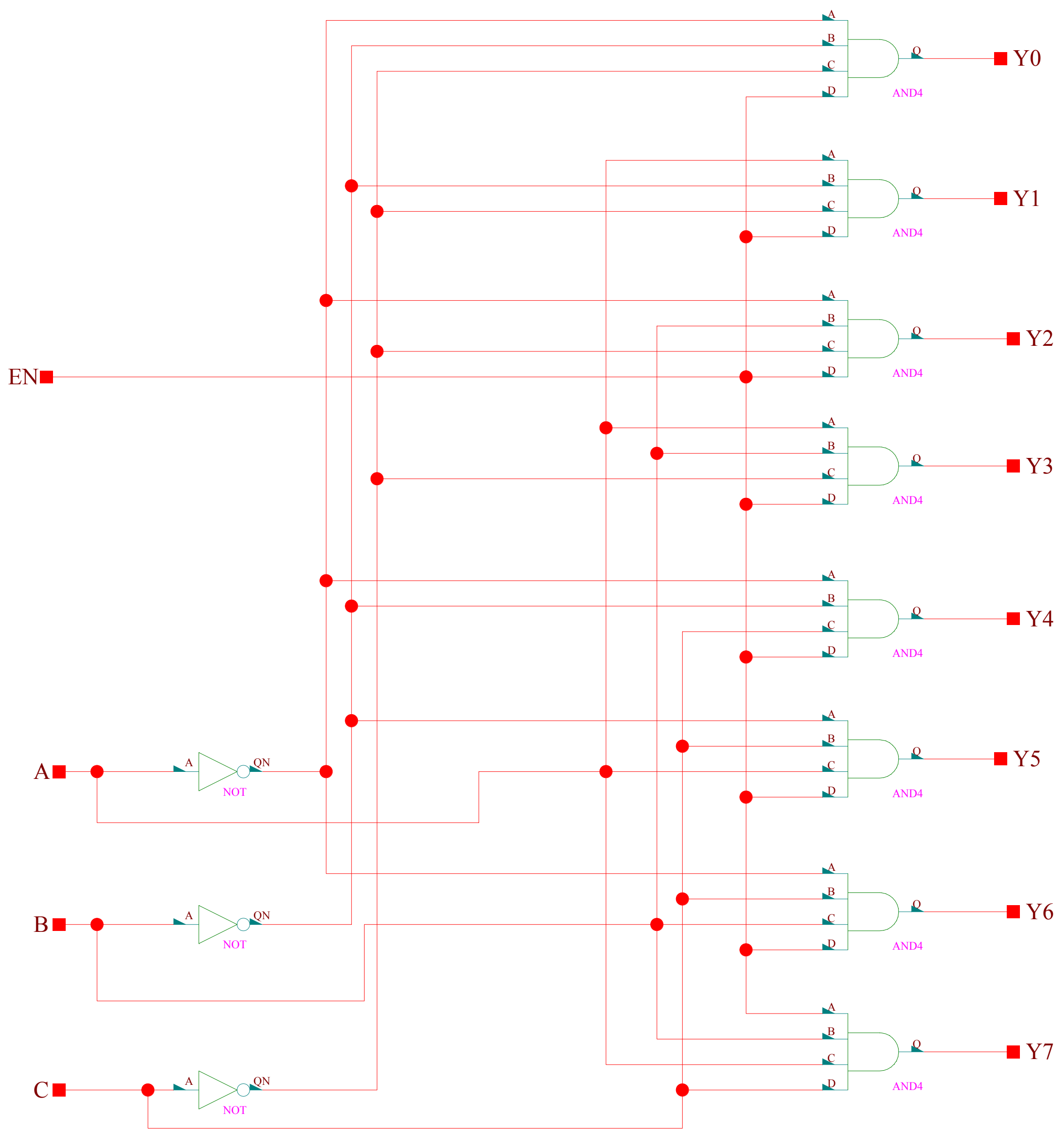
XILINX_CONF

Version: V2.0	Configuration
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by: HB	8-26-2005_14:23
checked by: CHECKER	0-00-0000_00:00

8bit_tribuf

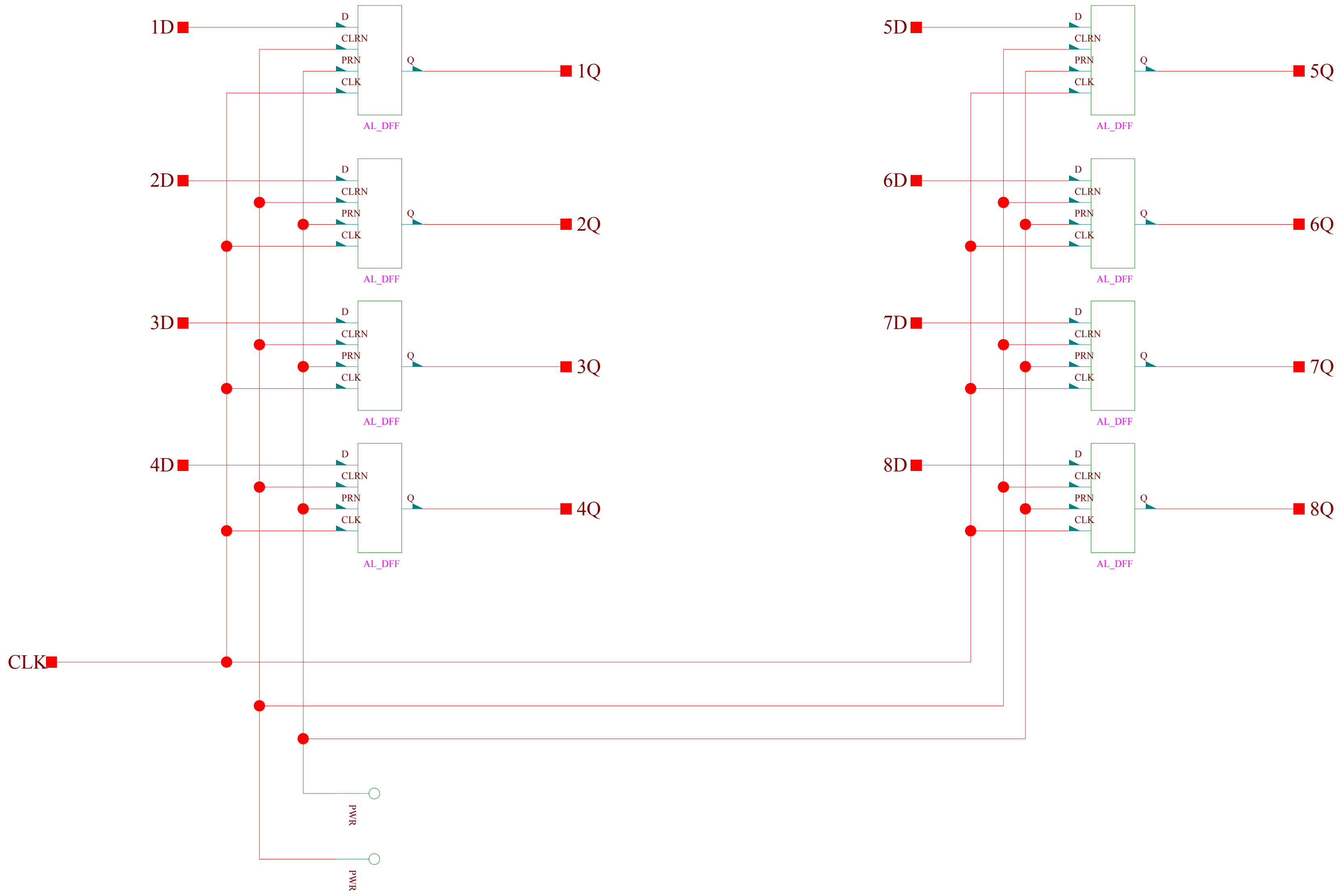


decoder_3to8

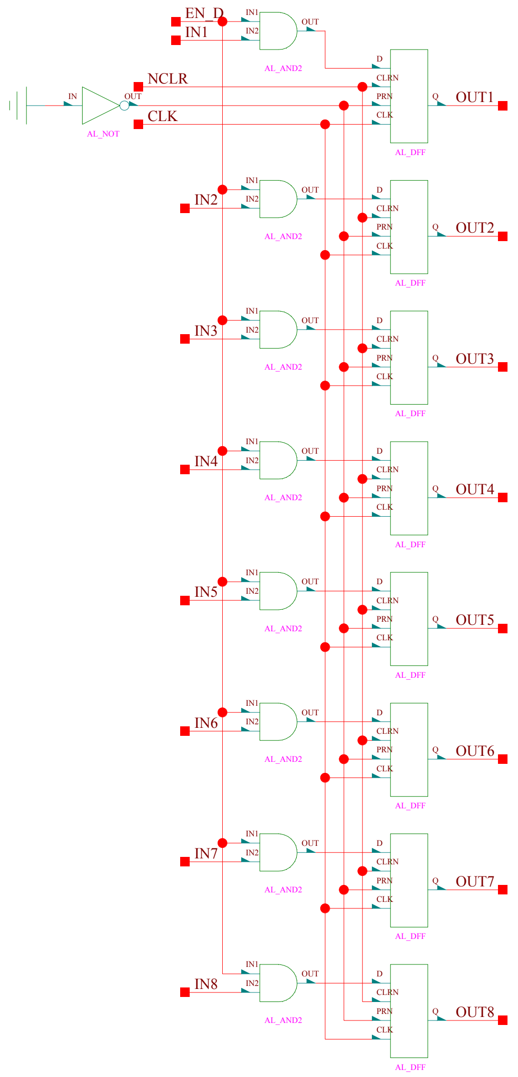


reg_8bit

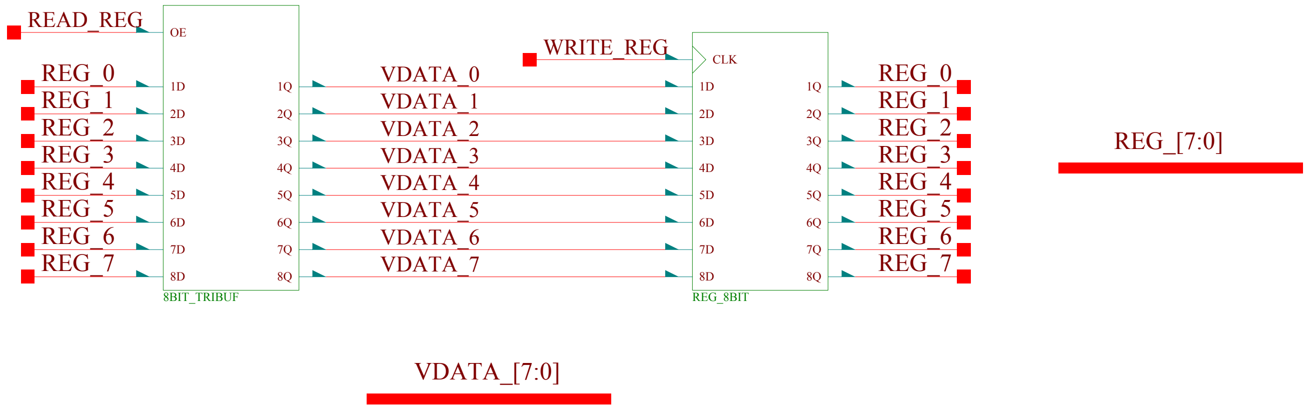
8-bit register generated by LAB3



reg_8bit_en_d



EN_D
IN1
NCLR
CLK
IN2
IN3
IN4
IN5
IN6
IN7
IN8
AL_NOT
AL_AND2
AL_DFF
IN
OUT
D
CLRN
PRN
Q
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
OUT8



VME-CHIP	
RW_REG	
Version: V1.0	
HEPHY VIENNA ELEKTRONIK 1	sheet 1 of 1
modified by:HB	9-9-2005_14:49
checked by: CHECKER	0-00-0000_00:00

```
------------------------------------------------------------
--                                                        --
-- LOGIC CORE: vme-chip logic                             --
-- MODULE NAME: chip_id_version                           --
-- INSTITUTION: Hephy Vienna                              --
-- DESIGNER: H. Bergauer                                  --
--                                                        --
-- VERSION: V2.0                                          --
-- DATE: 08 2005                                          --
--                                                        --
-- FUNCTIONAL DESCRIPTION:                                --
-- chip_id and version register (read only)               --
--                                                        --
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY altera;
USE altera.maxplus2.ALL;

USE work.constant_pkg.ALL;

ENTITY chip_id_version IS
    PORT(
        VDATA       : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        CARD_NR     : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        CHIP_ID     : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        VERSION     : IN    STD_LOGIC_VECTOR(3 DOWNTO 0));
END chip_id_version;

ARCHITECTURE rtl OF chip_id_version IS

--  CONSTANT for card_name and version defined in working-dir\vhdl !!!

    SIGNAL chip_id_value : std_logic_vector(31 downto 0);
--  CONSTANT chip_id_value: STD_LOGIC_VECTOR(31 DOWNTO 0) := X"0001B221"; -- TIM-6U_V2 #2 - V4 !!!
    CONSTANT cms_gt: STD_LOGIC_VECTOR(15 DOWNTO 0) := X"0001"; -- fixed code for GT-system = 0x0001
--  CONSTANT card_name: STD_LOGIC_VECTOR(3 DOWNTO 0) := X"B"; -- TIM-6U_V2 => 0xB
--  card_nr will be delivered from VME64x-chip in future HB250805
--  CONSTANT card_nr: STD_LOGIC_VECTOR(3 DOWNTO 0) := X"0"; -- fixed code for card_nr = 0x0
    CONSTANT chip_name: STD_LOGIC_VECTOR(3 DOWNTO 0) := X"2"; -- fixed code for chip_name = 0x2
    CONSTANT chip_nr: STD_LOGIC_VECTOR(3 DOWNTO 0) := X"1"; -- fixed code for chip_nr = 0x1

BEGIN
chip_id_value <= cms_gt & card_name & card_nr & chip_name & chip_nr;


```

```
-- chip_id and version register (read only)
tri_chip_id_3:
FOR i IN 0 TO 7 GENERATE
    call_chip_id_3: tri
   PORT MAP(chip_id_value(i+24),
      chip_id(3),
      vdata(i));
END GENERATE tri_chip_id_3;

tri_chip_id_2:
FOR i IN 0 TO 7 GENERATE
    call_chip_id_2: tri
   PORT MAP(chip_id_value(i+16),
      chip_id(2),
      vdata(i));
END GENERATE tri_chip_id_2;

tri_chip_id_1:
FOR i IN 0 TO 7 GENERATE
    call_chip_id_1: tri
   PORT MAP(chip_id_value(i+8),
      chip_id(1),
      vdata(i));
END GENERATE tri_chip_id_1;

tri_chip_id_0:
FOR i IN 0 TO 7 GENERATE
    call_chip_id_0: tri
   PORT MAP(chip_id_value(i),
      chip_id(0),
      vdata(i));
END GENERATE tri_chip_id_0;

tri_version_3:
FOR i IN 0 TO 7 GENERATE
    call_version_3: tri
   PORT MAP(version_value(i+24),
      version(3),
      vdata(i));
END GENERATE tri_version_3;

tri_version_2:
FOR i IN 0 TO 7 GENERATE
    call_version_2: tri
   PORT MAP(version_value(i+16),
      version(2),
      vdata(i));
END GENERATE tri_version_2;

tri_version_1:
FOR i IN 0 TO 7 GENERATE
    call_version_1: tri
   PORT MAP(version_value(i+8),
      version(1),
      vdata(i));
END GENERATE tri_version_1;

tri_version_0:
FOR i IN 0 TO 7 GENERATE
    call_version_0: tri
   PORT MAP(version_value(i),
      version(0),
      vdata(i));
END GENERATE tri_version_0;

END ARCHITECTURE rtl;
```

```
-------------------------------------------------------------------------------
-- Title      : Top of JTAG Controller for vme-chip of GT-boards
-- Project    : 
-------------------------------------------------------------------------------
-- File       : jtag_ctrl.vhd
-- Author     : H. Bergauer
-- Company    : 
-------------------------------------------------------------------------------
-- Description: top module for VIEWDRAW of JTAG Controller 
-------------------------------------------------------------------------------
-- $Date: 2006/01/05 08:54:17 $
-- $Revision: 1.5 $
-------------------------------------------------------------------------------

library IEEE;
library work;

use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

LIBRARY altera;
USE altera.maxplus2.ALL;

-------------------------------------------------------------------------
-- Entity Declaration
-------------------------------------------------------------------------

entity jtag_ctrl is
  port (
    addr       : in    std_logic_vector(3 downto 1);
    data       : inout    std_logic_vector(7 downto 0);
    clk        : in    std_logic;
    en_jtag    : in    std_logic;
--    dssync     : in    std_logic;
    write      : in    std_logic;
    nsysres    : in    std_logic;
    dtack      : out   std_logic;
    tck_0      : out   std_logic;
    tck_1      : out   std_logic;
    tms_0      : out   std_logic;
    tms_1      : out   std_logic;
    tdo_0      : out   std_logic;
    tdo_1      : out   std_logic;
    tdi_0      : in    std_logic;
    tdi_1      : in    std_logic
);
end;

-------------------------------------------------------------------------
-- Architecture declaration
-------------------------------------------------------------------------

architecture rtl of jtag_ctrl is

--  address parameters for JTAGController
  constant base_address : integer := 0;
  constant address_increment : integer := 2;
  constant addr_high : integer := 3;
  constant addr_low : integer := 1;

component JTAGController is
  generic (
    base_address      : integer := 0;   -- base address (in bytes)
    address_increment : integer := 2;   -- increment (2 for word access,
                                        -- 4 for long word access)
    addr_high : integer := 3;           -- upper index of vme_addr vector
    addr_low  : integer := 1            -- lower index of vme_addr vector
    );
  port (

    -- JTAG port
    -- to be connected directly to I/O pins of chip
    oTck  : out std_logic;
    oTms0 : out std_logic;
    oTms1 : out std_logic;
    oTdo  : out std_logic;
    iTdi0 : in  std_logic;
    iTdi1 : in  std_logic;

    -- VME port
    vme_addr       : in    std_logic_vector(addr_high downto addr_low);

    vme_data       : in    std_logic_vector(15 downto 0);  -- has to be at least 8 bit
    vme_en         : in    std_logic;   -- should not be a pulse when writing
    vme_wr         : in    std_logic;   -- state. must remain for at least two
                                        -- clocks after enable goes to 0
    vme_dtack      : out   std_logic;         -- not inverted

    vme_data_out   : out   std_logic_vector(15 downto 0);
    vme_en_out     : out   std_logic;

    -- Clock and control
    clk            : in    std_logic;
    reset          : in    std_logic);  -- asynchronous reset, active high
end component;

--  signal vme_en_jtag : std_logic;
  signal en_tri : std_logic;
  signal tck : std_logic;
  signal tdo : std_logic;
  signal data_jtag : std_logic_vector(15 downto 0);
  signal data_in : std_logic_vector(15 downto 0);
  signal data_out : std_logic_vector(15 downto 0);
  signal sysres : std_logic;

begin
-- verändert HB191205 wegen Quartus 5.1 Fehlermeldung

en_tri <= en_jtag AND NOT write;
--data <= data_out(7 downto 0);
data_in <= X"00" & data(7 downto 0);
tck_0 <= tck;
tck_1 <= tck;
tdo_0 <= tdo;
tdo_1 <= tdo;
sysres <= NOT nsysres;

inst_jtag: JTAGController
  generic map(base_address, address_increment, addr_high, addr_low)
  port map(
    oTck => tck,
    oTms0 => tms_0,
    oTms1 => tms_1,
    oTdo => tdo,
    iTdi0 => tdi_0,
    iTdi1 => tdi_1,
    vme_addr => addr,
    vme_data => data_in,
--    vme_en => vme_en_jtag,
    vme_en => en_jtag,
    vme_wr => write,
    vme_dtack => dtack,
    vme_data_out => data_jtag,
    clk => clk,
    reset => sysres
);

```

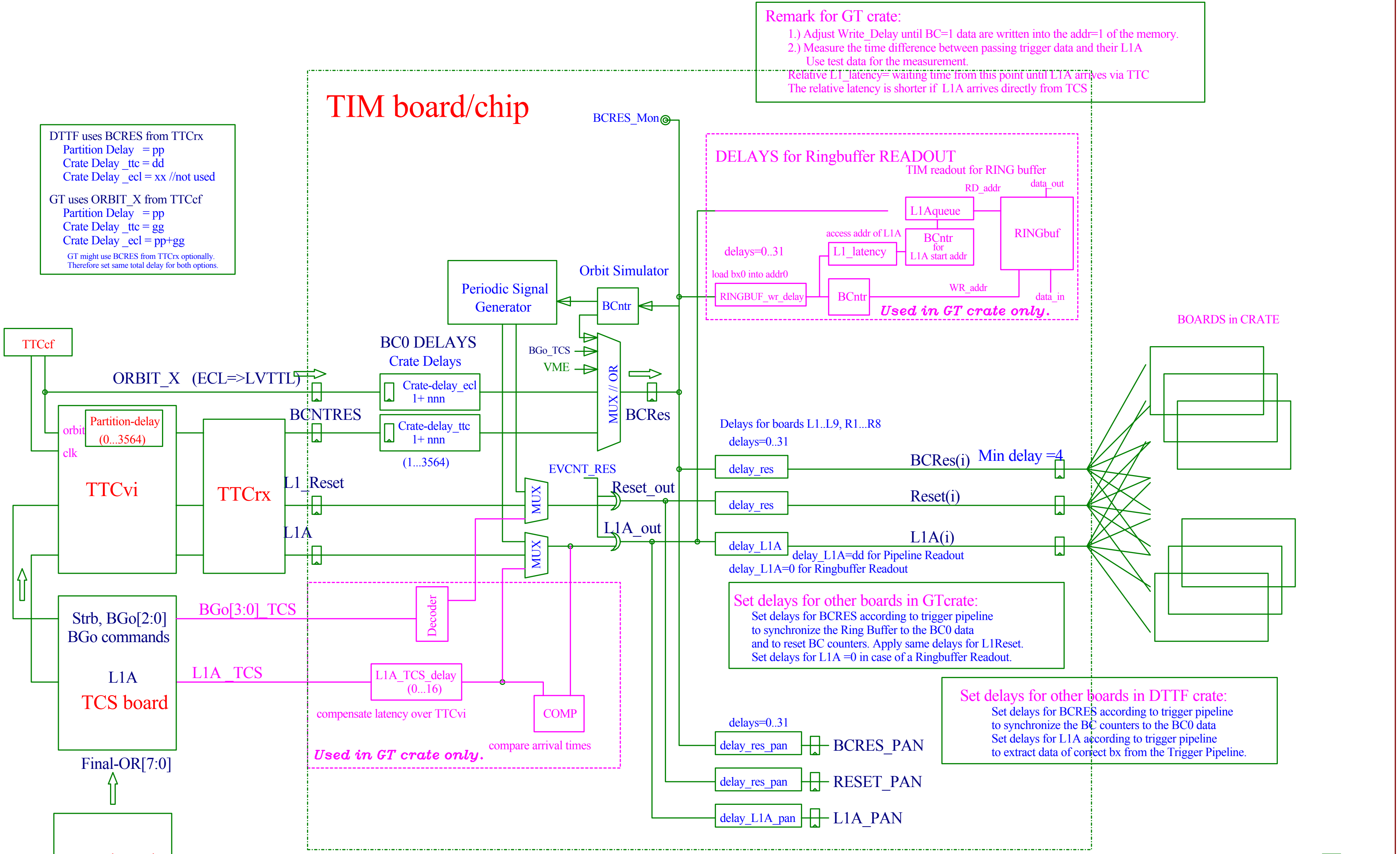
```
tri_loop:
FOR i IN 0 TO 7 GENERATE
  inst_tri: tri
  PORT MAP(data_jtag(i), en_tri, data(i));
END GENERATE tri_loop;

end rtl;
```

```vhdl
-------------------------------------------------------------------------------
-- Title      : JTAG Controller
-- Project    : 
-------------------------------------------------------------------------------
-- File       : JTAGController.vhd
-- Author     : SAKULIN Hannes  <hsakulin@dsy-srv3.cern.ch>
-- Company    : 
-------------------------------------------------------------------------------
-- Description: Simplified version of a ScanPSC100 JTAG controller 
-------------------------------------------------------------------------------
-- $Date: 2004/10/25 12:54:54 $
-- $Revision: 1.10 $
-------------------------------------------------------------------------------
-- Revision-Description:
-- generic parameter for address width of vme_addr implemented (HB)
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
-- based on:

-- File name:    Controller_soc2.vhd %
-- Title:        VHDL Controller Chip for the PHTF Soc2
-- Author:       JEJr. %
-- This VHDL Modul of the Board´s VME Controller
-- Version | Author | Mod. Date | Change |
-----------|--------|------------|      ------------------------------------------------
-- 1 | JEJr | 31.03.2004 | First design (derived from ETTF_contr_jxx) |
-----------|--------|------------|      ------------------------------------------------

library IEEE;
library work;

use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
USE IEEE.STD_LOGIC_UNSIGNED.ALL; -- for +- operators
------------------------------------------------------------------------------
-- Entity Declaration
------------------------------------------------------------------------------


entity JTAGController is
  generic (
    base_address      : integer := 0;   -- base address (in bytes)
    address_increment : integer := 2;   -- increment (2 for word access,
                                        -- 4 for long word access)
    addr_high : integer := 3;           -- upper index of vme_addr vector
    addr_low  : integer := 1            -- lower index of vme_addr vector
    );      
                                      
  port (

    -- JTAG port
    -- to be connected directly to I/O pins of chip
    oTck  : out std_logic;              
    oTms0 : out std_logic;
    oTms1 : out std_logic;
    oTdo  : out std_logic;
    iTdi0 : in  std_logic;
    iTdi1 : in  std_logic;

    -- VME port
    vme_addr       : in    std_logic_vector(addr_high downto addr_low);  
                                              
    vme_data       : in    std_logic_vector(15 downto 0);  -- has to be at least 8 bit
    vme_en         : in    std_logic;   -- should not be a pulse when writing
    vme_wr         : in    std_logic;   -- state. must remain for at least two
                                        -- clocks after enable goes to 0
    vme_dtack      : out   std_logic;         -- not inverted

    vme_data_out   : out   std_logic_vector(15 downto 0);
    vme_en_out     : out   std_logic;
        
    -- Clock and control
    clk            : in    std_logic;
    reset          : in    std_logic);  -- asynchronous reset, active high
  
end;

--------------------------------------------------------------------------------
-- VME signals:
-- 
-- the unit assumes that data and address are valid for the whole time
-- that vme_en is active
--
-- the unit further assumes that vme_en is a synchronous signal (no
-- asynchronous clear) as the falling edge of vme_en is used to trigger actions.
--------------------------------------------------------------------------------


-------------------------------------------------------------------------
-- Architecture declaration
-------------------------------------------------------------------------

architecture behavioral of JTAGController is

  -- need the following attributes to force Synplify to use
  -- input and output flip-flops
  -- (currently does not work for TDI, TMS)
  attribute syn_useioff               : boolean;
  attribute syn_useioff of behavioral : architecture is true;

-- JTAG Registers

  signal mode0_reg : std_logic_vector( 7 downto 0);
  signal mode1_reg : std_logic_vector( 7 downto 0);
  signal mode2_reg : std_logic_vector( 7 downto 0);
  signal cnt32_reg : unsigned (31 downto 0);
  signal tms0_reg  : std_logic_vector( 7 downto 0);
  signal tms1_reg  : std_logic_vector( 7 downto 0);
  signal tdo_reg   : std_logic_vector( 7 downto 0);
  signal tdi_reg   : std_logic_vector( 7 downto 0);


-- JTAG Register Enable Signals

  signal ena       : std_logic;
  signal mode0_ena : std_logic;
  signal mode1_ena : std_logic;
  signal mode2_ena : std_logic;
  signal cnt32_ena : std_logic;
  signal tms0_ena  : std_logic;
  signal tms1_ena  : std_logic;
  signal tdo_ena   : std_logic;
  signal tdi_ena   : std_logic;

  signal tdi_ena_d  : std_logic;
  signal tdi_ena_d2 : std_logic;

  signal tdi_stat : std_logic;

-- JTAG status signals
  signal cnt_loaded     : std_logic;
  signal tdo_empty      : std_logic;
  signal tms0_empty     : std_logic;
  signal tms1_empty     : std_logic;
  signal tdi_full       : std_logic;
  signal cnt_loaded_del : std_logic;
  signal tdo_empty_del  : std_logic;
  signal tms0_empty_del : std_logic;
  signal tms1_empty_del : std_logic;
  signal tdi_full_del   : std_logic;

  signal cnt_pointer : unsigned ( 1 downto 0);
  signal dtack_ff    : std_logic;

  signal tms0_a_exit : std_logic;
  signal tms1_a_exit : std_logic;

--DTACK flip-flops

  signal tdo_rdy_ff : std_logic;
  signal tdi_rdy_ff : std_logic;
  signal reg_rdy_ff        : std_logic;
  signal reg_rdy_ff_del    : std_logic;
  signal ds_del_sig        : std_logic;


-- JTAG Registers

  signal jtag_ck_cnt     : unsigned (2 downto 0);
  signal jtag_ck         : std_logic;
  signal jtag_ck_pulse   : std_logic;

  signal shift_enable : std_logic;


  signal tms01_sig : std_logic;  -- Memorize last TMS action


  signal cnt001 : std_logic;


  signal tdo_enable   : std_logic;
  signal tdi_enable   : std_logic;
  signal cnt32_enable : std_logic;
  signal tms0_enable  : std_logic;
  signal tms1_enable  : std_logic;
  signal auto_tms_h   : std_logic;
  signal int_reset    : std_logic;
  signal int_dtack    : std_logic;

  signal cnt_stat      : std_logic;
  signal cnt_stat_del  : std_logic;
  signal tms0_stat     : std_logic;
  signal tms0_stat_del : std_logic;
  signal tms1_stat     : std_logic;
  signal tms1_stat_del : std_logic;
  signal tdo_stat      : std_logic;
  signal tdo_stat_del  : std_logic;

-- JTAG Outputs internal name

  signal stms0 : std_logic;
  signal stms1 : std_logic;
  signal stdo  : std_logic;
  signal stdi0 : std_logic;
  signal stdi1 : std_logic;


  function addr_match (
    constant vme_addr : std_logic_vector;
    constant address  : integer)        -- in bytes
    return boolean is

    variable my_addr_vec : std_logic_vector(vme_addr'high downto 0);
  begin  -- process vme_addr_decode
    my_addr_vec := std_logic_vector( TO_UNSIGNED ( address, vme_addr'high+1 ) );
    return my_addr_vec(addr_high downto addr_low) = vme_addr(addr_high downto addr_low);
  end;

  signal vme_en_d  : std_logic;

  signal nreset  : std_logic;
  signal vme_cycend_p : std_logic;

begin  -- Main

  nreset <= not reset;

  -----------------------------------------------------------------------------
  -- Address Decode
  -----------------------------------------------------------------------------


  tdo_ena   <= vme_en when addr_match(vme_addr, base_address)    else '0';
  tdi_ena   <= vme_en when addr_match(vme_addr, base_address+ address_increment)  else '0'
  tms0_ena  <= vme_en when addr_match(vme_addr, base_address+ address_increment*2) else '0
  tms1_ena  <= vme_en when addr_match(vme_addr, base_address+ address_increment*3) else '0
  cnt32_ena <= vme_en when addr_match(vme_addr, base_address+ address_increment*4) else '0
  mode0_ena <= vme_en when addr_match(vme_addr, base_address+ address_increment*5) else '0
  mode1_ena <= vme_en when addr_match(vme_addr, base_address+ address_increment*6) else '0
  mode2_ena <= vme_en when addr_match(vme_addr, base_address+ address_increment*7) else '0

  ena <= mode0_ena
                or mode1_ena
                or mode2_ena
                or ( tdi_ena and tdi_full )
                or cnt32_ena
                or tdo_ena
                or tms0_ena
                or tms1_ena;

  -----------------------------------------------------------------------------
  -- Synchronization
  -----------------------------------------------------------------------------

  Synchro : process (clk, nreset)

  begin
    if nreset = '0' then
      vme_cycend_p <= '0';
      vme_en_d <= '0';
    elsif clk'event and clk='1' then
      vme_en_d <= vme_en;

      if (( vme_en = '0' ) and ( vme_en_d = '1' )) then
        vme_cycend_p <= '1';
      else
        vme_cycend_p <= '0';
      end if;
    end if;
  end process Synchro;


  -----------------------------------------------------------------------------
  -- Write registers
  -----------------------------------------------------------------------------

  write_reg: process (clk, reset)
  begin  -- process write_reg
    if reset = '1' then
      mode0_reg              <= "00100000";  --HS
      mode1_reg              <= (others => '0');
      mode2_reg (1 downto 0) <= (others => '0');
    elsif clk'event and clk = '1' then  -- rising clock edge
      if int_reset = '1' then
```

```
        mode0_reg              <= "00100000";  --HS
        mode1_reg              <= (others => '0');
        -- do not reset mode2 as it is being written to
      end if;
      if (vme_wr = '1') then
        if (mode0_ena = '1' and vme_en_d = '0') then mode0_reg <= vme_data(7 downto 0); en
        if (mode1_ena = '1' and vme_en_d = '0') then mode1_reg <= vme_data(7 downto 0); en
        if (mode2_ena = '1' and vme_en_d = '0') then mode2_reg(1 downto 0) <= vme_data(1 d
      end if;
    end if;
  end process write_reg;

-- mode registers are complete
  ----------------------------------------------------------------------------
  -- Connect status signals to registers
  ----------------------------------------------------------------------------

  tdo_enable   <= mode0_reg (7);
  tdi_enable   <= mode0_reg (6);
  cnt32_enable <= mode0_reg (5);
  tms0_enable  <= mode0_reg (4);
  tms1_enable  <= mode0_reg (3);
  auto_tms_h   <= mode0_reg (1);
  -- mode0, bit 0: loopback not implemented.


  int_reset <= mode2_reg (1);



  -- mode2, bit 0: single step not implemented
  -- mode2, bit 2: update status not implemented (status is always updated)
  -- mode2, bit 3: continuous update not implemented (status is always updated)

  mode2_reg (7) <= tdo_empty;
  mode2_reg (6) <= tdi_full;
  mode2_reg (5) <= not cnt_loaded;
  mode2_reg (4) <= tms0_empty;
  mode2_reg (3) <= tms1_empty;
  mode2_reg (2) <= shift_enable;  --????


  -- TBD: jtag mode 2 reg reset bit has to return to 0 after reset

  -- mode2, bit0: Single step CNT32 not implemented.

  ----------------------------------------------------------------------------
  -- Read registers
  ----------------------------------------------------------------------------

  vme_data_out <=      ( "00000000" & mode0_reg ) when mode0_ena = '1'
                 else ( "00000000" & mode1_reg ) when mode1_ena = '1'
                 else ( "00000000" & mode2_reg ) when mode2_ena = '1'
                 else ( "00000000" & tdi_reg )   when (( tdi_ena = '1' ) and ( tdi_full =
                 else ( "00000000" & std_logic_vector ( cnt32_reg ( 7 downto 0 )))   when
                 else ( "00000000" & std_logic_vector ( cnt32_reg ( 15 downto 8 )))  when
                 else ( "00000000" & std_logic_vector ( cnt32_reg ( 23 downto 16 ))) when
                 else ( "00000000" & std_logic_vector ( cnt32_reg ( 31 downto 24 ))) when
                 else (others => '0');

  vme_en_out <= ena;

  ------------------------------------------------------------------------------
  -- DTACK generation
  ------------------------------------------------------------------------------

  -- we do need a special dtack generation:
  --   dtack is delayed if registers are not ready when
  --   reading from tdi or writing to tms0/1, tdo and cnt32.

  -- HS : do not yet understand ds_synch story .


  Dtack_Synchro : process (clk, nreset)

  begin

    if nreset = '0' then
      reg_rdy_ff   <= '0';
--      ds_del_sig   <= '0';
    elsif clk'event and clk='1' then
--       if ds_synch = "00" then
--         ds_del_sig <= '0';
--       elsif ( not (ds_synch = "00" ) and valid_address = '1' ) then
--         ds_del_sig <= '1';
--       end if;

--       if ds_synch = "00" then
--         reg_rdy_ff   <= '0';
--       elsif ds_del_sig = '1' then
      if ena = '0' then
        reg_rdy_ff   <= '0';
      else
        if ( tdi_ena or tdo_ena or tms0_ena or tms1_ena or cnt32_ena) = '1' then
          reg_rdy_ff <= dtack_ff;
        else
          reg_rdy_ff <= '1';
        end if;
      end if;
    end if;
  end process Dtack_Synchro;


  Dtack_del_proc : process ( clk, nreset)

  begin
    if ( nreset = '0' ) then
      reg_rdy_ff_del <= '0';
--      dtack_del      <= '0';

    elsif clk'event and clk='1' then
      reg_rdy_ff_del <= reg_rdy_ff;
--      dtack_del      <= int_dtack;
    end if;
  end process Dtack_del_proc;

  int_dtack <= reg_rdy_ff_del and ena;

  vme_dtack <= int_dtack;



  -- generate dtack only when register is ready to be read/written

  -- HS: deadlocks: two successive writes to cnt without


  Jtag_dtack : process ( clk, nreset)

  begin
    if ( nreset = '0' ) then
      dtack_ff <= '0';
    elsif clk'event and clk='1'  then
      if ena = '1' then

        if (( cnt32_ena = '1' ) and ( vme_wr = '1' ) and ( cnt_loaded = '0' )) then
          dtack_ff <= '1';
        elsif (( cnt32_ena = '1' ) and ( vme_wr = '0' )) then
```

```
          dtack_ff <= '1';
        end if;

        if (( tdo_ena = '1' ) and ( vme_wr = '1' ) and ( tdo_empty = '1' )) then
          dtack_ff <= '1';
        elsif (( tdo_ena = '1' ) and ( vme_wr = '0' )) then
          dtack_ff <= '1';
        end if;

        if (( tms0_ena = '1' ) and ( vme_wr = '1' ) and ( tms0_empty = '1' )) then
          dtack_ff <= '1';
        elsif (( tms0_ena = '1' ) and ( vme_wr = '0' )) then
          dtack_ff <= '1';
        end if;

        if (( tms1_ena = '1' ) and ( vme_wr = '1' ) and ( tms1_empty = '1' )) then
          dtack_ff <= '1';
        elsif (( tms1_ena = '1' ) and ( vme_wr = '0' )) then
          dtack_ff <= '1';
        end if;

        if (( tdi_ena = '1' ) and ( vme_wr = '0' ) and ( tdi_full = '1' )) then
          dtack_ff <= '1';
        elsif (( tdi_ena = '1' ) and ( vme_wr = '1' )) then
          dtack_ff <= '1';
        end if;

      else
        dtack_ff   <= '0';
      end if;


    end if;
  end process Jtag_dtack;






  ----------------------------------
  -- Generate JTAG Clock 40 MHz / 4 = 10 MHz
  ----------------------------------
  JTAG_Clock : process (clk, nreset)

  begin

    if ( nreset = '0' ) then

      jtag_ck_cnt     <= (others => '0');
      jtag_ck         <= '0';
      jtag_ck_pulse   <= '0';


    elsif clk'event and clk='1' then

      if ( shift_enable = '1' ) then
        jtag_ck_cnt <= jtag_ck_cnt + 1;
      end if;

      if ( shift_enable = '0' ) then
        jtag_ck_cnt <= (others => '0');
      end if;

      jtag_ck <= jtag_ck_cnt (2);

      -- pulse on rising JTAG clock
      if (( jtag_ck_cnt (2) = '1' ) and ( jtag_ck = '0' )) then
        jtag_ck_pulse <= '1';
      else
        jtag_ck_pulse <= '0';
      end if;

    end if;
  end process JTAG_Clock;

  ------------------------------------------------------------------------------
  -- memorize last TMS operation
  ------------------------------------------------------------------------------

  TMSmem : process (clk, nreset)

  begin
    if ( nreset = '0' ) then
      tms01_sig   <= '0';
    elsif clk'event and clk='1' then
      if tms0_enable = '1' then
        tms01_sig <= '0';
      elsif tms1_enable = '1' then
        tms01_sig <= '1';
      end if;
    end if;
  end process TMSmem;


  cnt001 <= '1' when ( cnt32_reg = "00000000000000000000000000000001" ) else '0';

  ------------------------------------------------------------------------------
  -- 32 bit counter
  ------------------------------------------------------------------------------

  Counter : process (clk, nreset)

  begin

    if ( nreset = '0' ) then

      cnt32_reg      <= (others => '0');
      cnt_pointer    <= (others => '0');
      cnt_stat       <= '0';
      cnt_stat_del   <= '0';
      cnt_loaded     <= '0';
      cnt_loaded_del <= '0';
      tms0_a_exit           <= '0';
      tms1_a_exit           <= '0';

    elsif clk'event and clk='1' then
      if int_reset = '1' then
        cnt32_reg      <= (others => '0');
        cnt_pointer    <= (others => '0');
        cnt_stat       <= '0';
        cnt_stat_del   <= '0';
        cnt_loaded     <= '0';
        cnt_loaded_del <= '0';
        tms0_a_exit           <= '0';
        tms1_a_exit           <= '0';
      end if;

      --
      -- Load counter
      --
      if (( vme_wr = '1' ) and ( cnt32_ena = '1' ) and ( cnt_loaded = '0' )) then
        cnt_stat <= '1';
        case cnt_pointer is
          when "00"   => cnt32_reg ( 7 downto 0 )   <= unsigned ( vme_data ( 7 downto 0 ))
          when "01"   => cnt32_reg ( 15 downto 8 )  <= unsigned ( vme_data ( 7 downto 0 ))
          when "10"   => cnt32_reg ( 23 downto 16 ) <= unsigned ( vme_data ( 7 downto 0 ))
          when "11"   => cnt32_reg ( 31 downto 24 ) <= unsigned ( vme_data ( 7 downto 0 ))
          when others => cnt32_reg ( 7 downto 0 )   <= unsigned ( vme_data ( 7 downto 0 ))
        end case;
      end if;

      --
      -- Increment pointer at end of VME cycle
      --
      if (( vme_cycend_p = '1' ) and ( cnt_stat = '1' )) then
        cnt_pointer      <= cnt_pointer + 1;
        cnt_stat     <= '0';
        if cnt_pointer = "11" then
          cnt_loaded_del <= '1';
        end if;
      end if;

      cnt_loaded <= cnt_loaded_del;

      if cnt_loaded = '1' then
        cnt_pointer <= (others => '0');
      end if;

      --
      -- count down
      --
      if (( jtag_ck_pulse = '1' ) and ( shift_enable = '1' )) then
        cnt32_reg        <= cnt32_reg - 1;
        if cnt001 = '1' then
          cnt_loaded_del <= '0';
        end if;
      end if;

      --
      -- auto TMS high
      --
      if (( auto_tms_h = '1') and (cnt001 = '1') and (cnt_loaded = '1' )) then
        if tms01_sig = '0' then
          tms0_a_exit <= '1';
        else
          tms1_a_exit <= '1';
        end if;
      end if;

      if ( cnt32_reg =  "00000000000000000000000000000000" ) then
        tms0_a_exit <= '0';
        tms1_a_exit <= '0';
      end if;

    end if;
  end process Counter;


  -------------------------------------------------------------------------------
  -- TMS 0
  -------------------------------------------------------------------------------
  tms0 : process (clk, nreset)

    variable tms0_bytcnt : unsigned ( 3 downto 0 );

  begin

    if ( nreset = '0' ) then

      tms0_reg       <= (others => '0');
      tms0_empty     <= '1';
      tms0_empty_del <= '1';
      tms0_bytcnt := (others    => '0');
      tms0_stat      <= '0';
      tms0_stat_del  <= '0';

    elsif clk'event and clk='1' then
      if int_reset = '1' then
        tms0_reg       <= (others => '0');
        tms0_empty     <= '1';
        tms0_empty_del <= '1';
        tms0_bytcnt := (others    => '0');
        tms0_stat      <= '0';
        tms0_stat_del  <= '0';
      end if;

      tms0_empty <= tms0_empty_del;

      -- load TMS0 register
      if ( vme_wr and tms0_ena ) = '1' and tms0_empty = '1' then
        tms0_reg      <= vme_data ( 7 downto 0);
        tms0_stat_del <= '1';
      end if;

      tms0_stat <= tms0_stat_del;

      if (( tms0_stat = '1' ) and ( vme_cycend_p = '1' )) then
        tms0_bytcnt := "1000";
        tms0_empty_del <= '0';
        tms0_stat_del  <= '0';
      end if;

      if (( jtag_ck_pulse = '1' ) and ( shift_enable = '1' ) and ( tms0_enable = '1' )) th
        tms0_reg <= '0' & tms0_reg ( 7 downto 1);

        tms0_bytcnt := tms0_bytcnt - 1;

        if tms0_bytcnt = "0000" then
          tms0_empty_del <= '1';
        end if;
      end if;
      if cnt_loaded_del = '0' then
        tms0_empty_del   <= '1';
      end if;
    end if;
  end process tms0;

  stms0 <= tms0_reg (0) or tms0_a_exit;


  -----------------------------------------------------------------------------
  -- TMS 1
  -----------------------------------------------------------------------------
  tms1 : process (clk, nreset)

    variable tms1_bytcnt : unsigned ( 3 downto 0 );

  begin

    if ( nreset = '0' ) then

      tms1_reg       <= (others => '0');
      tms1_empty     <= '1';
      tms1_empty_del <= '1';
      tms1_bytcnt := (others    => '0');
      tms1_stat      <= '0';
      tms1_stat_del  <= '0';

    elsif clk'event and clk='1' then
      if int_reset = '1' then
        tms1_reg       <= (others => '0');
        tms1_empty     <= '1';
        tms1_empty_del <= '1';
        tms1_bytcnt := (others    => '0');
        tms1_stat      <= '0';
        tms1_stat_del  <= '0';
      end if;

      tms1_empty <= tms1_empty_del;

      if ( vme_wr and tms1_ena ) = '1' and  tms1_empty = '1' then
        tms1_reg      <= vme_data ( 7 downto 0);
        tms1_stat_del <= '1';
      end if;

      tms1_stat <= tms1_stat_del;

      if (( tms1_stat = '1' ) and ( vme_cycend_p = '1' )) then
        tms1_bytcnt := "1000";
        tms1_empty_del <= '0';
        tms1_stat_del  <= '0';
      end if;
      if (( jtag_ck_pulse = '1' ) and ( shift_enable = '1' ) and ( tms1_enable = '1' )) th
        tms1_reg <= '0' & tms1_reg ( 7 downto 1);

        tms1_bytcnt := tms1_bytcnt - 1;

        if tms1_bytcnt = "0000" then
          tms1_empty_del <= '1';
        end if;
      end if;
      if cnt_loaded = '0' then
        tms1_empty_del   <= '1';
      end if;
    end if;
  end process tms1;

  stms1 <= tms1_reg (0) or tms1_a_exit;

  -------------------------------------------------------------------------------
  -- TDO
  -------------------------------------------------------------------------------
  tdo : process (clk, nreset)

    variable tdo_bytcnt : unsigned ( 3 downto 0 );

  begin

    if ( nreset = '0' ) then

      tdo_reg       <= (others => '0');
      tdo_empty     <= '1';
      tdo_empty_del <= '1';
      tdo_bytcnt := (others    => '0');
      tdo_stat      <= '0';
      tdo_stat_del  <= '0';

    elsif clk'event and clk='1' then
      if int_reset = '1' then
        tdo_reg       <= (others => '0');
        tdo_empty_del <= '1';
        tdo_bytcnt := (others    => '0');
        tdo_stat      <= '0';
        tdo_stat_del  <= '0';
      end if;

      tdo_empty <= tdo_empty_del;

      if ( vme_wr and tdo_ena ) = '1' and tdo_empty = '1' then
        tdo_reg      <= vme_data ( 7 downto 0);
        tdo_stat_del <= '1';
      end if;

      tdo_stat <= tdo_stat_del;

      if tdo_stat = '1' and vme_cycend_p = '1' then
        tdo_bytcnt := "1000";
        tdo_empty_del <= '0';
        tdo_stat_del  <= '0';
      end if;

      if (( jtag_ck_pulse = '1' ) and ( shift_enable = '1' ) and ( tdo_enable = '1')) then
        tdo_reg <= '0' & tdo_reg ( 7 downto 1);

        tdo_bytcnt := tdo_bytcnt - 1;

        if tdo_bytcnt = "0000" then
          tdo_empty_del <= '1';
        end if;
      end if;

      if cnt_loaded = '0' then
        tdo_empty_del <= '1';
      end if;
    end if;
  end process tdo;

  stdo <= tdo_reg (0);


  ----------------------------------------------------------------------------
  -- TDI
  ----------------------------------------------------------------------------
  tdi : process (clk, nreset)

    variable tdi_bytcnt : unsigned ( 3 downto 0 );

  begin

    if ( nreset = '0' ) then

      tdi_reg      <= (others => '0');
      tdi_full_del <= '0';
      tdi_full     <= '0';
      tdi_bytcnt := "1000";
  --    tdi_ena_d  <= '0';
  --    tdi_ena_d2 <= '0';
      tdi_stat <= '0';
    elsif clk'event and clk='1' then
      if int_reset = '1' then
        tdi_reg  <= (others => '0');
        tdi_full_del <= '0';
        tdi_full <= '0';
        tdi_bytcnt := "1000";
        tdi_stat <= '0';
      end if;

      -- delay tdi_ena so that it is still valid during vme_cycend_p
--      tdi_ena_d  <= tdi_ena;
--      tdi_ena_d2 <= tdi_ena_d;
      -- vme_wr stays on bus longer, do not need to delay

      -- set stat if there is a read on the TDI register
      if (( vme_wr = '0' ) and ( tdi_ena = '1' ) and ( tdi_full = '1' )) then
        tdi_stat <= '1';
      end if;

      -- if stat is set, clear full on end of cycle
      if ( (tdi_stat = '1') and (vme_cycend_p = '1')) then
        tdi_stat <= '0';
        tdi_full_del <= '0';
        tdi_bytcnt := "1000";
      end if;

      tdi_full <= tdi_full_del;  -- Delay full signal by one clock

      if (( jtag_ck_pulse = '1' ) and ( shift_enable = '1' ) and ( tdi_enable = '1' )) the

        tdi_reg ( 6 downto 0 ) <= tdi_reg ( 7 downto 1 );

        if tms01_sig = '0' then
          tdi_reg ( 7 ) <= stdi0;
        else
          tdi_reg ( 7 ) <= stdi1;
        end if;

        tdi_bytcnt := tdi_bytcnt - 1;

        -- HS: condition unnecessary? would not get here if shift_enable was 0

        if (( tdi_bytcnt = "0000" ) or (shift_enable = '0')) then
          tdi_full_del <= '1';
        end if;
      end if;

      -- stop shifting if at last JTAG clock pulse
      if (( jtag_ck_pulse = '1' ) and ( cnt001 = '1' ) and ( tdi_enable = '1' ))then
        tdi_full_del   <= '1';
      end if;
    end if;


-- HS: why different stop conditions? cnt_loaded, cnt_loaded_del, cnt001



  end process tdi;




  ---------------------------------------------------------------------------
  -- Shift Enable
  ---------------------------------------------------------------------------

  -- HS: shift enable, when counter loaded _AND_ one of the following:
  --     TMS0 enabled and not empty
  --     TMS1 enabled and not empty
  --     TD0 enabled and not empty _AND_ TDI enabled and not full

  -- only works if TDO and TDI are enabled and if TDI is actually read out

  shift_en_proc : process (clk, nreset)

  begin

    if ( nreset = '0' ) then

      shift_enable <= '0';

    elsif clk'event and clk='1' then

      if cnt_loaded = '1' then
        if (( tms0_enable = '1') and ( tms0_empty = '0' )) then
          shift_enable <= '1';
        elsif (( tms1_enable = '1') and ( tms1_empty = '0' )) then
          shift_enable <= '1';
        elsif (( tdo_enable = '1') and ( tdo_empty = '0' )
               and ( tdi_enable = '1') and ( tdi_full = '0' )) then
          shift_enable <= '1';
        else
          shift_enable <= '0';
        end if;
      else
        shift_enable   <= '0';

      end if;
    end if;
  end process shift_en_proc;

  ------------------------------------------------------------------------------
  -- Test clock generation
  ------------------------------------------------------------------------------

  -- HS: why invert the clock ??
  -- internal operations shpuld happen on falling edge, external ones on rising
  -- edge. 

  tclk_generation : process (clk, nreset)

  begin

    if ( nreset = '0' ) then

      oTck <= '0';

    elsif clk'event and clk='1' then
      if shift_enable = '1' then
        oTck <= not jtag_ck;
      else
        oTck <= '0';
      end if;
    end if;

  end process tclk_generation;



-------------------------

  oTdo   <= stdo;
  oTms0  <= stms0;
  oTms1  <= stms1; 
  stdi0 <= iTdi0; 
  stdi1 <= iTdi1;                   -- Separate TDI inputs (as long no tri-state at ETTF_F


end behavioral;
```

```
------------------------------------------------------------
--                                                        --
-- LOGIC CORE: GT logic                                   --
-- MODULE NAME: sel_test_outputs                          --
-- INSTITUTION: Hephy Vienna                              --
-- DESIGNER: H. Bergauer                                  --
--                                                        --
-- VERSION: V1.0                                          --
-- DATE: 08 2005                                          --
--                                                        --
-- FUNCTIONAL DESCRIPTION:                                --
-- selection of 1 of 16 signals for                       --
-- 4 test_out-pins (scope)                                --
--                                                        --
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY sel_test_outputs IS
    PORT(
--      test_signals_0      : IN    STD_LOGIC;
--      test_signals_1      : IN    STD_LOGIC;
--      test_signals_2      : IN    STD_LOGIC;
--      test_signals_3      : IN    STD_LOGIC;
--      test_signals_4      : IN    STD_LOGIC;
--      test_signals_5      : IN    STD_LOGIC;
--      test_signals_6      : IN    STD_LOGIC;
--      test_signals_7      : IN    STD_LOGIC;
--      test_signals_8      : IN    STD_LOGIC;
--      test_signals_9      : IN    STD_LOGIC;
--      test_signals_10     : IN    STD_LOGIC;
--      test_signals_11     : IN    STD_LOGIC;
--      test_signals_12     : IN    STD_LOGIC;
--      test_signals_13     : IN    STD_LOGIC;
--      test_signals_14     : IN    STD_LOGIC;
--      test_signals_15     : IN    STD_LOGIC;
        test_signals        : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
        en_1_signals        : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        en_2_signals        : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        en_3_signals        : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        en_4_signals        : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
        test_outputs        : OUT   STD_LOGIC_VECTOR(4 DOWNTO 1)
        );
END sel_test_outputs;

ARCHITECTURE rtl OF sel_test_outputs IS
--COMPONENT test_out_coded IS
--  PORT(
--      en_signals          : IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
--      test_signals        : IN    STD_LOGIC_VECTOR(15 DOWNTO 0);
--      test_out            : OUT   STD_LOGIC
--      );
--END COMPONENT test_out_coded;

--  SIGNAL test_signals_vec : STD_LOGIC_VECTOR(15 DOWNTO 0);

    TYPE en_signals_type IS ARRAY (4 DOWNTO 1)
        OF STD_LOGIC_VECTOR(3 DOWNTO 0);

    SIGNAL en_signals_arr : en_signals_type;
BEGIN
-- ***************************************************************
-- ERKLÄRUNG:
-- en_signals sind codiert, 4 bits aus VME-registers (2x8 bits für
-- 4 test_outputs mit je 16 test-signals), die angeben,
-- welches interne signal auf test-output gelegt wird.
-- test_signals ist der vector, der die internen signals enthält.
-- Korrespondierend mit dem value der en_signals wird das jeweilige
-- interne signal auf den test-output gelegt. Definition des internen
-- signals in vector notwendig!!!
-- *****************************************************************

en_signals_arr(4) <= en_4_signals;
en_signals_arr(3) <= en_3_signals;
en_signals_arr(2) <= en_2_signals;
en_signals_arr(1) <= en_1_signals;

--test_signals_vec <=
--  test_signals_15 & test_signals_14 & test_signals_13 & test_signals_12 &
--  test_signals_11 & test_signals_10 & test_signals_9 & test_signals_8 &
--  test_signals_7 & test_signals_6 & test_signals_5 & test_signals_4 &
--  test_signals_3 & test_signals_2 & test_signals_1 & test_signals_0;

loop_test_outputs:
    for i in 1 to 4 generate
        test_outputs(i) <= test_signals(CONV_INTEGER(en_signals_arr(i)));
    end generate loop_test_outputs;

--loop_test_outputs:
--for i in 1 to 4 generate
--  call_test_out: test_out_coded
--      PORT MAP(en_signals_arr(i), test_signals_vec, test_outputs(i));
--end generate loop_test_outputs;

END rtl;
```



DTTF uses BCRES from TTCrx
Partition Delay = pp
Crate Delay _ttc = dd
Crate Delay _ecl = xx //not used

GT uses ORBIT_X from TTCcf
Partition Delay = pp
Crate Delay _ttc = gg
Crate Delay _ecl = pp+gg

GT might use BCRES from TTCrx optionally.
Therefore set same total delay for both options.

Remark for GT crate:
1.) Adjust Write_Delay until BC=1 data are written into the addr=1 of the memory.
2.) Measure the time difference between passing trigger data and their L1A
Use test data for the measurement.
Relative L1_latency= waiting time from this point until L1A arrives via TTC
The relative latency is shorter if L1A arrives directly from TCS

Set delays for other boards in GTcrate:
Set delays for BCRES according to trigger pipeline
to synchronize the Ring Buffer to the BC0 data
and to reset BC counters. Apply same delays for L1Reset.
Set delays for L1A =0 in case of a Ringbuffer Readout.

Set delays for other boards in DTTF crate:
Set delays for BCRES according to trigger pipeline
to synchronize the BC counters to the BC0 data
Set delays for L1A according to trigger pipeline
to extract data of correct bx from the Trigger Pipeline.

Remark for GT and DTTF crate:
BCRES has to arrive early enough on the board that receives trigger data first.
If the delay in the connected TTCvi doesn't fit or if ORBIT arrives directly from TTCcf
then the CRATE DELAY is set accordingly.
The delay for the board that receives trigger data first should be set to a minimum value close to 0.
Then a delay of <30bx will be sufficient for the last board (FDL in GTcrate, SORTER in DTTFcrate)

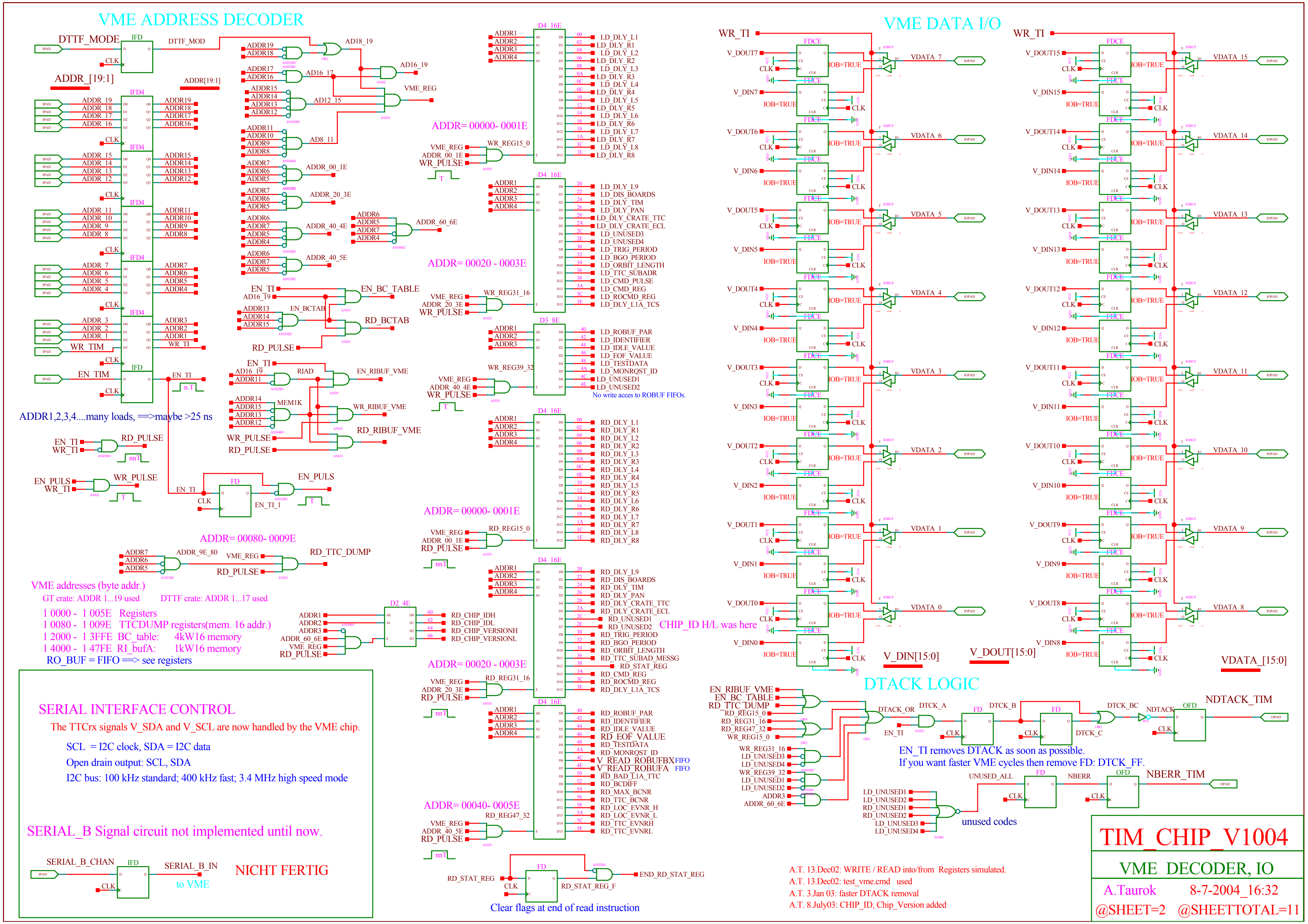
V1001: L1A, ORBIT_X as positive active signals
V1001: New VME command (BGo cmds)
V1002: DCM Factory_JF set to default value

Version 1003: STATUS to FDL pg.5
Version 1003: Encoded BGO to backplane, pg.6
Version 1003: Stop L1A by RUN_FF at next BCRES, pg.6
Version 1003: DTTF ignores Private Gap and Orbit

CHIP ID = 0001 4211 // 0001=GT, 4=TIM card 2=TIMchip 1=card#, 1=chip#
CHIP VERSION = 0001 0011 //=1003

TO BE synthesized!!

TIM_CHIP_V1004	
DELAY OVERVIEW	
A.Taurok	8-7-2004_16:32
@SHEET=1	@SHEETTOTAL=11



VME addresses (byte addr.)

GT crate: ADDR 1...19 used   DTTF crate: ADDR 1...17 used

1 0000 - 1 005E	Registers	
1 0080 - 1 009E	TTCDUMP registers(mem. 16 addr.)	
1 2000 - 1 3FFE	BC_table:	4kW16 memory
1 4000 - 1 47FE	RI_bufA:	1kW16 memory

RO_BUF = FIFO ==> see registers

SERIAL INTERFACE CONTROL

The TTCrx signals V_SDA and V_SCL are now handled by the VME chip.

SCL = I2C clock, SDA = I2C data

Open drain output: SCL, SDA

I2C bus: 100 kHz standard; 400 kHz fast; 3.4 MHz high speed mode

SERIAL_B Signal circuit not implemented until now.

NICHT FERTIG

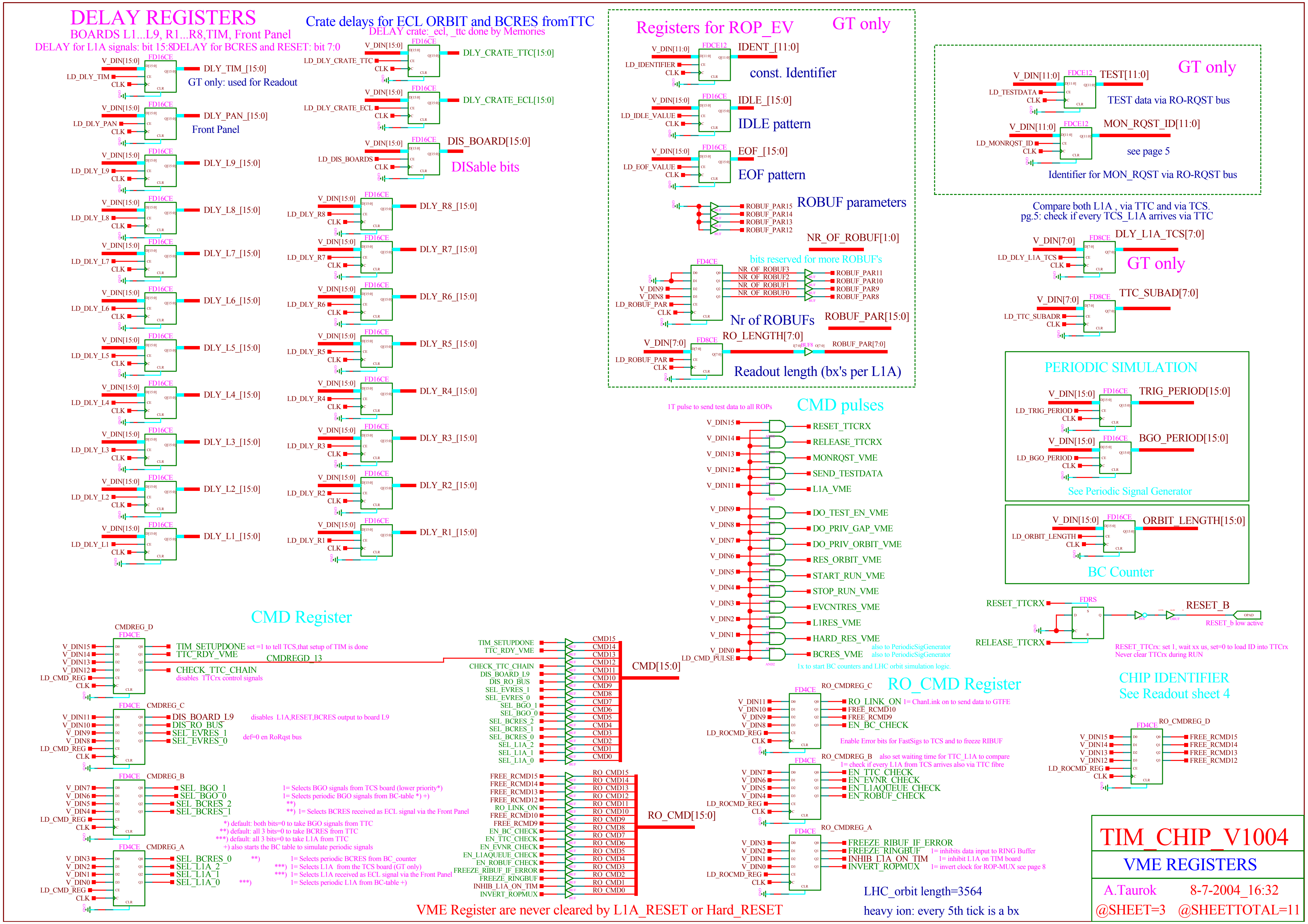
A.T. 13.Dec02: WRITE / READ into/from Registers simulated.
A.T. 13.Dec02: test_vme.cmd used
A.T. 3.Jan 03: faster DTACK removal
A.T. 8.July03: CHIP_ID, Chip_Version added

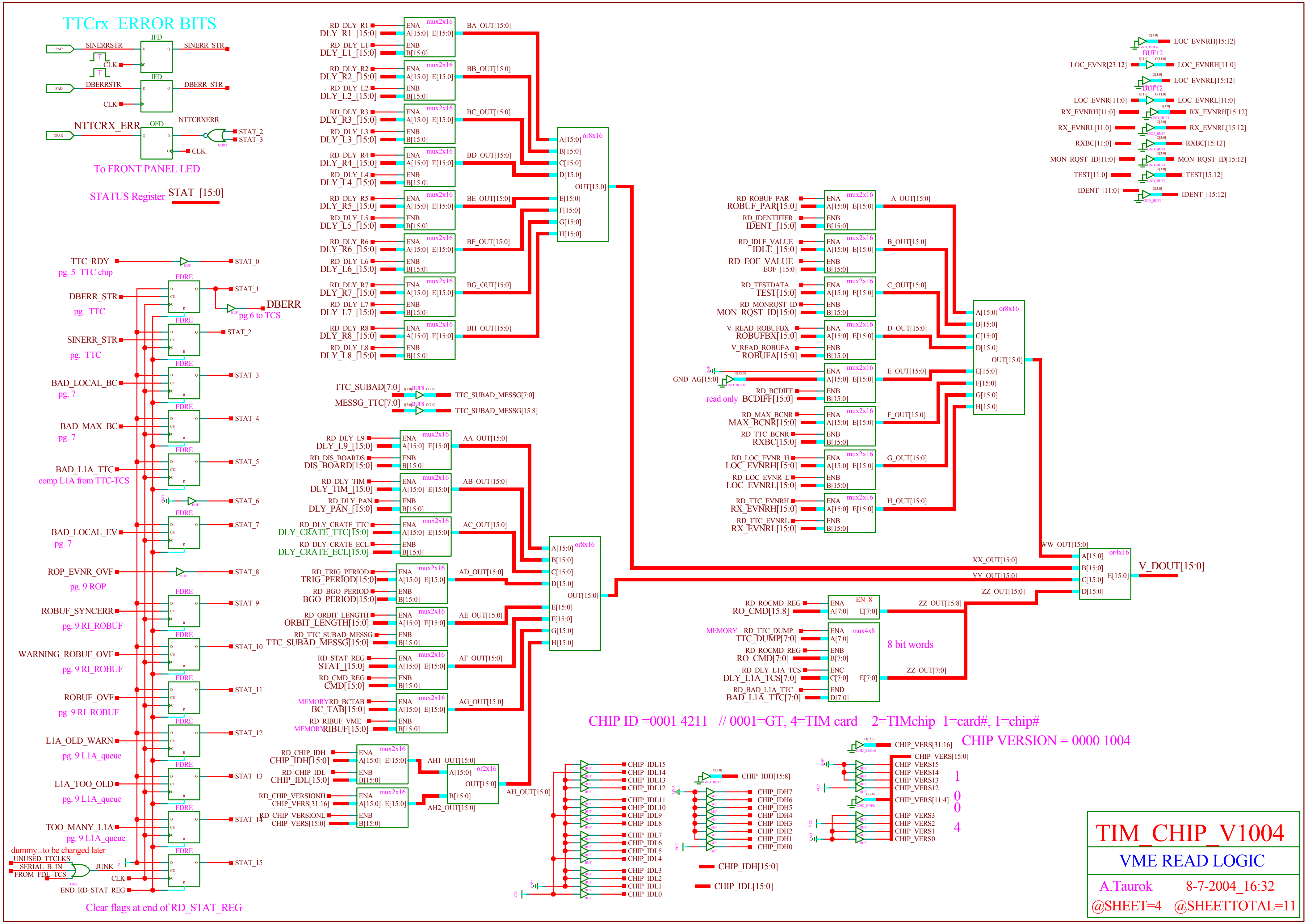
TIM_CHIP_V1004

VME DECODER, IO

A.Taurok   8-7-2004_16:32

@SHEET=2   @SHEETTOTAL=11


DELAY REGISTERS
BOARDS L1...L9, R1...R8,TIM, Front Panel
DELAY for L1A signals: bit 15:8DELAY for BCRES and RESET: bit 7:0
DLY_TIM_[15:0]
GT only: used for Readout
DLY_PAN_[15:0]
Front Panel
DLY_L9_[15:0]
DLY_L8_[15:0]
DLY_L7_[15:0]
DLY_L6_[15:0]
DLY_L5_[15:0]
DLY_L4_[15:0]
DLY_L3_[15:0]
DLY_L2_[15:0]
DLY_L1_[15:0]
Crate delays for ECL ORBIT and BCRES fromTTC
DELAY crate:_ecl, _ttc done by Memories
DLY_CRATE_TTC[15:0]
DLY_CRATE_ECL[15:0]
DIS_BOARD[15:0]
DISable bits
DLY_R8_[15:0]
DLY_R7_[15:0]
DLY_R6_[15:0]
DLY_R5_[15:0]
DLY_R4_[15:0]
DLY_R3_[15:0]
DLY_R2_[15:0]
DLY_R1_[15:0]
Registers for ROP_EV
GT only
IDENT_[11:0]
const. Identifier
IDLE_[15:0]
IDLE pattern
EOF_[15:0]
EOF pattern
ROBUF parameters
NR_OF_ROBUF[1:0]
bits reserved for more ROBUF's
Nr of ROBUFs
ROBUF_PAR[15:0]
RO_LENGTH[7:0]
Readout length (bx's per L1A)
GT only
TEST[11:0]
TEST data via RO-RQST bus
MON_RQST_ID[11:0]
see page 5
Identifier for MON_RQST via RO-RQST bus
Compare both L1A , via TTC and via TCS.
pg.5: check if every TCS_L1A arrives via TTC
DLY_L1A_TCS[7:0]
GT only
TTC_SUBAD[7:0]
PERIODIC SIMULATION
TRIG_PERIOD[15:0]
BGO_PERIOD[15:0]
See Periodic Signal Generator
ORBIT_LENGTH[15:0]
BC Counter
1T pulse to send test data to all ROPs
CMD pulses
RESET_TTCRX
RELEASE_TTCRX
MONRQST_VME
SEND_TESTDATA
L1A_VME
DO_TEST_EN_VME
DO_PRIV_GAP_VME
DO_PRIV_ORBIT_VME
RES_ORBIT_VME
START_RUN_VME
STOP_RUN_VME
EVCNTRES_VME
L1RES_VME
HARD_RES_VME
BCRES_VME
also to PeriodicSigGenerator
also to PeriodicSigGenerator
1x to start BC counters and LHC orbit simulation logic.
RESET_B
RESET_b low active
RESET_TTCrx: set 1, wait xx us, set=0 to load ID into TTCrx
Never clear TTCrx during RUN
CMD Register
CMDREG_D
TIM_SETUPDONE set =1 to tell TCS,that setup of TIM is done
TTC_RDY_VME
CMDREGD_13
CHECK_TTC_CHAIN
disables TTCrx control signals
CMDREG_C
DIS_BOARD_L9
disables L1A,RESET,BCRES output to board L9
DIS_RO_BUS
SEL_EVRES_1
SEL_EVRES_0
def=0 en RoRqst bus
CMDREG_B
SEL_BGO_1
SEL_BGO_0
SEL_BCRES_2
SEL_BCRES_1
1= Selects BGO signals from TCS board (lower priority*)
1= Selects periodic BGO signals from BC-table *) +)
**) 1= Selects BCRES received as ECL signal via the Front Panel
*) default: both bits=0 to take BGO signals from TTC
**) default: all 3 bits=0 to take BCRES from TTC
***) default: all 3 bits=0 to take L1A from TTC
+) also starts the BC table to simulate periodic signals
CMDREG_A
SEL_BCRES_0
SEL_L1A_2
SEL_L1A_1
SEL_L1A_0
1= Selects periodic BCRES from BC_counter
***) 1= Selects L1A from the TCS board (GT only)
***) 1= Selects L1A received as ECL signal via the Front Panel
1= Selects periodic L1A from BC-table +)
CMD[15:0]
RO_CMD[15:0]
RO_CMD Register
RO_CMDREG_C
RO_LINK_ON 1= ChanLink on to send data to GTFE
FREE_RCMD10
FREE_RCMD9
EN_BC_CHECK
Enable Error bits for FastSigs to TCS and to freeze RIBUF
RO_CMDREG_B also set waiting time for TTC_L1A to compare
1= check if every L1A from TCS arrives also via TTC fibre
EN_TTC_CHECK
EN_EVNR_CHECK
EN_L1AQUEUE_CHECK
EN_ROBUF_CHECK
RO_CMDREG_A
FREEZE_RIBUF_IF_ERROR
FREEZE_RINGBUF 1= inhibits data input to RING Buffer
INHIB_L1A_ON_TIM 1= inhibit L1A on TIM board
INVERT_ROPMUX 1= invert clock for ROP-MUX see page 8
RO_CMDREG_D
FREE_RCMD15
FREE_RCMD14
FREE_RCMD13
FREE_RCMD12
CHIP IDENTIFIER
See Readout sheet 4
LHC_orbit length=3564
heavy ion: every 5th tick is a bx
VME Register are never cleared by L1A_RESET or Hard_RESET
TIM_CHIP_V1004
VME REGISTERS
A.Taurok
8-7-2004_16:32
@SHEET=3
@SHEETTOTAL=11

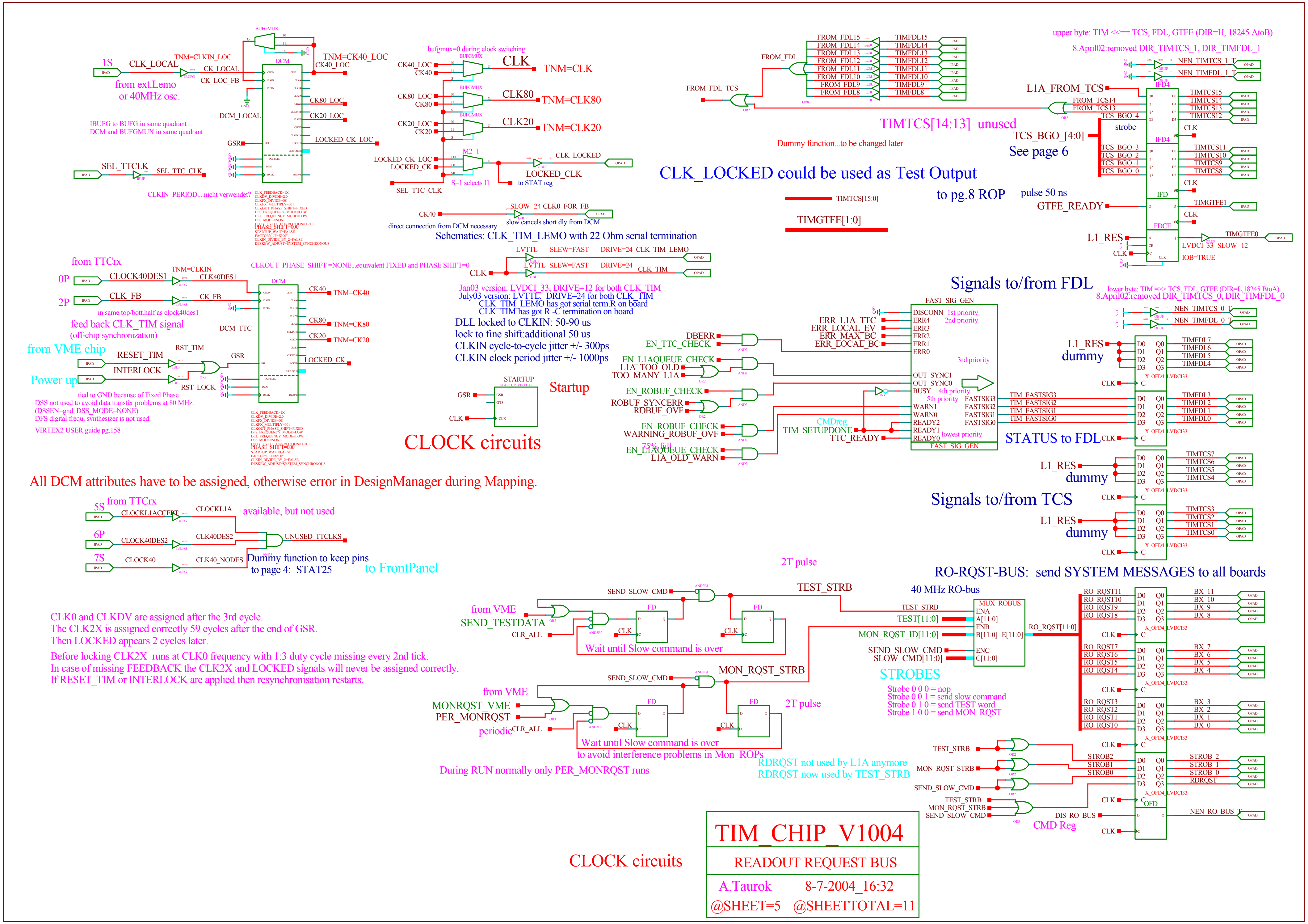


TIM_CHIP_V1004

VME READ LOGIC

A.Taurok 8-7-2004_16:32

@SHEET=4 @SHEETTOTAL=11



All DCM attributes have to be assigned, otherwise error in DesignManager during Mapping.

CLK0 and CLKDV are assigned after the 3rd cycle.
The CLK2X is assigned correctly 59 cycles after the end of GSR.
Then LOCKED appears 2 cycles later.
Before locking CLK2X runs at CLK0 frequency with 1:3 duty cycle missing every 2nd tick.
In case of missing FEEDBACK the CLK2X and LOCKED signals will never be assigned correctly.
If RESET_TIM or INTERLOCK are applied then resynchronisation restarts.

Jan03 version: LVDCI_33, DRIVE=12 for both CLK_TIM
July03 version: LVTTL DRIVE=24 for both CLK_TIM
CLK_TIM_LEMO has got serial term.R on board
CLK_TIM has got R -C termination on board

DLL locked to CLKIN: 50-90 us
lock to fine shift:additional 50 us
CLKIN cycle-to-cycle jitter +/- 300ps
CLKIN clock period jitter +/- 1000ps

tied to GND because of Fixed Phase
DSS not used to avoid data transfer problems at 80 MHz.
(DSSEN=gnd, DSS_MODE=NONE)
DFS digital frequ. synthesizer is not used.
VIRTEX2 USER guide pg.158

IBUFG to BUFG in same quadrant
DCM and BUFGMUX in same quadrant

Schematics: CLK_TIM_LEMO with 22 Ohm serial termination

CLK_LOCKED could be used as Test Output

Dummy function...to be changed later

upper byte: TIM <== TCS, FDL, GTFE (DIR=H, 18245 AtoB)
8.April02:removed DIR_TIMTCS_1, DIR_TIMFDL_1

lower byte: TIM => TCS, FDL, GTFE (DIR=L,18245 BtoA)
8.April02:removed DIR_TIMTCS_0, DIR_TIMFDL_0

Wait until Slow command is over
to avoid interference problems in Mon_ROPs

During RUN normally only PER_MONRQST runs

RDRQST not used by L1A anymore
RDRQST now used by TEST_STRB

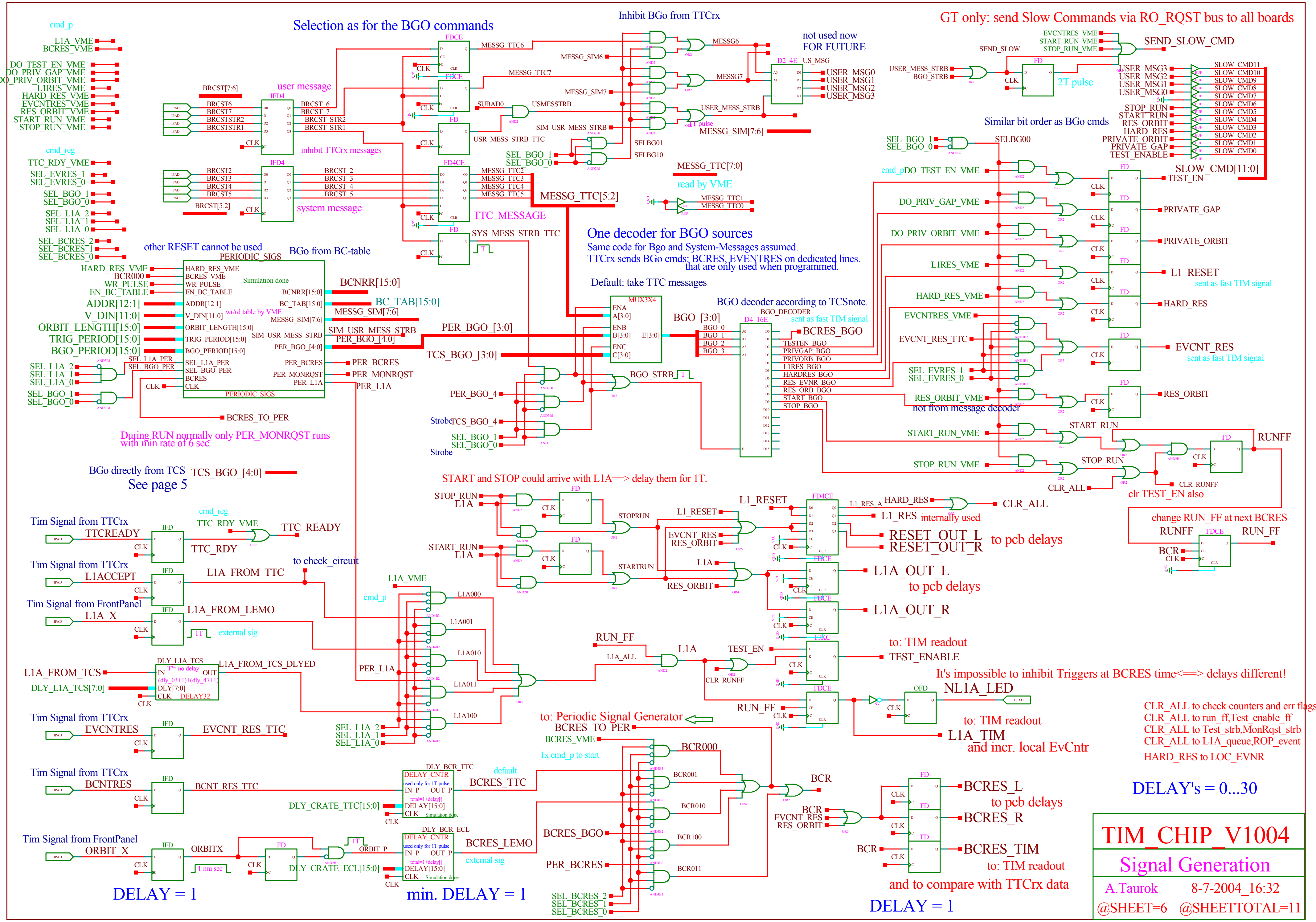
Strobe 0 0 0 = nop
Strobe 0 0 1 = send slow command
Strobe 0 1 0 = send TEST word
Strobe 1 0 0 = send MON_RQST

TIM_CHIP_V1004

READOUT REQUEST BUS

A.Taurok 8-7-2004_16:32

@SHEET=5 @SHEETTOTAL=11



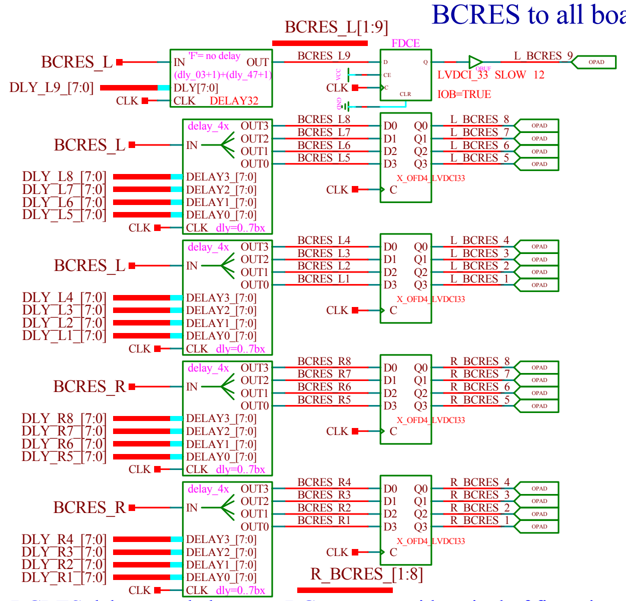
TIM_CHIP_V1004

Signal Generation

A.Taurok 8-7-2004_16:32

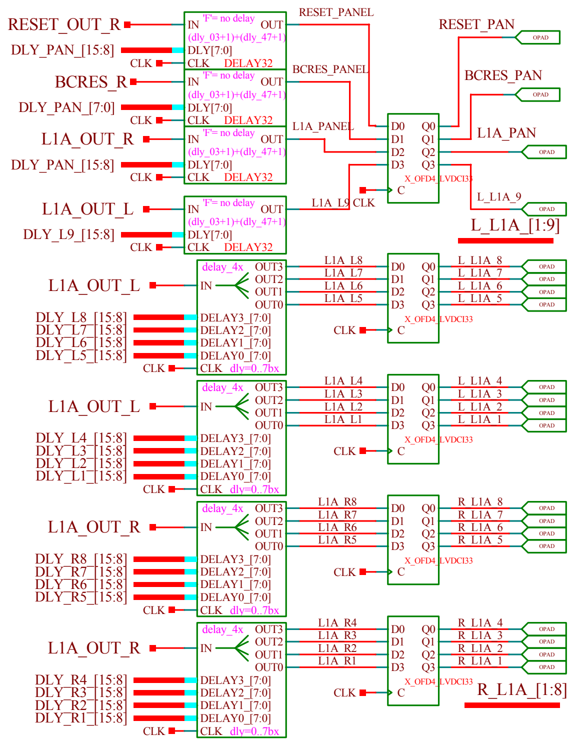
@SHEET=6 @SHEETTOTAL=11

BCRES to all boards



BCRES delays needed to start BC counters with arrival of first trigger word.

to FrontPanel

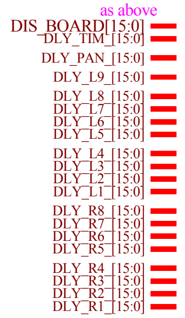


L1A to all boards

L1A delays needed in case of pipeline memory to take data at right time.
GT crate uses RING BUFFERS and doesn't need the L1A delays.
GT crate: set all delays for L1A =0 to get them as early as possible.

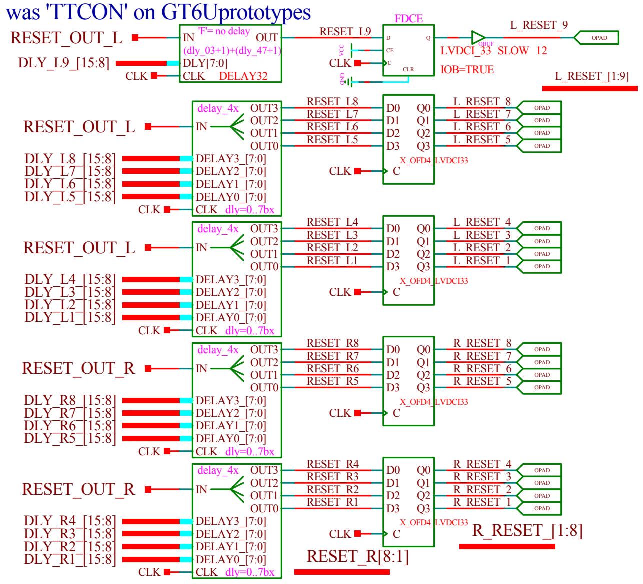
DELAY REGISTERS

BCRES delay: bit 7:0
Total delay = (delay_03+ 1) + (delay_47+ 1)
delay_03: value of bit 3:0
delay_47: value of bit 7:4
'F' = no delay
L1A delay: bit 15:8
as above



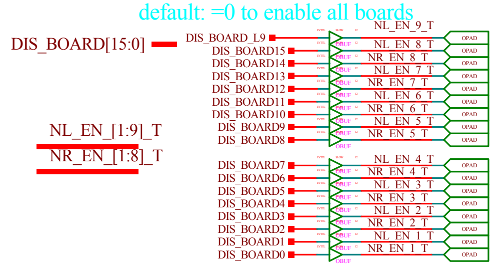
RESET was 'TTCON' on GT6Uprototypes

RESET to all boards



Disable LVDS drivers for TIM SIGNALS

default: =0 to enable all boards



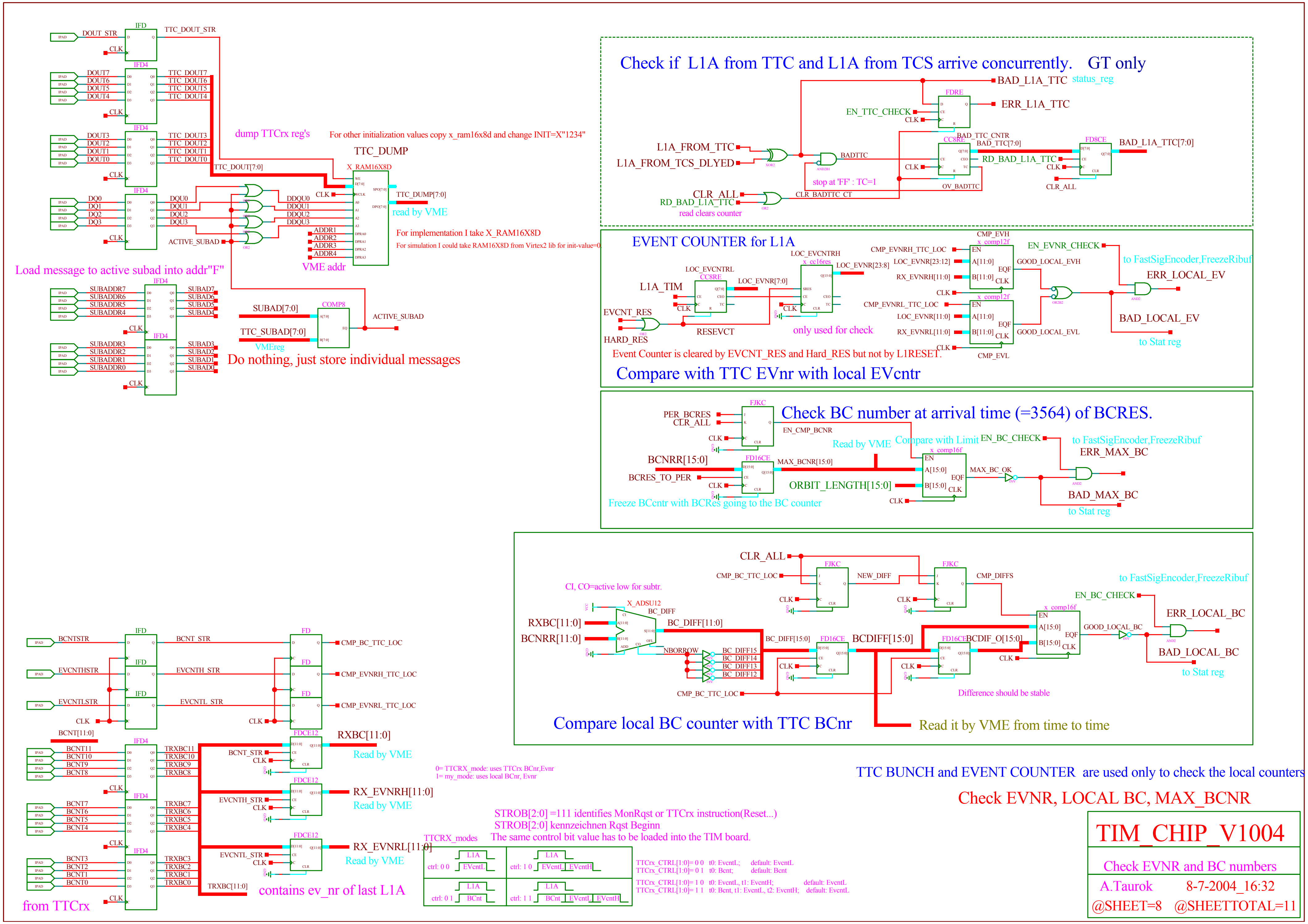
Encoded L1A and RESET signals
to send EVCNT_RES as '11'
Therefore RESET and L1A with same delays!

FAST TIM SIGNALS
0 0 NOP
0 1 RESET
1 0 L1A
1 1 EVCNT_RES

Use the same delay for BCRES and RESET.
Each board with individual delay.

BCRES, RESET, L1A delays simulated by A.T. 17.12.02
x_ofd4_lvdci33 added by A.T. 9 Jan 03

TIM_CHIP_V1004	
FAST TIM SIGNALS to Backplane	
A.Taurok	8-7-2004_16:32
@SHEET=7	@SHEETTOTAL=11



Check if L1A from TTC and L1A from TCS arrive concurrently. GT only
BAD_L1A_TTC
status_reg
ERR_L1A_TTC
EN_TTC_CHECK
L1A_FROM_TTC
L1A_FROM_TCS_DLYED
BADTTC
BAD_TTC_CNTR
BAD_TTC[7:0]
BAD_L1A_TTC[7:0]
RD_BAD_L1A_TTC
stop at 'FF' : TC=1
OV_BADTTC
CLR_ALL
RD_BAD_L1A_TTC
read clears counter
CLR_BADTTC_CT
EVENT COUNTER for L1A
L1A_TIM
EVCNT_RES
HARD_RES
RESEVCT
LOC_EVCNTRL
LOC_EVCNTRH
LOC_EVNR[7:0]
LOC_EVNR[23:8]
only used for check
CMP_EVNRH_TTC_LOC
LOC_EVNR[23:12]
RX_EVNRH[11:0]
CMP_EVNRL_TTC_LOC
LOC_EVNR[11:0]
RX_EVNRL[11:0]
CMP_EVH
CMP_EVL
GOOD_LOCAL_EVH
GOOD_LOCAL_EVL
EN_EVNR_CHECK
to FastSigEncoder,FreezeRibuf
ERR_LOCAL_EV
BAD_LOCAL_EV
to Stat reg
Event Counter is cleared by EVCNT_RES and Hard_RES but not by L1RESET.
Compare with TTC EVnr with local EVcntr
Check BC number at arrival time (=3564) of BCRES.
PER_BCRES
CLR_ALL
EN_CMP_BCNR
BCNRR[15:0]
BCRES_TO_PER
MAX_BCNR[15:0]
Read by VME
Compare with Limit
EN_BC_CHECK
to FastSigEncoder,FreezeRibuf
ERR_MAX_BC
ORBIT_LENGTH[15:0]
MAX_BC_OK
BAD_MAX_BC
to Stat reg
Freeze BCcntr with BCRes going to the BC counter
CLR_ALL
CMP_BC_TTC_LOC
NEW_DIFF
CMP_DIFFS
CI, CO=active low for subtr.
X_ADSU12
BC_DIFF
RXBC[11:0]
BCNRR[11:0]
BC_DIFF[11:0]
NBORROW
BC_DIFF15
BC_DIFF14
BC_DIFF13
BC_DIFF12
BC_DIFF[15:0]
BCDIFF[15:0]
BCDIF_O[15:0]
EN_BC_CHECK
GOOD_LOCAL_BC
ERR_LOCAL_BC
BAD_LOCAL_BC
to FastSigEncoder,FreezeRibuf
to Stat reg
Difference should be stable
Compare local BC counter with TTC BCnr
Read it by VME from time to time
DOUT_STR
TTC_DOUT_STR
DOUT7 DOUT6 DOUT5 DOUT4
TTC_DOUT7 TTC_DOUT6 TTC_DOUT5 TTC_DOUT4
DOUT3 DOUT2 DOUT1 DOUT0
TTC_DOUT3 TTC_DOUT2 TTC_DOUT1 TTC_DOUT0
TTC_DOUT[7:0]
dump TTCrx reg's
For other initialization values copy x_ram16x8d and change INIT=X"1234"
TTC_DUMP
X_RAM16X8D
TTC_DUMP[7:0]
read by VME
DQ0 DQ1 DQ2 DQ3
DQU0 DQU1 DQU2 DQU3
DDQU0 DDQU1 DDQU2 DDQU3
ACTIVE_SUBAD
ADDR1 ADDR2 ADDR3 ADDR4
For implementation I take X_RAM16X8D
For simulation I could take RAM16X8D from Virtex2 lib for init-value=0
VME addr
Load message to active subad into addr"F"
SUBADDR7 SUBADDR6 SUBADDR5 SUBADDR4
SUBAD7 SUBAD6 SUBAD5 SUBAD4
SUBADDR3 SUBADDR2 SUBADDR1 SUBADDR0
SUBAD3 SUBAD2 SUBAD1 SUBAD0
SUBAD[7:0]
COMP8
ACTIVE_SUBAD
TTC_SUBAD[7:0]
VMEreg
Do nothing, just store individual messages
BCNTSTR
BCNT_STR
CMP_BC_TTC_LOC
EVCNTHSTR
EVCNTH_STR
CMP_EVNRH_TTC_LOC
EVCNTLSTR
EVCNTL_STR
CMP_EVNRL_TTC_LOC
BCNT[11:0]
BCNT11 BCNT10 BCNT9 BCNT8
TRXBC11 TRXBC10 TRXBC9 TRXBC8
BCNT7 BCNT6 BCNT5 BCNT4
TRXBC7 TRXBC6 TRXBC5 TRXBC4
BCNT3 BCNT2 BCNT1 BCNT0
TRXBC3 TRXBC2 TRXBC1 TRXBC0
TRXBC[11:0]
RXBC[11:0]
Read by VME
RX_EVNRH[11:0]
Read by VME
RX_EVNRL[11:0]
Read by VME
contains ev_nr of last L1A
from TTCrx
0= TTCRX_mode: uses TTCrx BCnr,Evnr
1= my_mode: uses local BCnr, Evnr
STROB[2:0] =111 identifies MonRqst or TTCrx instruction(Reset...)
STROB[2:0] kennzeichnen Rqst Beginn
The same control bit value has to be loaded into the TIM board.
TTCRX_modes
ctrl: 0 0 L1A EVcntL
ctrl: 1 0 L1A EVcntL EVcntH
ctrl: 0 1 L1A BCnt
ctrl: 1 1 L1A BCnt EVcntL EVcntH
TTCrx_CTRL[1:0]= 0 0 t0: EventL; default: EventL
TTCrx_CTRL[1:0]= 0 1 t0: Bcnt; default: Bcnt
TTCrx_CTRL[1:0]= 1 0 t0: EventL, t1: EventH; default: EventL
TTCrx_CTRL[1:0]= 1 1 t0: Bcnt, t1: EventL, t2: EventH; default: EventL
TTC BUNCH and EVENT COUNTER are used only to check the local counters
Check EVNR, LOCAL BC, MAX_BCNR

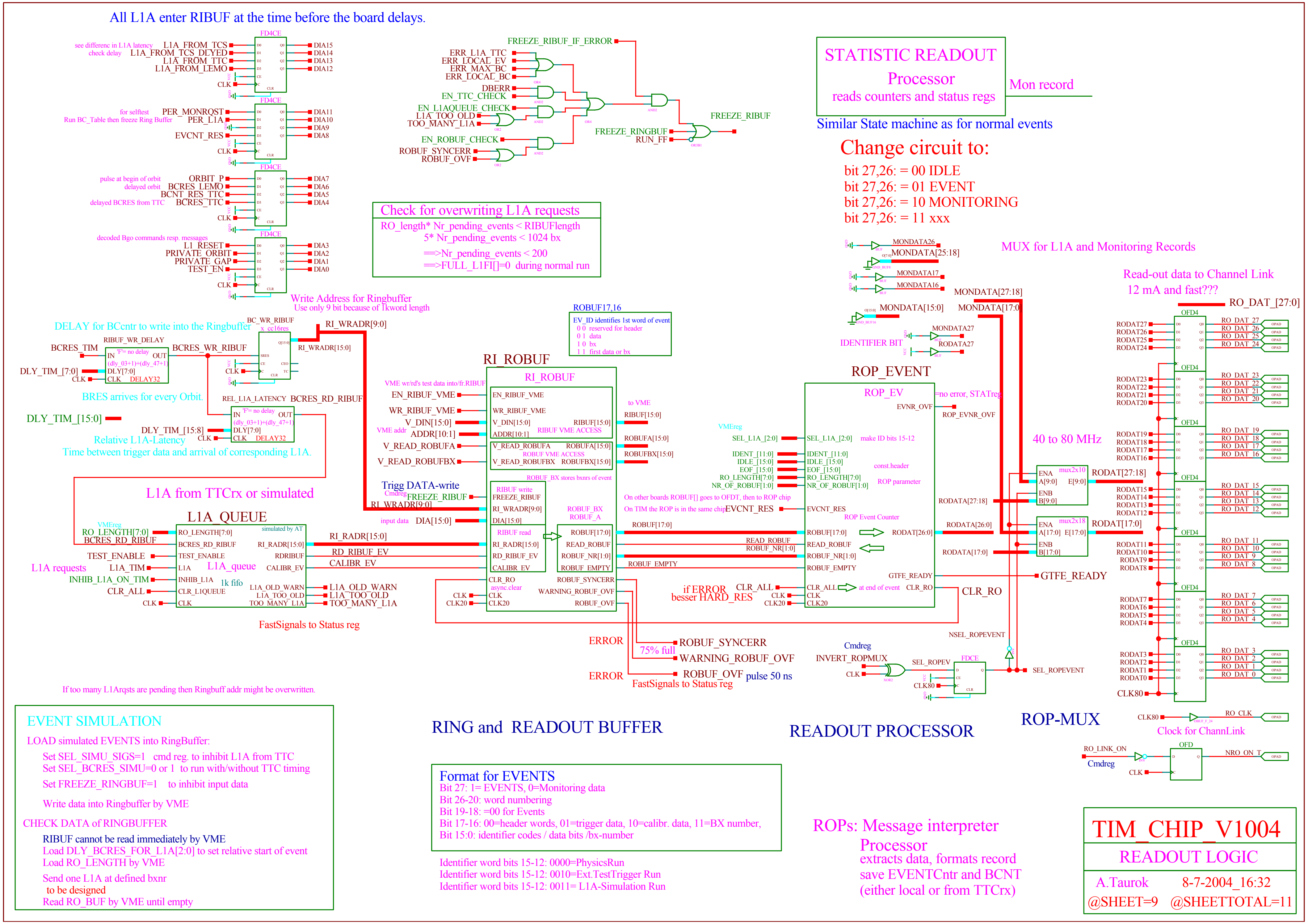
TIM_CHIP_V1004

Check EVNR and BC numbers

A.Taurok 8-2004_16:32

@SHEET=8 @SHEETTOTAL=11

All L1A enter RIBUF at the time before the board delays.



EVENT SIMULATION

LOAD simulated EVENTS into RingBuffer:

Set SEL_SIMU_SIGS=1 cmd reg. to inhibit L1A from TTC

Set SEL_BCRES_SIMU=0 or 1 to run with/without TTC timing

Set FREEZE_RINGBUF=1 to inhibit input data

Write data into Ringbuffer by VME

CHECK DATA of RINGBUFFER

RIBUF cannot be read immediately by VME

Load DLY_BCRES_FOR_L1A[2:0] to set relative start of event

Load RO_LENGTH by VME

Send one L1A at defined bxnr

to be designed

Read RO_BUF by VME until empty

Format for EVENTS

Bit 27: 1= EVENTS, 0=Monitoring data

Bit 26-20: word numbering

Bit 19-18: =00 for Events

Bit 17-16: 00=header words, 01=trigger data, 10=calibr. data, 11=BX number,

Bit 15:0: identifier codes / data bits /bx-number

Identifier word bits 15-12: 0000=PhysicsRun

Identifier word bits 15-12: 0010=Ext.TestTrigger Run

Identifier word bits 15-12: 0011= L1A-Simulation Run

ROPs: Message interpreter

Processor

extracts data, formats record

save EVENTCntr and BCNT

(either local or from TTCrx)

TIM_CHIP_V1004

READOUT LOGIC

A.Taurok 8-7-2004_16:32

@SHEET=9 @SHEETTOTAL=11

BROADCAST COMMANDS.

BCNTRES, EVCNTRES reset also internal TTCrx counters.
Broadcast message: BRCST[0] = BCNTRES delayed by coarse dly[3:0], pulse=1bx
Broadcast message: BRCST[1] = EVCNTRES delayed by coarse dly[3:0], pulse=1bx
System brdcast message: BRCST[5:2] delayed by coarse dly[3:0], synchronous to CLK40DES1, =register output
User broadcast message: BRCST[7:6] delayed by coarse dly[7:4], synchronous to CLK40DES2 or 1, =register output

INDIVIDUAL COMMANDS: 14 bit ID used
2 Fine Dly regs, coarse delay reg, control reg
INDIVIDUAL COMMANDS to all TTCrx: ID=0
INTERNAL COMMANDS: 14 bit ID used
ERDUMP: sends int.err.counters to DOUT[7:0], DQ=1..4, DoutStr
CRDUMP: sends int.regs to DOUT[7:0], DQ=5..a, DoutStr
RESET via TTC: afterwards send BCNTRES and EVCNTRES to synchronise TIM to TTCrx chip.

CLEAR_ALL Hard Reset: reset all error counters and error flags
Hard Reset: clear buffers, stop data transfers
Hard Reset: go into idle state
Hard Reset: Clear RO-BUF memories

VME instructions in TIM chip:
Reset TTCrx chip
REGs
I2C acces via VME and via external frontside connector
2 I2Cadresses Find I2C bus definition!!!!

TTC COMMANDS
Decode individual TTC instructions (DOUT,SUBADDR,DQ,DOUTStr,) defined by us
Decode broadcast TTC instructions: defined by...see CalibrWorkingGroup
Send Reset over RO-rqst bus

READOUT BUS
L1A, Mon requests
Commands: Reset etc

Monitoring Readout Request logic:
insert Monitoring readout request

Simulate LHC orbit: BCNTRES resets Local BunchCounter
BCNTRes: reset local BCNTR, send it to all boards, delay it by n-bx ??
BCNTRes makes the local GT-time
Check if local BCNT=3564 when BCNTRes arrives.

Simulate L1A signals periodically, aligned to BCNT, immediately by VME instr.
BCNT-table

FAST OUTPUTS
CLK40, BCNTRES, TTCON, L1A

L1A Readout Request logic:
send L1A readout request
DEFAULT: L1A only with Ev-cnter to get min. dead time
Check if local EVcntr agrees with TTC-evnr.
Add local BCnr, because TTC-bcnr doeas not arrive in default mode.
BCNT[11:0] :bx-number of L1A, compare it to local BCntr (=not default, maybe in test mode)
..but I don't know which mode is running when L1A arrives!! see pg 24 of TTCrx_manual
If there is no L1A, BCNT[11:0]= depends from ControlReg[1:0]
TIM monitoring:
store L1A's arrival times.
error by L1A overflow
warning by L1A overflow

SPECIAL Virtex2 PINS:

			see JTAG schematic
M0	io/ INIT_B	CCLK	TCK
M1	io/ DOUT	PROG_B	TDI
M2	io/ D0	DONE	TDO
			TMS

HSWAP_EN
PWRDWN_B
DXN
DXP
VBATT
RSVD

io/ VRN_x x=bank_nr
io/ VRP_x x=bank_nr
io/ VREF_x x=bank_nr

VCCAUX 8pins
VCCINT xx pins
VCCO ...xx pins per bank

io/ GCLK0,2,4,6S or P
io/ GCLK1,3,5,7P or S
io/ rdwr_b, cs_b, d7..0

M2	M1	M0	
0	0	0	MASTER SERIAL
1	1	1	SLAVE SERIAL
1	0	1	BOUNDARY SCAN

TTCrx monitoring:
Write TTCrx register contents into registers/mem? (DOUT,DQ,DOUTStr)
Set ERR flags after DBErrStr, SINErrStr (err-counter?)

CLOCK circuit: BUFGMUX: switch between 2 clocks!

I will use fine delay to adjust phase of GT to the TTCvi's
Test: compare L1A _fromTCS with L1A received in TIM by up/down counter

PACKAGE OPTIONS:
XC2V500 FG456 264io not enough
XC2V1000 FG456 324 io 1mm
XC2V1000 BG575 328 io 1.27mm
XC2V1500 FG676 392 io 1mm

according to trigger pipeline

DRIVERS with TERMINATION
OBUF(T) attribute: IOSTANDARD LVDCI_33 //for Zo, VCCO=3.3V
OBUF(T) attribute: IOSTANDARD LVDCI_DV2_33 //for Zo/2, VCCO=3.3V
IBUF attribute: IOSTANDARD LVDCI_33 //for Zo, VCCO= N/A
IBUF attribute: IOSTANDARD LVDCI_DV2_33 //for Zo/2, VCCO=3.3V
4 - 5 power/gnd pairs per bank
LVTTL 12 mA fast: <10 drivers/ pwr_gnd pair
LVDCI_33 with 50 Ohm: <13 drivers/ pwr_gnd pair
TIM chip: up to 40 outputs per bank

TIM_CHIP FUNCTIONS:

Use of CLKL1A is unclear to me.
...disable it in TTCrx chip!

ICAP_VIRTEX2..provides access to internal configuration
?? for partial reprogramming ??

TIM_CHIP_V1004
Remarks
A.Taurok 8-7-2004_16:32

MESSAGES

L1A_Reset=reset Readout Buffers
L1A_Reset   after Out_of_Sync
Hard_Reset   after Error: reset StateMachines, reload FPGAs from Prom??

StartTrigger, Stop Trigger...send it to TCS!!!!

TriggType ??
BC_Res
BC0 after BCRes
EvCntrRes
24 bit EventNr

L1A Reset

L1AReset= stop BCntrs? resync. ChannLinks ?

OUTofSync onboard : stores immediately present EvCntrs,BCcntrs, Addrpointers .
(These regs are cleared by reading them during MonRqst of L1AReset)
Send OUTofSync to TCS
Optionally freeze all activity on board for further investigation.

L1AReset=TIM: stop MonRqsts until GT is 'READY' <==TCS
L1AReset= FDL: stop finalORs
L1AReset= all boards: clear Out_ofSync error
L1AReset= ROPs: clear DerandomBuff's + MonRqst procedure

Calibr+Test MESSAGES

TEST Enable: n-bx before L1A (n=150) at predefined bx-nr
TEST Enable: Inhibit normal TRIGGERS ==>TCS
Next L1A removes Inhibit
pg 8
Next L1A make empty Event ==>ROPs
Private GAP: next gap for private use
Private ORBIT: next orbit for private use

ASYNC_RESET
clear robuf
clear error flags, error counters

SYNC_RESET
clear robuf
clear error flags, error counters

CLEAR_ALL
Hard Reset: reset all error counters and error flags
Hard Reset: clear buffers, stop data transfers
Hard Reset: go into idle state

TCS-io
BROADCAST MESSAGES
USER MESSAGES
PRIVATE GAP/ORBIT LOGIC MISSING
Calibr+Test EVENT?....to be DONE

MON_RQST[11:0] fehlt noch

Fast Monitoring Signals to TCS

7 Send HardReset <--- Error_State: DBERR
6 Send L1Reset <---   Out_of_Sync: ???
5 Inhibit L1As <---   Warning Overflow: xxxxx
4 Inhibit L1As <---   Busy: Setup not done
3 Allow L1As <---   Ready: Readout is ready, TTCready,

Additional Fast Signals to TCS

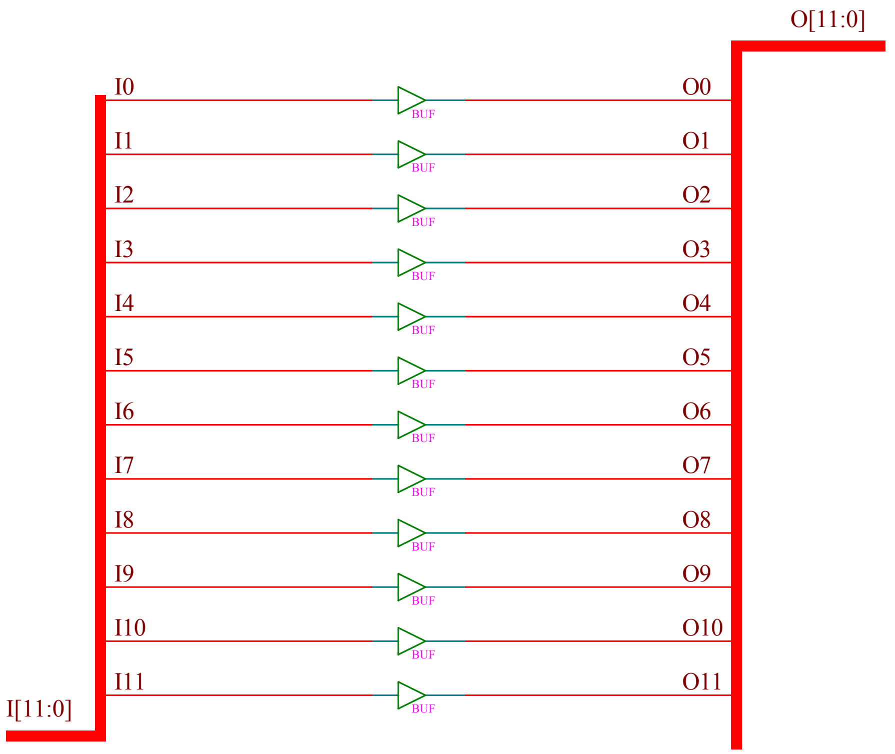
2   unused

TIM<=>TCS
TIM<=>GTFE
TIM<=>FDL

Do not send inhibit sigs to TCS, it knows better
Only stop your own Readout if requested by TCS directly or via TTC

Set BUSY=1, clearBuffers and Pipelines, reset StaeMachines, then BUSY=0,(READY=1?)

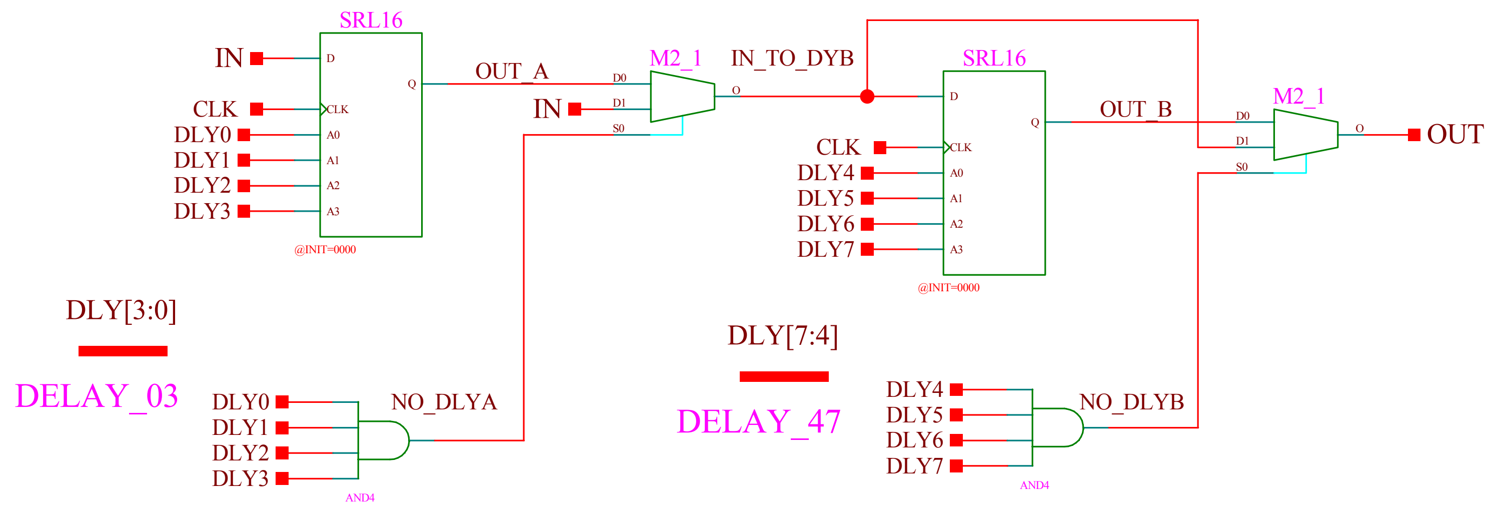
TIM_CHIP_V1004	
REMARKS 2	
A.Taurok	8-7-2004_16:32
@SHEET=11	@SHEETTOTAL=11



Changed to buf12 by A.T. for TIM chip Dec01

Title:	VIRTEX Family BUFE16 Macro
Comments:	16-Bit 3-State Buffer w/ an Active High Enable
Date: 23rd February 1993	Ver: 1
Sheet Size: A	Rev: A

drawn by KS
Copyright (c) 1993, Xilinx Inc.



DLY[7:0]

Total delay = (delay_03+ 1) + (delay_47+ 1)
'F' = no delay

Programmable DELAYs:

0 DFF = FF
1 DFF = 0F, F0
2 DFF = 00, 1F,F1
3 DFF = 01, 10, F2, 2F
4 DFF = 11, 02, 20, 3F,F3

Simulation done by A.T. 31 Oct 02
For simulation add STARTUP symbol and net GSR

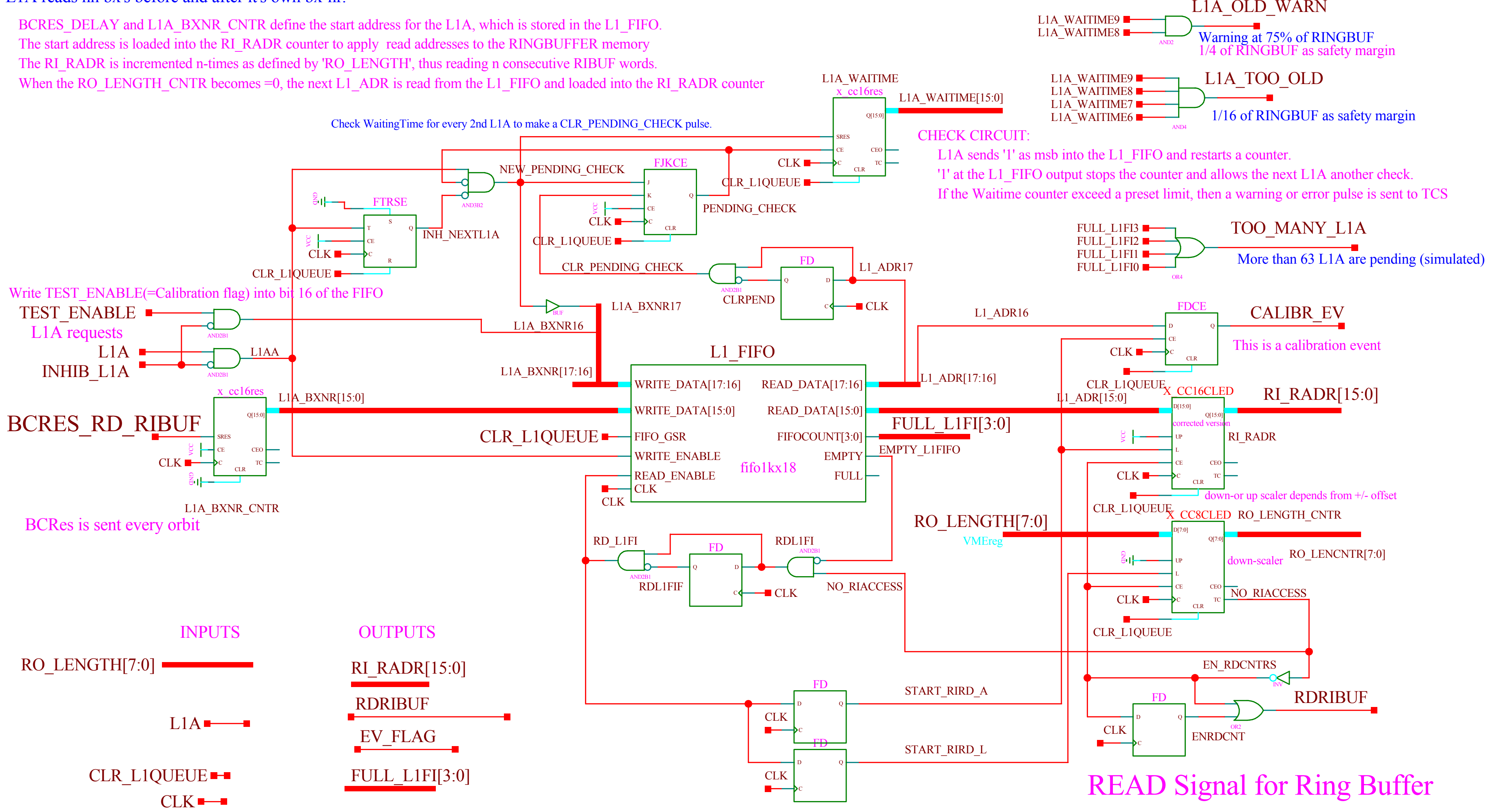
Programmable DELAY for fast signals of the TIM chip

A.T. 31 Oct 02

L1A reads nn bx's before and after it's own bx-nr.

BCRES_DELAY and L1A_BXNR_CNTR define the start address for the L1A, which is stored in the L1_FIFO.
The start address is loaded into the RI_RADR counter to apply read addresses to the RINGBUFFER memory
The RI_RADR is incremented n-times as defined by 'RO_LENGTH', thus reading n consecutive RIBUF words.
When the RO_LENGTH_CNTR becomes =0, the next L1_ADR is read from the L1_FIFO and loaded into the RI_RADR counter

CHECK CIRCUIT:
L1A sends '1' as msb into the L1_FIFO and restarts a counter.
'1' at the L1_FIFO output stops the counter and allows the next L1A another check.
If the Waitime counter exceed a preset limit, then a warning or error pulse is sent to TCS



READ Signal for Ring Buffer

If too many L1A requests are pending then Ringbuff addr might be overwritten.
For the Ringbuffer the read and write frequency is equal to 40MHz.
The Waiting-time of a L1A in the L1A-queue is checked periodically.
If the waiting time becomes too long a warning or error flag will be sent.
Waiting_time + Latency (GT until L1A from TTC) + RingBufferExtractionTime(RO_length) < RINGBUFFER size

Simulated by A.T 9.Dec 02
Add Startup symbol and net GSR for stand-alone simulation.

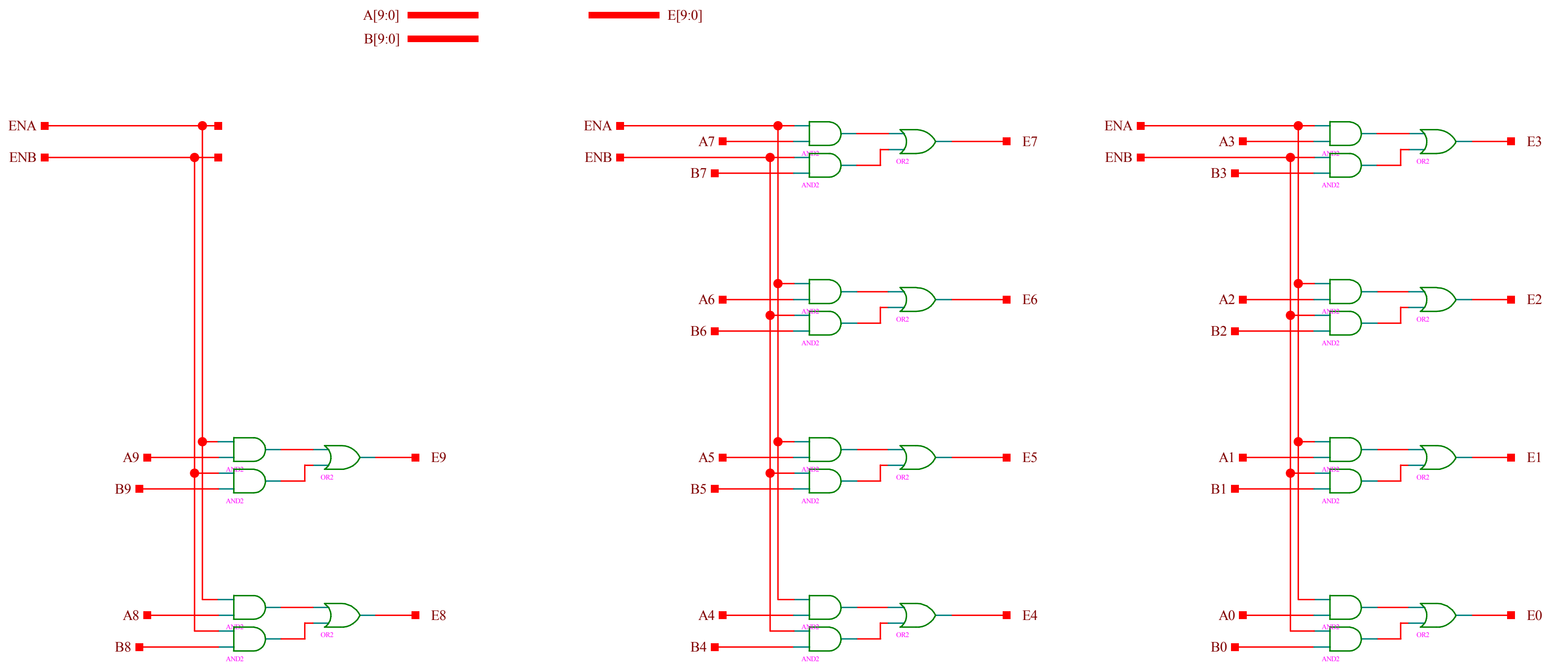
Extraction Length = RO_LENGTH +1(BX)
Extraction Time= RO_LENGTH +1+1 (BX)
10MHz extraction (event) rate for 3 words/L1A

FIFO simulated
WAITIME simulated
RO_LENGTH counter simulated
RI_RADR counter simulated

used in TIM chip

L1A_QUEUE

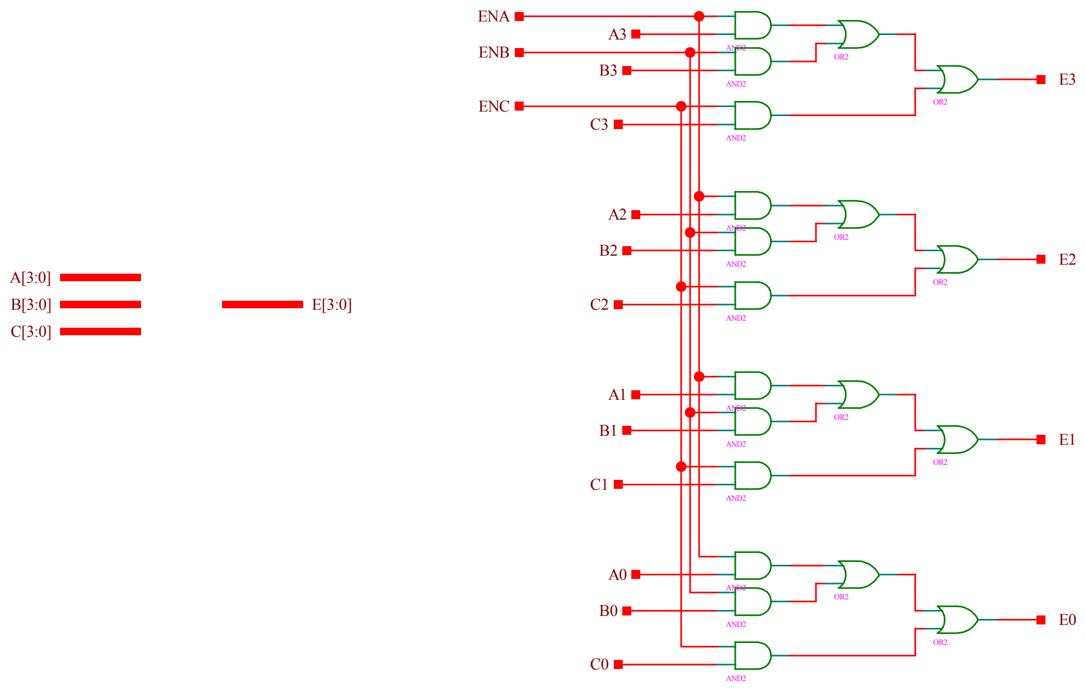
21-10-2003_10:46



for ringbuffer

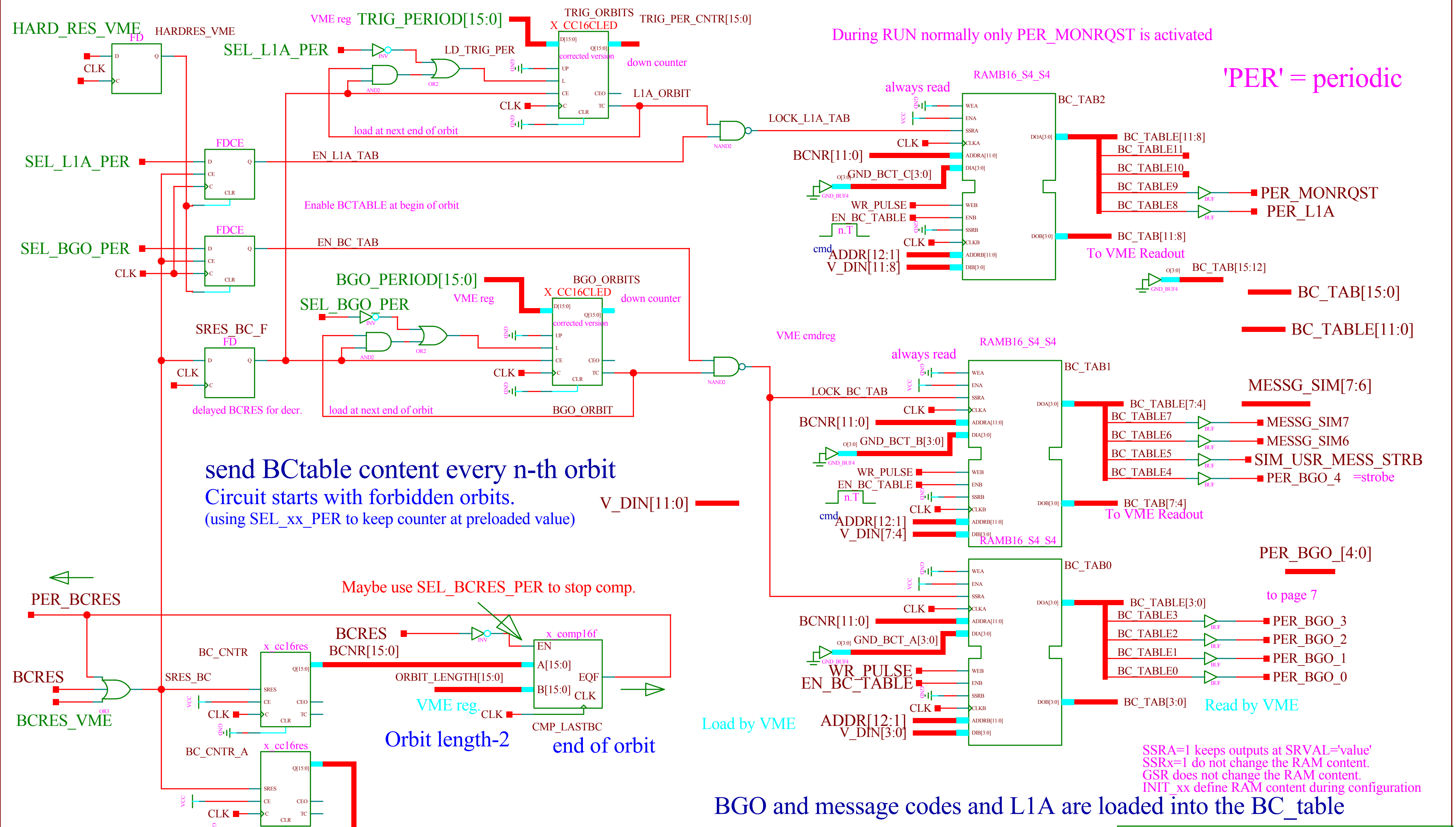
MUX2x10

A.T. 2 dec 01



MUX3x4

A.T. 7.7 03



SSRA=1 keeps outputs at SRVAL='value'
SSRx=1 do not change the RAM content.
GSR does not change the RAM content.
INIT_xx define RAM content during configuration

BGO and message codes and L1A are loaded into the BC_table

For standalone simulation add STARTUP symbol to schematic.

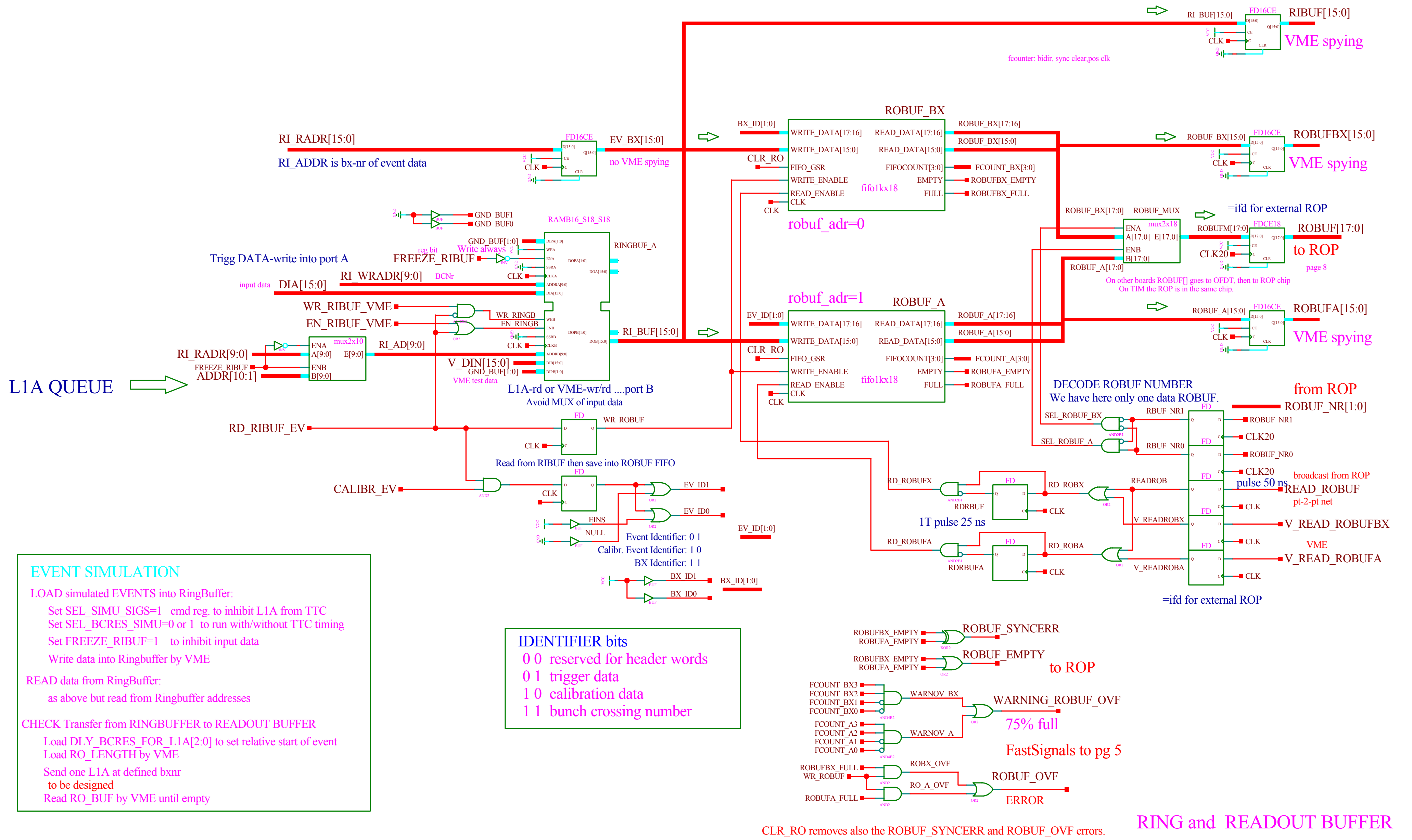
18.11.02. A.T: Logic Simulation done.
10.12.02. A.T: all attribute values assigned; otherwise error in DesignManager
2.1.03. A.T: BCRES_VME added, also to comp to remove 'X'
3.1.03. A.T: HARD_RES and EN_xx_TAB added, to start periodic sigs at begin of orbit.
8.7.03. A.T: disable Comp with BCRES only

PERIODIC_SIGS

Simulate BGo and User Messages,L1A

A.Taurok 8-7-2004_16:32

@SHEET=1 @SHEETTOTAL=1



EVENT SIMULATION

LOAD simulated EVENTS into RingBuffer:

Set SEL_SIMU_SIGS=1 cmd reg. to inhibit L1A from TTC
Set SEL_BCRES_SIMU=0 or 1 to run with/without TTC timing

Set FREEZE_RIBUF=1 to inhibit input data

Write data into Ringbuffer by VME

READ data from RingBuffer:

as above but read from Ringbuffer addresses

CHECK Transfer from RINGBUFFER to READOUT BUFFER

Load DLY_BCRES_FOR_L1A[2:0] to set relative start of event
Load RO_LENGTH by VME

Send one L1A at defined bxnr
to be designed
Read RO_BUF by VME until empty

IDENTIFIER bits

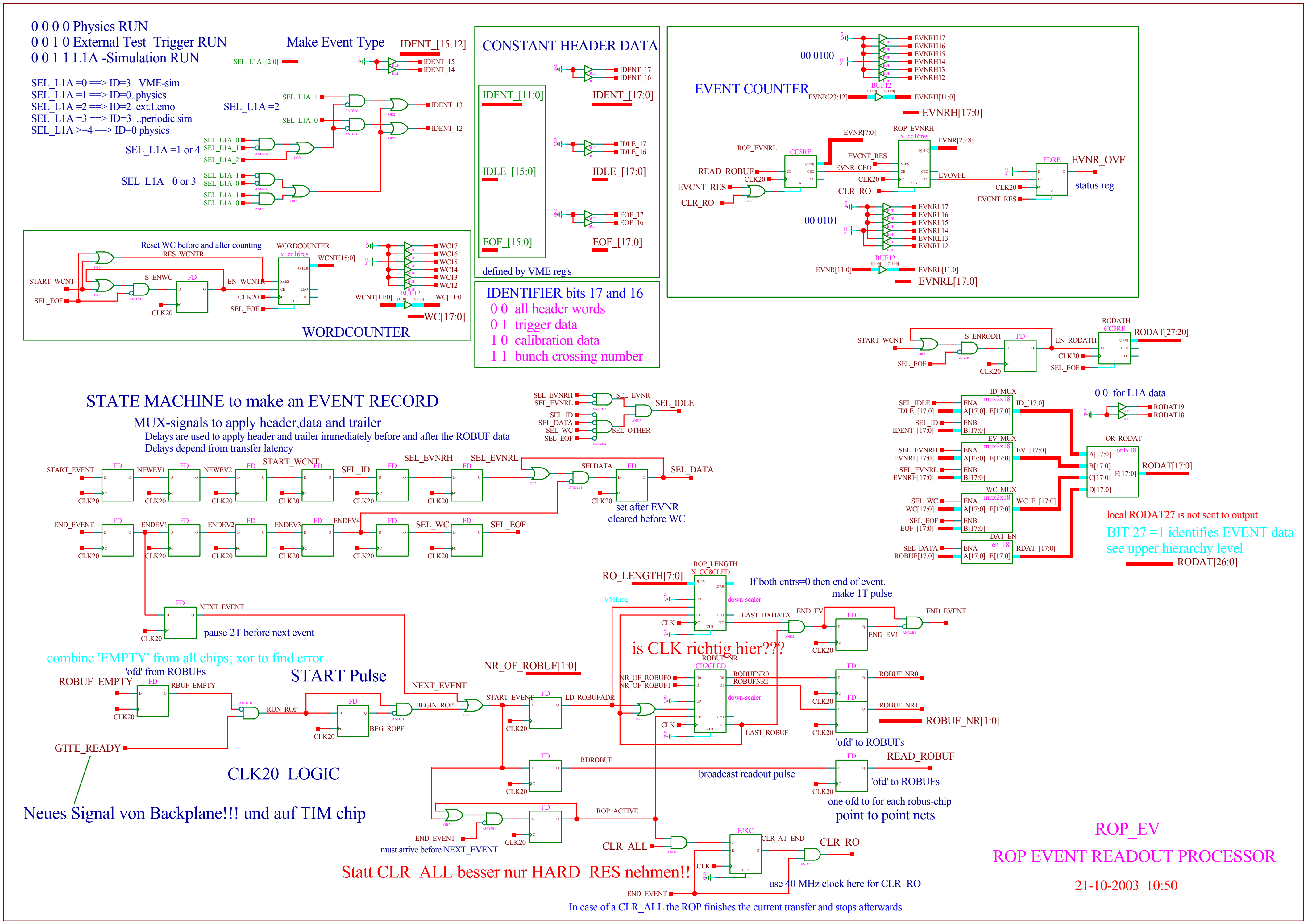
0 0	reserved for header words
0 1	trigger data
1 0	calibration data
1 1	bunch crossing number

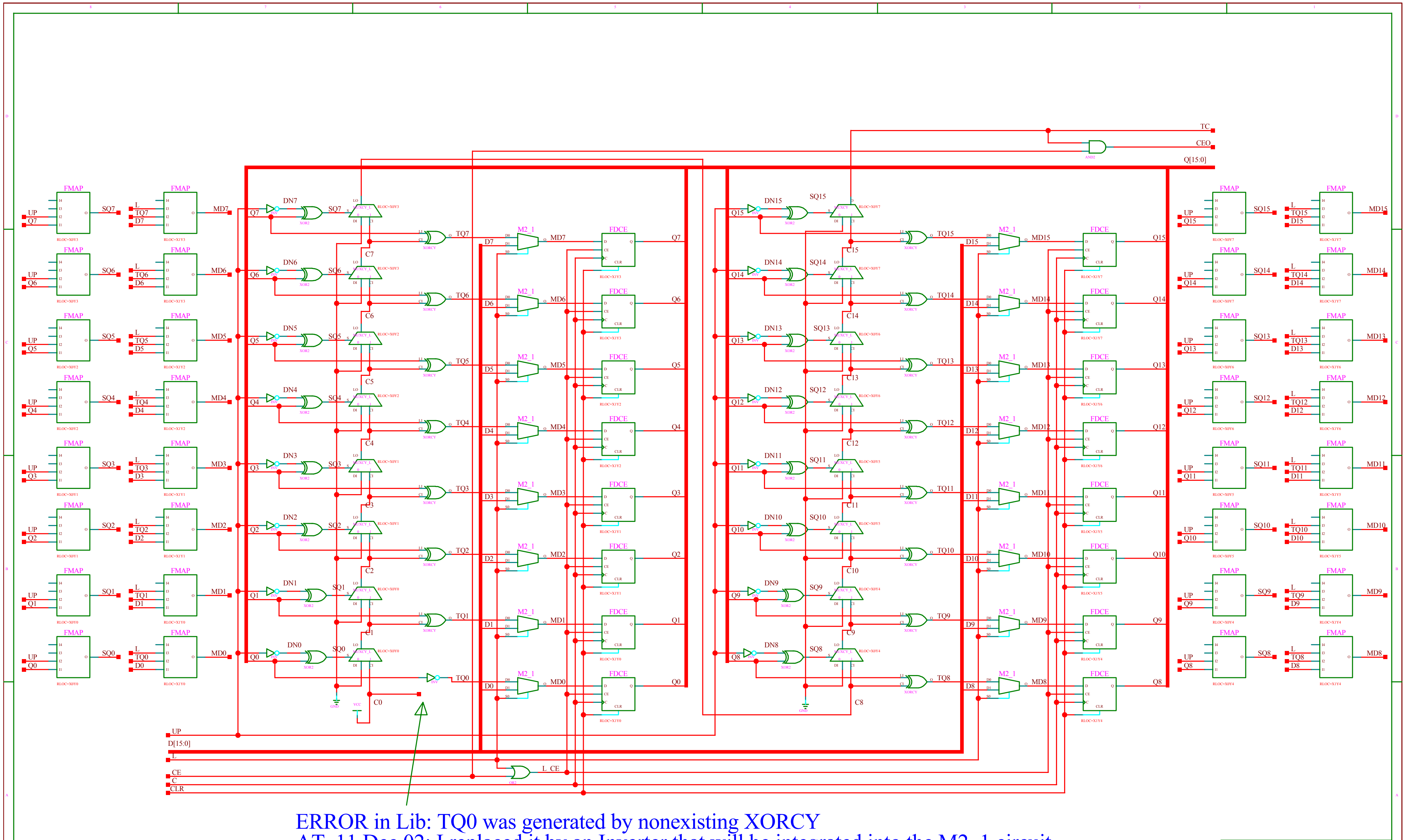
CLR_RO removes also the ROBUF_SYNCERR and ROBUF_OVF errors.

RING and READOUT BUFFER

RI_ROBUF

21-10-2003_10:47


0 0 0 0 Physics RUN
0 0 1 0 External Test Trigger RUN
0 0 1 1 L1A -Simulation RUN
SEL_L1A =0 ==> ID=3 VME-sim
SEL_L1A =1 ==> ID=0..physics
SEL_L1A =2 ==> ID=2 ext.Lemo
SEL_L1A =3 ==> ID=3 ..periodic sim
SEL_L1A >=4 ==> ID=0 physics
Make Event Type
IDENT_[15:12]
SEL_L1A =2
SEL_L1A =1 or 4
SEL_L1A =0 or 3
CONSTANT HEADER DATA
IDENT_[11:0]
IDENT_[17:0]
IDLE_[15:0]
IDLE_[17:0]
EOF_[15:0]
EOF_[17:0]
defined by VME reg's
EVENT COUNTER
00 0100
00 0101
EVNRH[17:0]
EVNRL[17:0]
EVNR_OVF
status reg
IDENTIFIER bits 17 and 16
0 0 all header words
0 1 trigger data
1 0 calibration data
1 1 bunch crossing number
Reset WC before and after counting
WORDCOUNTER
WC[17:0]
STATE MACHINE to make an EVENT RECORD
MUX-signals to apply header,data and trailer
Delays are used to apply header and trailer immediately before and after the ROBUF data
Delays depend from transfer latency
set after EVNR
cleared before WC
0 0 for L1A data
RODAT[27:20]
RODAT[17:0]
local RODAT27 is not sent to output
BIT 27 =1 identifies EVENT data
see upper hierarchy level
RODAT[26:0]
RO_LENGTH[7:0]
If both cntrs=0 then end of event.
make 1T pulse
is CLK richtig hier???
pause 2T before next event
combine 'EMPTY' from all chips; xor to find error
'ofd' from ROBUFs
ROBUF_EMPTY
START Pulse
NR_OF_ROBUF[1:0]
ROBUF_NR[1:0]
'ofd' to ROBUFs
READ_ROBUF
broadcast readout pulse
'ofd' to ROBUFs
one ofd to for each robus-chip
point to point nets
GTFE_READY
CLK20 LOGIC
Neues Signal von Backplane!!! und auf TIM chip
must arrive before NEXT_EVENT
Statt CLR_ALL besser nur HARD_RES nehmen!!
CLR_ALL
CLR_RO
use 40 MHz clock here for CLR_RO
In case of a CLR_ALL the ROP finishes the current transfer and stops afterwards.
ROP_EV
ROP EVENT READOUT PROCESSOR
21-10-2003_10:50

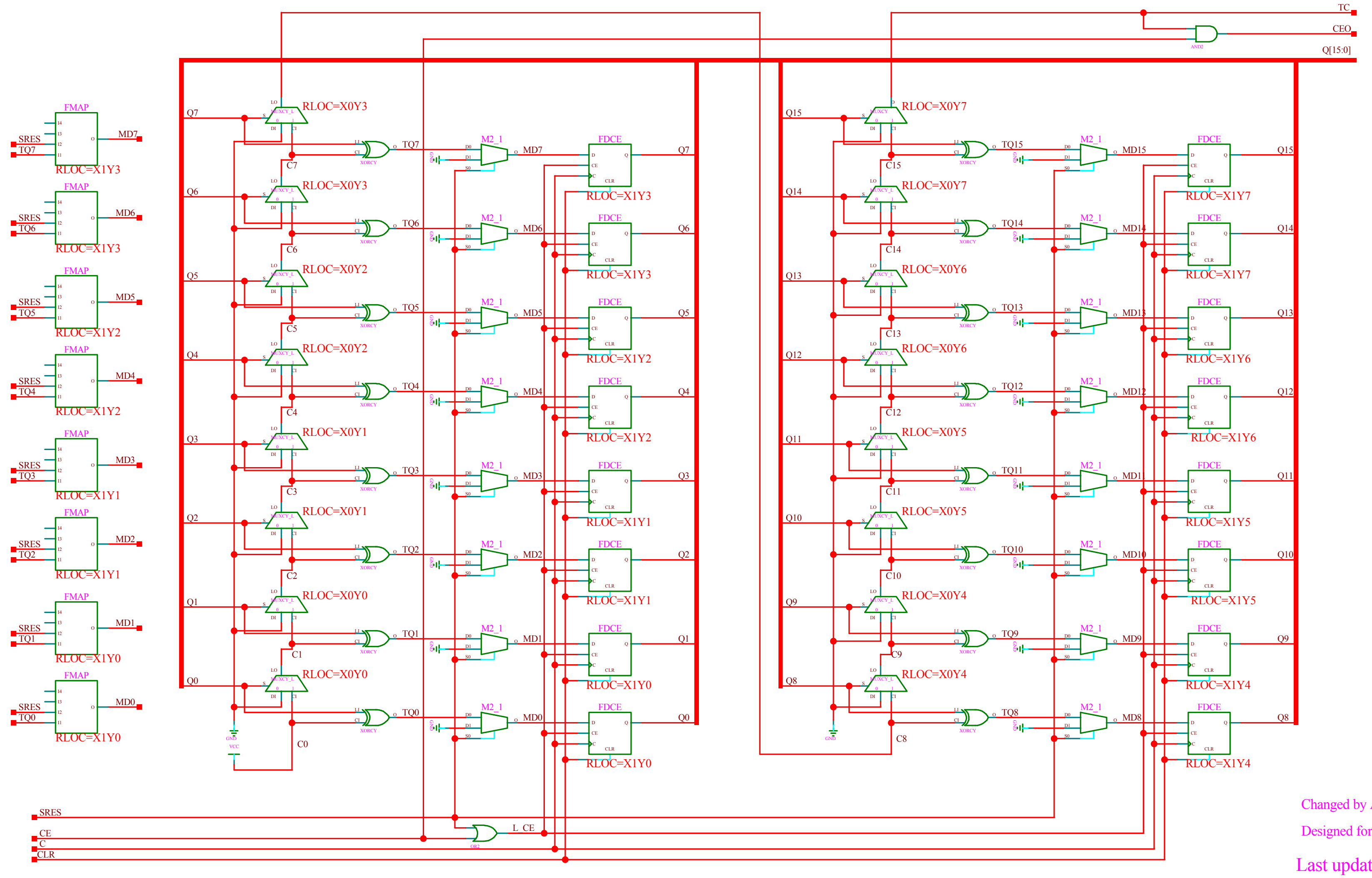

FMAP
UP
L
SQ0 SQ1 SQ2 SQ3 SQ4 SQ5 SQ6 SQ7 SQ8 SQ9 SQ10 SQ11 SQ12 SQ13 SQ14 SQ15
MD0 MD1 MD2 MD3 MD4 MD5 MD6 MD7 MD8 MD9 MD10 MD11 MD12 MD13 MD14 MD15
TQ0 TQ1 TQ2 TQ3 TQ4 TQ5 TQ6 TQ7 TQ8 TQ9 TQ10 TQ11 TQ12 TQ13 TQ14 TQ15
DN0 DN1 DN2 DN3 DN4 DN5 DN6 DN7 DN8 DN9 DN10 DN11 DN12 DN13 DN14 DN15
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15
INV
XOR2
MUXCY_L
XORCY
M2_1
FDCE
GND
VCC
AND2
OR2
TC
CEO
Q[15:0]
D[15:0]
L CE
CE
C
CLR
UP

ERROR in Lib: TQ0 was generated by nonexisting XORCY
AT. 11.Dec 02: I replaced it by an Inverter that will be integrated into the M2_1 circuit

Info Message from XIL4.2 Mapper
INFO:MapLib:149 - Failed to find a MUXCY that associates with XORCY symbol
"$6I548/BGO_ORBITS/$1I6" (output signal=$6I548/BGO_ORBITS/TQ0).
Cannot recognize the standard carry chain structure.

Title:	virtex2 Family CC16CLED Macro
Comments:	16-Bit Casc,Ldble,Dir Bin Counter w/ Clk En & Async Clr (using MUXCY_L)
Date: 1st June 1994	Ver: 1
Sheet Size: D	Rev: A

Copyright (c) 1994, Xilinx Inc.



Changed by AT. to SYNC RESET Sept.01

Designed for TIM chip

Last update: 23-1-2002_16:47

Title:	VIRTEX Family CC16CLE Macro
Comments:	16-Bit Casc,Ldble Bin Counter w/ Clk En & Async Clr (using MUXCY_L)
Date: 31st May 1994	Ver: 1
Sheet Size: D	Rev: A

Copyright (c) 1994, Xilinx Inc.

TTCrq Manual

P. Moreira[*]

CERN - EP/MIC, Geneva Switzerland

November 2004

Version 1.5

[*]Technical contact e-mail: Paulo.Moreira@cern.ch

SUMMARY OF CHANGES;

Version 1.5 — 2005-01-11

I2C terminations configuration table added. See page 11

Version 1.4 — 2004-11-17

TTCrq mezzanine board was redesigned to include an AC coupled resistive divider that reduces the power delivered to the quartz crystal. At the request of the users, as detailed below, some other modifications were introduced to the card. These modifications are fully compatible with the previous version; In particular, all the physical dimensions and connector positions have remained the same.

- Introduction of an AC couple resistive divider to reduce the power delivered to the QPLL quartz crystal.
- Introduction of a 2.5 V power pin on connector J2. See page 6.
- Possibility of internal terminating all of the LVDS signals. See page 6
- TTCrq configuration tables updated. See page 10

Version 1.3 — 2004-11-09

- Footprint of the TTCrq was define and is available through the CERN components library. See page 10.

Version 1.2 — 2004-02-27

- Section on terminating the LVDS clock signals added. See page 6.

RELATED DOCUMENTS

To understand the operation of the TTCrq mezzanine card, the user should be familiar with the functionality of the TTCrx and the QPLL. Documentation for these two devices can be found at the following web addresses:

TTCrx: http://www.cern.ch/TTC/intro.htmll

QPLL: http://www.cern.ch/proj-qpll

Note: updates of this document can be obtained from the QPLL site.

INTRODUCTION

A mezzanine card (the **TTCrq**) was designed by the CERN microelectronics group to replace the TTCrm. The device is supported by the CERN Electronics Pool (EP-ESS Group). For further information on support please refer to the EP-ESS Croup TTC support web page: http://ess.web.cern.ch/ESS/TTCsupport .

The TTCrq can be mounted on a standard VME unit without imposing restrictions on the space between two VME modules. The card contains a TTCrx, a QPLL with its associated crystal and a TrueLight pin-preamplifier (TRR-1B43-000).

The TTCrq mezzanine card is backward-compatible with the TTCrm. That means that the existing electrical connectors (J1 and J2) are kept in the same physical positions with the same pinout as in the TTCrm. An additional connector (J3) is added to the card located on the PCB side opposite to the optical connector side as represented in Figure 1. J3 is a 26-pin connector. (VME board areas under the dotted/shadowed regions (top view drawing) should remain free on the mother board for tool insertion during board removal.)

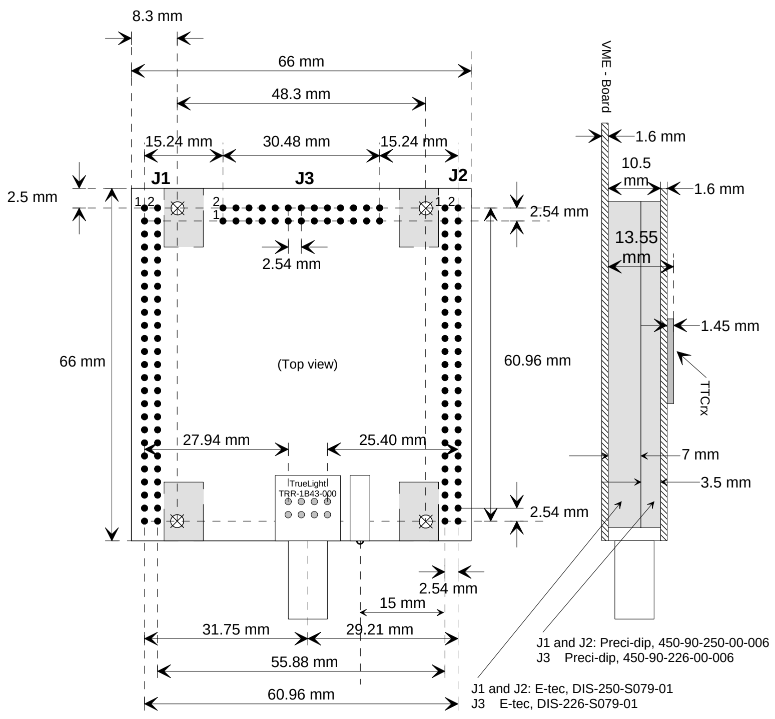


Figure 1 TTCrx and QPLL mezzanine card

To reduce the module height the optical receiver is mounted under the card on the same PCB side as the electrical connectors. The optical connector was moved to the bottom side of the PCB but its distance to the J1 and J2 connectors remained unchanged (please see details in Figure 1).

As shown in Fig. 1, the overall height of the components on the mounted TTCrq mezzanine card is 13.55 mm above the components side of the VME motherboard. To ensure compliance with VMEbus Rule 7.14, assembled VME modules should be measured to verify that the sum of TTCrm component height and board warpage does not exceed 13.71 mm.

The 13.71 mm limit allows a guaranteed 2.44 mm clearance between the TTCrx and the longest component leads on the adjacent VME board. More importantly for the cooling airflow, it allows a nominal 4.91 mm space to an unwarped adjacent VMEboard. According to VMEbus Observation 7.11, this space allows adequate airflow for cooling. However, designers should avoid putting high-dissipation components on the VME board underneath the mezzanine board, as the horizontal orientation of the connectors is likely to restrict airflow for cooling.

CIRCUIT

A block diagram of the mezzanine card is represented in Figure 2. The card contains a pin-preamplifier (the Truelight TRR-1B43-000), a TTCrx, a QPLL, a PROM and a bank of SMD pull-up/pull-down resistors and jumpers to setup the TTCrx address and operation modes. All of the QPLL pins (with the exception of the crystal dedicated pins and the VCXO decoupling capacitor pin) are accessible through the J3 connector. The QPLL input can be taken either from the J3 connector (external source) or from one of the TTCrx clock outputs (Clock40, Clock40Des1 or Clock40Des2). When using the QPLL with an external source the reference clock signal can be either LVDS or CMOS. If necessary, the QPLL clock signal (after LVDS to CMOS conversion) can be routed to the Clock40Des1 signal on the J1 connector. This allows using the TTCrq on boards that were designed to receive the TTCrm card while at same time profiting from the QPLL as a jitter filter.

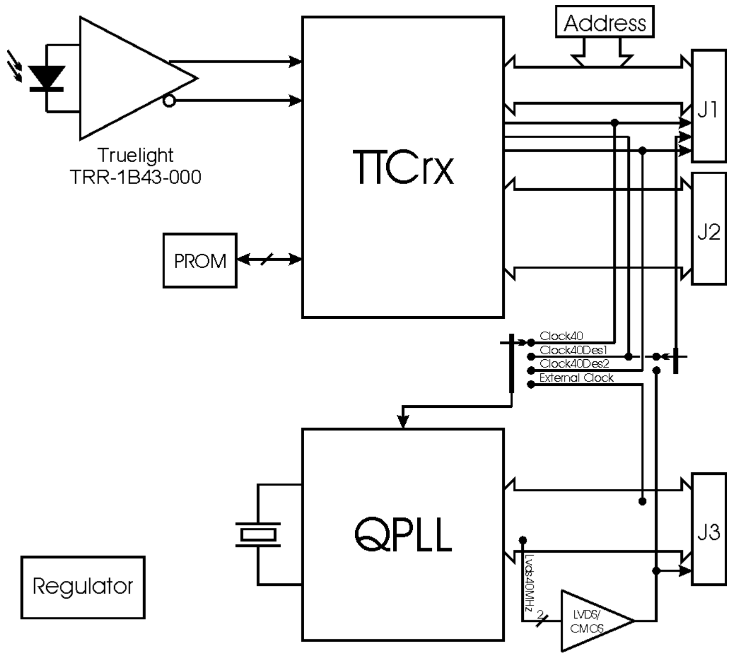


Figure 2 TTCrq block diagram

All the QPLL clock signals are available in the J3 connector as LVDS signals. Additionally, the 40 MHz clock output is also present as a CMOS output.

Users of the TTCrq should be aware that due to limited lock range of the QPLL (≈ ±160 ppm), high precision clock references centred around the LHC clock frequency are required to guaranty lock during laboratory test.

Power supply

Three independent power connections are present on the board: one dedicated to the pin-preamplifier, another to the QPLL and the third to the remaining circuitry.

The TTCrx, the PROM and the LVDS/CMOS level converter can be either powered from 3.3 or 5 V. The choice of this voltage will set the CMOS levels of all the TTCrx signals as well as that of the 40 MHz CMOS clock output in the J3 connector. Notice however, that the pin-preamplifier has to be powered from a 5V power supply.

The power supply for the QPLL is obtained from the TTCrx power using a 2.5V low dropout regulator. Alternatively, it is possible to provide the QPLL power through pin 39 of connector J2. In this case the internal regulator should not be present in the circuit and the resistor R59 (0 Ω) must be mounted.

Termination resistors

When the LVDS clock signals are to be carried out of the board termination resistors (100 Ω) must be provided external to the board. These resistors should be located at the end of the transmission lines that carry the signals. The transmission lines should be designed to have 100 Ω differential characteristic impedance.

Optionally all LVDS signals (input and outputs) can be terminated on the board. In principle, the LVDS signals that are not in use do not require a termination resistor. However, by default, the 40 MHz LVDS clock signal is terminated on the board for proper operation of the internal LVDS to CMOS level translator. If this signal is to be used outside the mezzanine card, the onboard termination should be removed and an external termination resistor must be provided.

TTCrq pin assignments:

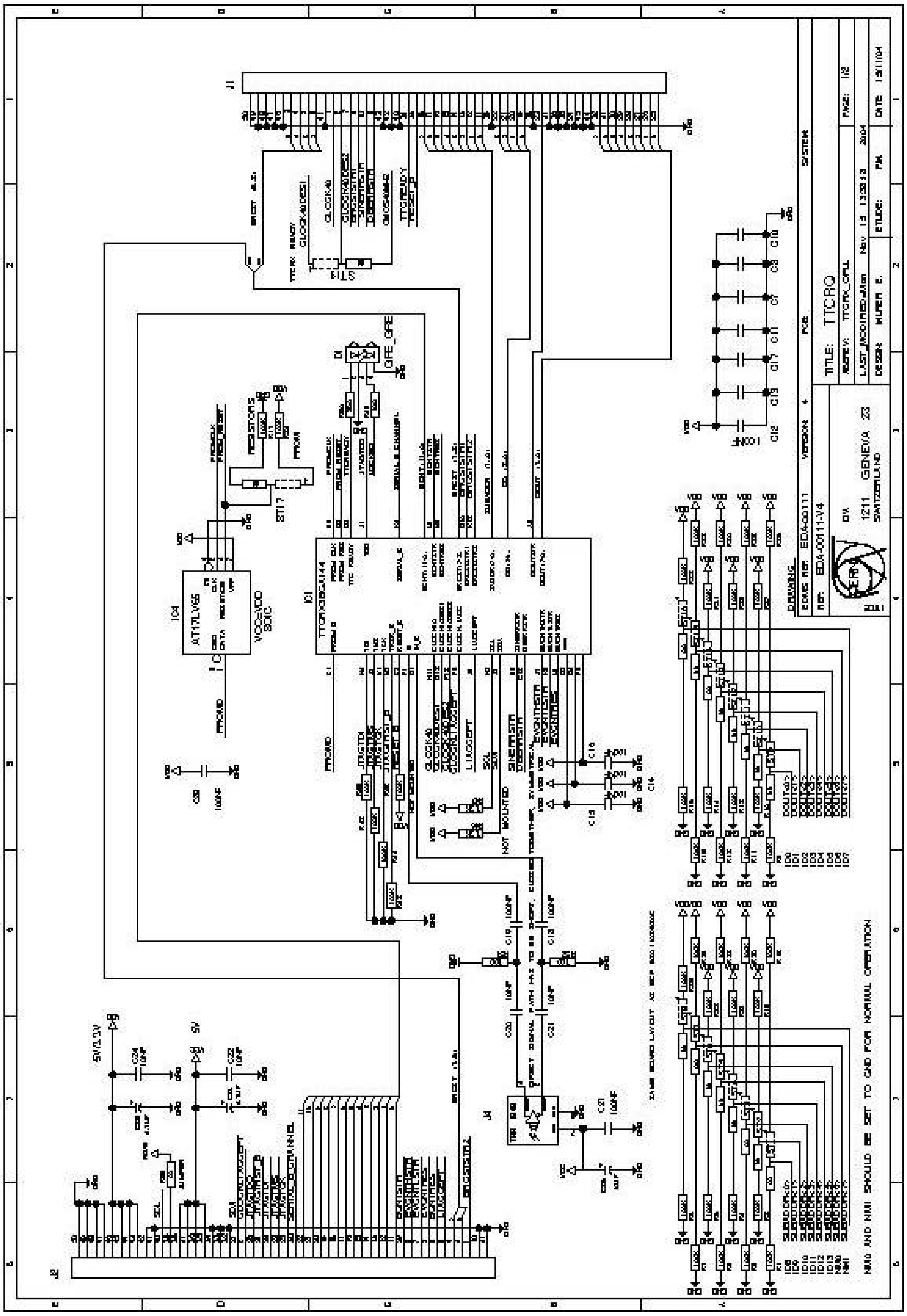
J1 Connector		J2 Connector		J3 Connector	
Pin Number	**Signal Name**	**Pin Number**	**Signal Name**	**Pin Number**	**Signal Name**
1	Clock40	1	BrcstStr2	1	f_0Select<0>
2	Clock40Des1	2	ClockL1Accept	2	mode
3	Brcst<5>	3	Brcst<6>	3	inLVDS+
4	Brcst<4>	4	Brcst<7>	4	inLVDS-
5	Brcst<3>	5	EvCntRes	5	gnd
6	Brcst<2>	6	L1Accept	6	externalClock
7	Clock40Des2	7	EvCntLStr	7	autoRestart
8	BrcstStr1	8	EvCntHStr	8	externalControl
9	DbErrStr	9	BcntRes	9	f_0Select<3>
10	SinErrStr	10	GND	10	~reset
11	SubAddr<0>	11	BCnt<0>	11	locked
12	SubAddr<1>	12	BCnt<1>	12	error
13	SubAddr<2>	13	BCnt<2>	13	gnd
14	SubAddr<3>	14	BCnt<3>	14	lvds80MHz-
15	SubAddr<4>	15	BCnt<4>	15	lvds80MHz+
16	SubAddr<5>	16	BCnt<5>	16	gnd

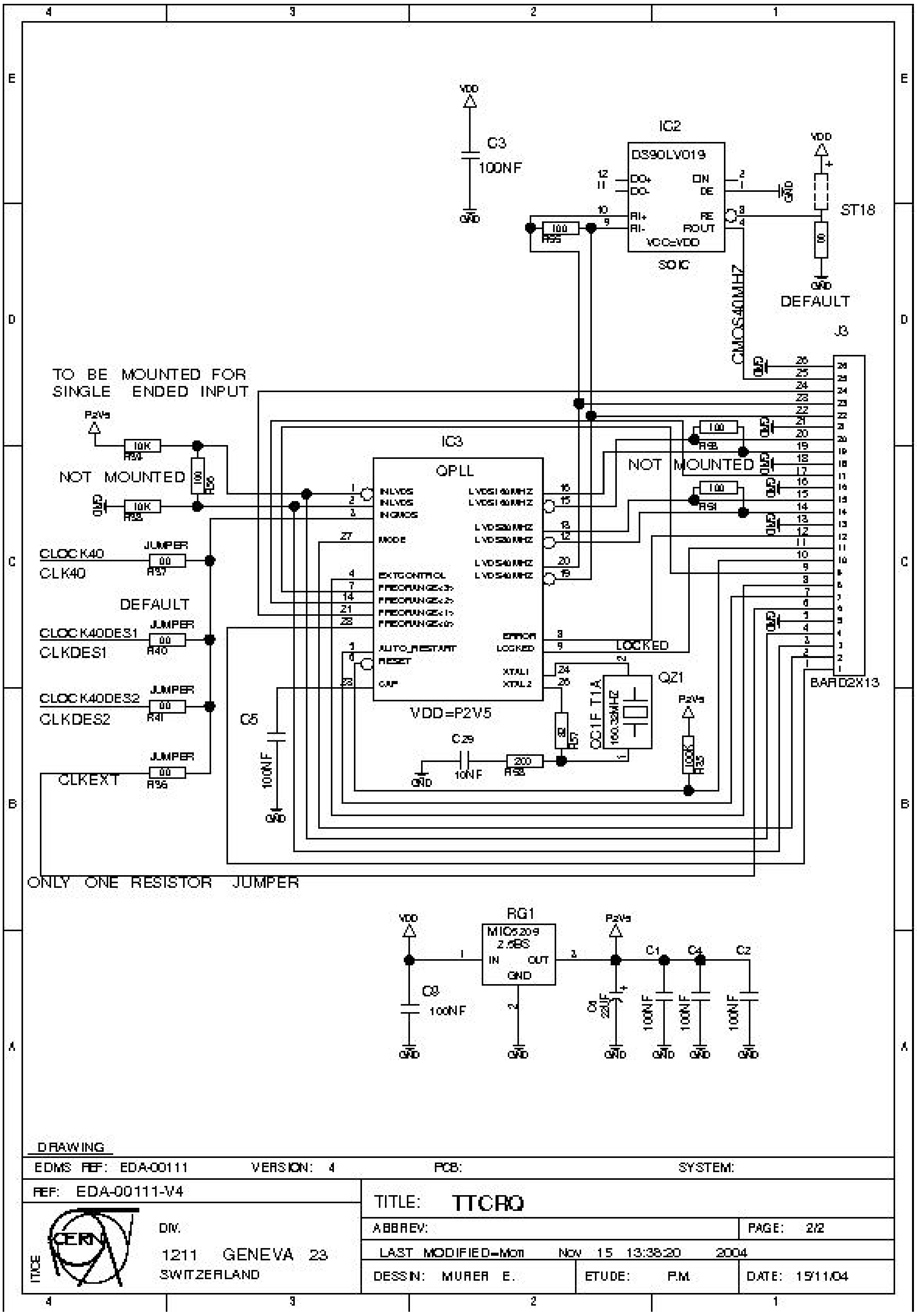
17	SubAddr<6>	17	BCnt<6>	17	f_0Select<2>
18	SubAddr<7>	18	BCnt<7>	18	gnd
19	DQ<0>	19	BCnt<8>	19	lvds160MHz+
20	DQ<1>	20	BCnt<9>	20	lvds160MHz-
21	DQ<2>	21	BCnt<10>	21	gnd
22	DQ<3>	22	BCnt<11>	22	lvds40MHz-
23	DoutStr	23	JTAGTMS	23	lvds40MHz+
24	GND	24	JTAGTRST_b	24	f_0Select<1>
25	Dout<0>	25	JTAGTCK	25	cmos40MHz
26	Dout<1>	26	JTAGTDO	26	gnd
27	Dout<2>	27	SDA		
28	Dout<3>	28	JTAGTDI		
29	Dout<4>	29	BCntStr		
30	Dout<5>	30	Serial_B_Channel		
31	Dout<6>	31	GND		
32	Dout<7>	32	GND		
33	Reset_b	33	GND		
34	TTCReady	34	GND		
35	GND	35	PIN_Preamp_VCC		
36	GND	36	PIN_Preamp_VCC		
37	GND	37	PIN_Preamp_VCC		
38	GND	38	PIN_Preamp_VCC		
39	GND	39	QPLL power (2.5 V)		
40	GND	40	SCL		
41	GND	41	GND		
42	GND	42	GND		
43	GND	43	TTCrx_VDD		
44	GND	44	TTCrx_VDD		
45	GND	45	TTCrx_VDD		
46	GND	46	TTCrx_VDD		
47	GND	47	GND		
48	GND	48	GND		
49	GND	49	GND		
50	GND	50	GND		

TTCrq schematics

Note: These schematics can also be obtained for the QPLL site:

http://www.cern.ch/proj-qpll





IC2
DS90LV019
SOIC
C3
100NF
VDD
GND
ST18
DEFAULT
CMOS40MHZ
J3
R55
TO BE MOUNTED FOR
SINGLE ENDED INPUT
P2V5
NOT MOUNTED
IC3
QPLL
INLVDS
INCMOS
MODE
EXTCONTROL
FREQRANGE<3>
FREQRANGE<2>
FREQRANGE<1>
FREQRANGE<0>
AUTO_RESTART
RESET
CAP
LVDS160MHZ
LVDS80MHZ
LVDS40MHZ
ERROR
LOCKED
XTAL1
XTAL2
VDD=P2V5
NOT MOUNTED
LOCKED
QZ1
QC1F T1A
160.32MHZ
BARD2X13
CLOCK40
CLK40
JUMPER
DEFAULT
CLOCK40DES1
CLKDES1
CLOCK40DES2
CLKDES2
CLKEXT
C5
100NF
C29
10NF
ONLY ONE RESISTOR JUMPER
RG1
MIC5209
2.5BS
IN
OUT
GND
C9
100NF
C1
C4
C2
22UF
100NF
DRAWING
EDMS REF: EDA-00111
VERSION: 4
PCB:
SYSTEM:
REF: EDA-00111-V4
TITLE: TTCRQ
DIV.
1211 GENEVA 23
SWITZERLAND
ABBREV:
PAGE: 2/2
LAST MODIFIED=Mon Nov 15 13:38:20 2004
DESSIN: MURER E.
ETUDE: P.M.
DATE: 15/11/04

TTCrq footprint

For layout purposes a footprint of the TTCrq was defined and is available for Allegro (Cadence) through the CERN components library. The library name is CNSPECIAL and the symbol name is TTCRQ_MEZZ.

TTCrq configuration

The TTCrq is setup by soldering SMD jumpers and resistors on the appropriate locations. For each jumper two positions are possible, these are indicated in the PCB silk layer by a “+” and a “-“ sign. The following tables describe the purpose of these jumpers and resistors and their default configurations:

ADRESS selection and Master modes

Jumper	+	-	Default	Function
ST1	PROM	Data bus	Not mounted	ID<8>
ST2	PROM	Data bus	Not mounted	ID<9>
ST3	PROM	Data bus	Not mounted	ID<10>
ST4	PROM	Data bus	Not mounted	ID<11>
ST5	PROM	Data bus	Not mounted	ID<12>
ST6	PROM	Data bus	Not mounted	ID<13>
ST7	PROM	Data bus	“-“	MN0 (No change allowed)
ST8	PROM	Data bus	“-“	MN1 (No change allowed)
ST9	PROM	Data bus	Not mounted	ID<0>
ST10	PROM	Data bus	Not mounted	ID<1>
ST11	PROM	Data bus	Not mounted	ID<2>
ST12	PROM	Data bus	Not mounted	ID<3>
ST13	PROM	Data bus	Not mounted	ID<4>
ST14	PROM	Data bus	Not mounted	ID<5>
ST15	PROM	Data bus	Not mounted	ID<6>
ST16	PROM	Data bus	Not mounted	ID<7>

PROM selection

Jumper	+	-	Default	Function
ST17	PROM	Data bus	“-“	TTCrx initialization method

J1 output clock selection

Jumper	+	-	Default	Function
ST19	TTCrx	QPLL	“-“	Clock source for clock pin 2 on J1

J3 CMOS clock output enable

Jumper	+	-	Default	Function
ST18	Disabled	Enabled	"-"	QPLL CMOS clock output enable

QPLL clock source selection

Resistor	Mounted	Value	Default	Function
R36	External	0 Ω	Not mounted	QPLL clock input[1,2,3]
R37	Clock40	0 Ω	Not mounted	QPLL clock input[1,2,3]
R38	Single ended clock input	10 kΩ	Mounted	Disables the LVDS clock input[2,3]
R39	Single ended clock input	10 kΩ	Mounted	Disables the LVDS clock input[2,3]
R40	Clock40Des1	0 Ω	Mounted	QPLL clock input[1,2,3]
R41	Clock40Des2	0 Ω	Not mounted	QPLL clock input[1,2,3]

Note 1: Only one of R36, R37, R40 and R41 can be mounted at a time.

Note2: To use the external LVDS clock input the resistors R36, R37, R38, R39, R40 and R41 must be all unmounted.

Note 3: to use the QPLL single ended input resistors R38 and R39 must be mounted.

J2 2.5V power

Resistor	Mounted	Value	Default	Function
R59	External Power	0 Ω	Not mounted	On board regulator provides the QPLL power[4]

Note 4: If the 2.5V power is provided from pin J2 – 39 the onboard regulator <u>must not</u> be assembled on the board.

I2C terminations

Resistor	Mounted	Value	Default	Function
R46	onboard termination	12 kΩ	Mounted	I2C SDA pull up
R47	onboard termination	12 kΩ	Mounted	I2C SCL pull up

LVDS terminations

Resistor	Mounted	Value	Default	Function
R53	onboard termination	100 Ω	Not mounted	LVDS 160 MHz clock output termination
R54	onboard termination	100 Ω	Not mounted	LVDS 80 MHz clock output termination
R55	onboard	100 Ω	Mounted	LVDS 40 MHz clock output

	termination			termination
R56	onboard termination	100 Ω	Not mounted	LVDS 40 MHz clock input termination

Timing

As illustrated in Figure 3, in the TTCrq, phase offsets exist between the reference clock fed to the QPLL and the QPLL clock signals. Since in the TTCrq the QPLL clock reference can be any of the TTCrx clocks (*Clock40*, *Clock40Des1* and *Clock40Des2*) in Figure 3 these signals are represented by the generic name of "*TTCrx Clock*". The timing of the signal "*Cmos40MHz*" depends not only on the QPLL but as well on the LVDS/CMOS translator. For further details on the timing of that device please see: http://cache.national.com/ds/DS/DS90LV019.pdf.

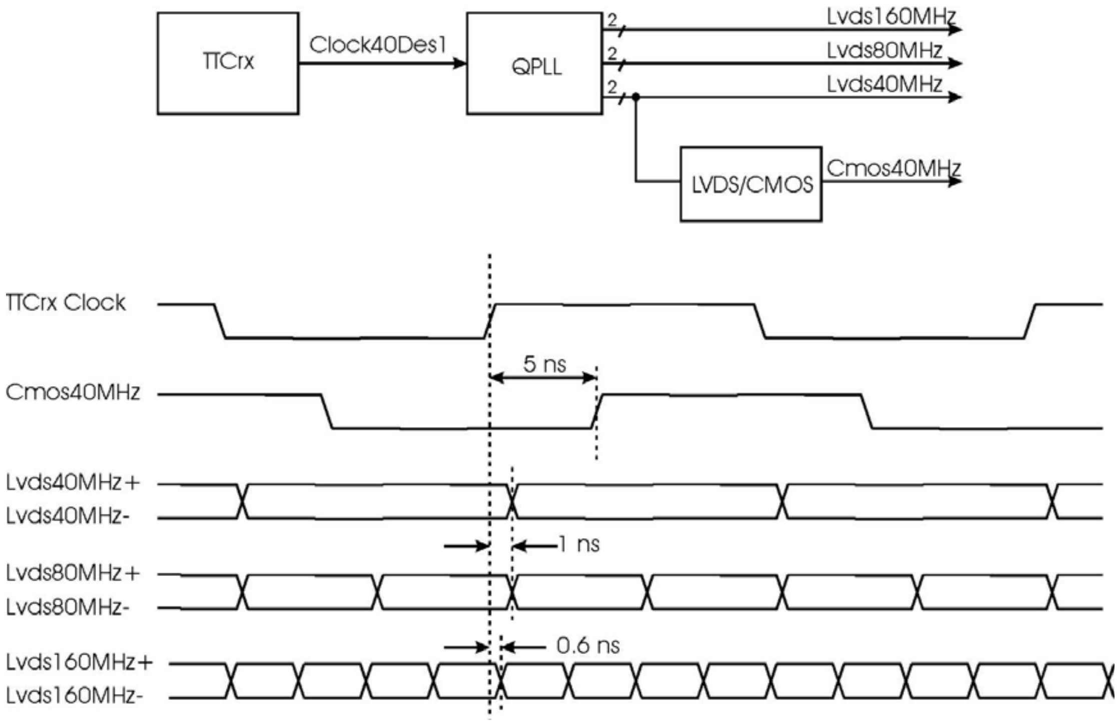


Figure 3 TTCrq timing

TTCrx and QPLL Mezzanine Card (TTCrq)

Paulo Moreira, Ernest Murer – 12/02/2003

Rev 0.1 – Preliminary

Introduction

A new mezzanine card is being designed as an alternative to the existing TTCrm supported by the CERN Electronics Pool (EP-ESS Group). The aim is twofold: to produce a card that can be mounted on a standard VME unit without imposing restrictions on the spacing between two modules and to add the QPLL functionality to the board[1]. Additionally the TrueLight pin-preamp (TRR-1B43-000) will replace the Agilent (HFBR – 2316).

The new TTCrx mezzanine card will be backward-compatible with the TTCrm. That means that the existing electrical connectors (J1 and J2) will be kept in the same physical positions with the same pinout. An additional connector (J3) will be added to the card located on the PCB side opposite to the optical connector side as represented in Figure 1. J3 will be a 26-pin connector. (VME board areas under the dotted/shadowed regions (top view drawing) should remain free for tool insertion during board removal.)

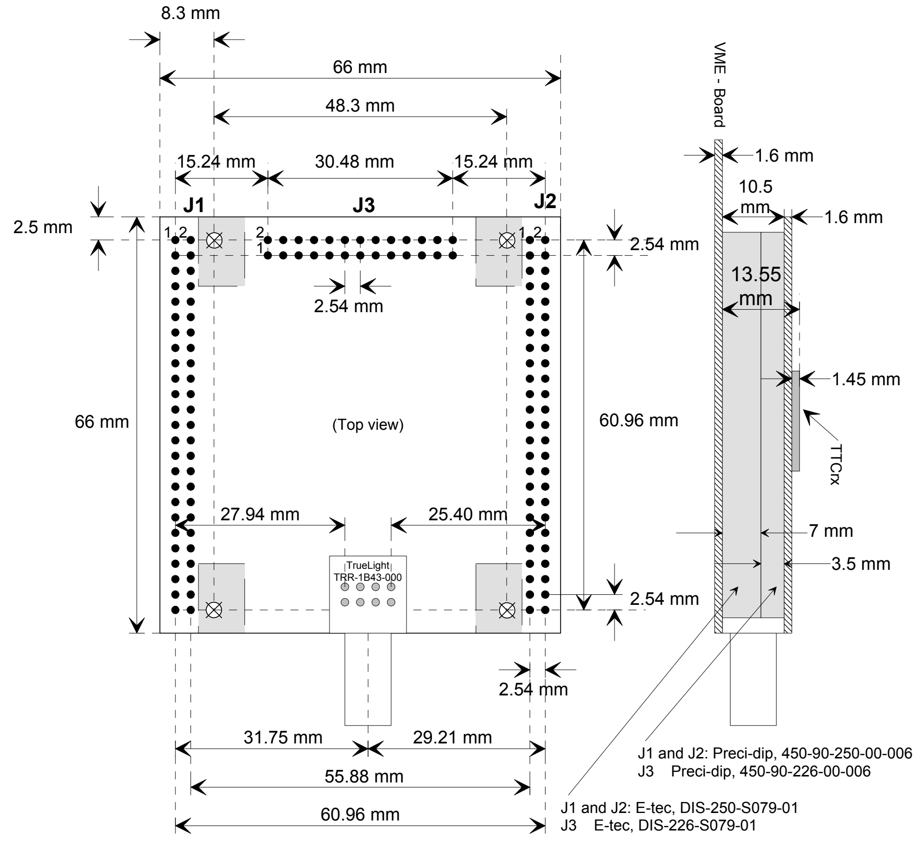


Figure 1 TTCrx and QPLL mezzanine card

To reduce the module height the optical receiver will be mounted under the card on the same PCB side as the electrical connectors. Therefore, the optical connector will moved to the other side of the PCB but its distance to the J1 and J2 connectors will remain unchanged (please see details in Figure 1).

As shown in Fig. 1, the overall height of the components on the mounted TTCrq mezzanine card is 13.55 mm above the components side of the VME motherboard. To ensure compliance with VMEbus Rule 7.14, assembled VME modules should be measured to verify that the sum of TTCrm component height and board warpage does not exceed 13.71 mm.

[1] Support by CERN Electronics Pool for the new mezzanine is under discussion.

The 13.71 mm limit allows a guaranteed 2.44 mm clearance between the TTCrx and the longest component leads on the adjacent VME board. More importantly for the cooling airflow, it allows a nominal 4.91 mm space to an unwarped adjacent VMEboard. According to VMEbus Observation 7.11, this space allows adequate airflow for cooling. However, designers should avoid putting high-dissipation components on the VME board underneath the mezzanine board, as the horizontal orientation of the connectors is likely to restrict cooling airflow to them.

Circuit

A block diagram of the mezzanine card is represented in Figure 2. As before the card contains a pin-preamplifier (the Truelight TRR-1B43-000), the TTCrx, the PROM and a bank of pull-up/pull-down resistors to setup the TTCrx address. The address jumpers were removed to satisfy the space constraints. Dipswitches will replace the jumpers if the available PCB space will allow for them. Otherwise, address setting will be done by soldering the appropriate pull-up / pull-down resistors. Connectors J1 and J2 remain in the same relative positions with the same pin assignments. Additionally, a QPLL and its associated crystal are added to the card. All the QPLL pins (with the exception of the crystal dedicated pins and VCXO decoupling capacitor pin) are accessible through the connector J3. The QPLL input can be taken either from the J3 connector (external source) or from one of the TTCrx clock outputs (Clock40, Clock40Des1 or Clock40Des2). When using the QPLL with an external source the reference clock signal can be either LVDS or CMOS.

All the QPLL clock signals are available in the J3 connector as LVDS signals. Additionally, the 40 MHz clock output is also present as a CMOS output.
As before, two independent power connections are present on the board: one dedicated to the pin-preamplifier and the other for the remaining circuitry. The TTCrx, the PROM and the LVDS/CMOS level converter can be either powered from 3.3 or 5 V. The choice of this voltage will set the CMOS levels of all the TTCrx signals as well as that of the 40 MHz CMOS clock output in the J3 connector. Notice however, that the pin-preamplifier has to be powered from a 5V power supply. The power supply for the QPLL is obtained from the TTCrx power using a 2.5V low dropout regulator.

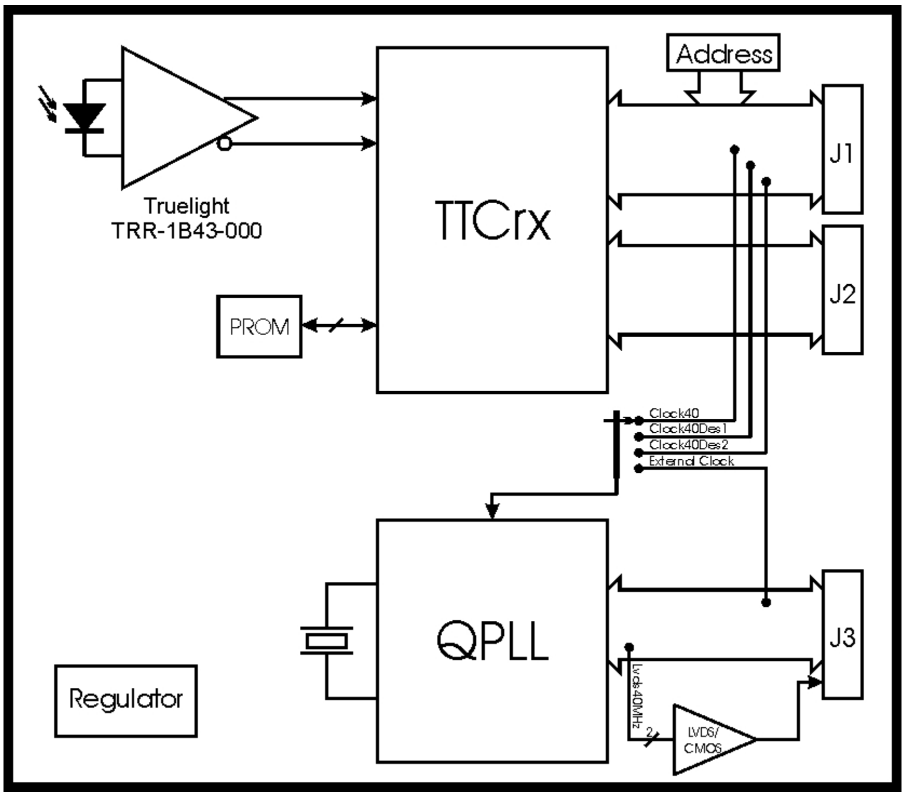


Figure 2 TTCrq block diagram

Users of the TTCrq should be aware that due to limited lock range expected for the QPLL (≈ ±50 ppm), high precision clock references centered around the LHC clock frequency will be required to guaranty lock during laboratory test.

J3 connector pin assignments[2]

Pin Number	Signal Name	Signal type
1	f_0Select<0>	Input, CMOS 5V compatible
2	mode	Input, CMOS 5V compatible
3	inLVDS+	Input, LVDS
4	inLVDS-	Input, LVDS
5	gnd	Power
6	externalClock	Input, CMOS 5V compatible
7	autoRestart	Input, CMOS 5V compatible
8	externalControl	Input, CMOS 5V compatible
9	f_0Select<3>	Input, CMOS 5V compatible
10	~reset	Input, CMOS 5V compatible
11	locked	Output, CMOS 2.5 V
12	error	Output, 2.5 V compatible
13	gnd	Power
14	lvds80MHz-	Output, LVDS
15	lvds80MHz+	Output, LVDS
16	gnd	Power
17	f_0Select<2>	Input, CMOS 5V compatible
18	gnd	Power
19	lvds160MHz+	Output, LVDS
20	lvds160MHz-	Output, LVDS
21	gnd	Power
22	lvds40MHz-	Output, LVDS
23	lvds40MHz+	Output, LVDS
24	f_0Select<1>	Input, CMOS 5V compatible
25	cmos40MHz	Output, CMOS
26	gnd	Power

[2] J1 and J2 pin assignments are left unchanged. Please see the TTCrx reference manual in the TTC system web site (http://ttc.web.cern.ch/TTC/intro.html).

QPLL Manual

Quartz Crystal Based Phase-Locked Loop for Jitter Filtering Application in LHC

Paulo Moreira

CERN - EP/MIC, Geneva Switzerland

2005-01-10

Version 1.1

Technical inquires: Paulo.Moreira@cern.ch

Preliminary

Summary of Changes

Version 1.1:

This version of the manual applies to both QPLL2 and QPLL3. Both versions of the ASIC are functionally identical. However QPLL3 has a higher tolerance to ionizing radiation (total dose). It is thus recommended to restrict the use of the QPLL2 to systems were radiation tolerance is not a concern.

Manual changes:

- Crystal specifications changed. See “Crystal specification”;
- Addition of the section “Power Reduction Network”;
- Section: “PCB Layout recommendations”: expanded.

Version 1.0:

This version of the manual reflects the changes that were introduced in the second version of the QPLL. To avoid any confusion with the previous version, these chips are now marked as “**QPLL2**”.

QPLL version 2 is 100% pin compatible with version 1. Except for operation mode 2 (see QPLL operation modes) the two versions are functionally identical. Users that already developed boards based on version 1 will be able to simply replace each QPLL by a QPLL2.

ASIC changes:

- The frequency select bus was expanded to 6 bits;
- Pins **autoRestart** and **~reset** become dual function;

Manual changes:

- Section “OPERATION”: expanded;
- Section “Timing”: new;
- Section “Crystal specification”: new.
- “Power supply sensitivity”: new;
- Section: “PCB Layout recommendations”: expanded;
- Section “Procedure to verify the PCB parasitic capacitance” new.

Version 0.3:

- Dielectric thickness corrected in Figure 10 (recommend layout).

Version 0.2:

- Legend corrected in Figure 10 (recommend layout).

Version 0.1:

- Section “PCB Layout recommendations” added to the manual.

INTRODUCTION

The QPLL is a Quartz crystal based Phase-Locked Loop. Its function is to act as a jitter-filter for clock signals operating synchronously with the LHC bunch-crossing clock. Two frequency multiplication modes are implemented: 120 MHz and 160 MHz modes[1]. In the 160 MHz mode, the ASIC generates three clock signals synchronous with the reference clock at 40 MHz, 80 MHz and 160 MHz while in the 120 MHz mode the synthesized frequencies are 40 MHz, 60 MHz and 120 MHz. In both cases, the highest frequency is generated directly from a Voltage Controlled Crystal Oscillator (VCXO) and the lower frequencies are obtained by synchronous division. The two frequency multiplication modes require Quartz crystals cut to the appropriate frequencies.

Features:

- Phase-Locked Loop based on a Voltage Controlled Crystal Oscillator
- Designed to frequency and phase-lock to the LHC master clock: f = 40.0786 MHz
- Locking range: $\Delta \approx \pm 3.7$ KHz arround f = 40.0786 MHz
- Loop bandwidth: < 7 KHz
- Locking time – including a frequency calibration cycle (mode 1): ~180 ms
- Locking time – excluding a frequency calibration cycle (mode 0): ~250 μs
- Two frequency multiplication modes:
  - ×1, ×2 and ×4
  - ×1, ×1.5 and ×3
- Output jitter: < 50 ps peak-to-peak for an input signal jitter less than 120 ps RMS
- Reference clock input levels:
  - LVDS
  - CMOS single-ended, 2.5 V to 5 V compatible
- Three LVDS clock outputs
- Package: LPCC-28 (5 mm × 5 mm, 0.5 mm pitch)
- Power supply voltage: 2.5V nominal (allowed operation range 2.4V to 2.7V)
- Phase error sensitivity to the power supply voltage: less than -0.72 ps/mV
- VCXO free-running frequency sensitivity to the power supply: 0.14 Hz/mV (typical)
- Power consumption: 100 mW
- Radiation tolerant
- 0.25 μm CMOS technology
- Crystal: A quartz crystal is provided with each QPLL.

[1] Please note that frequency numbers in this document are often rounded to the nearest integer. This is just a simplification to facilitate document reading. In fact, these numbers should be interpreted to be the exact multiples of the LHC bunch-crossing clock frequency.

OPERATION

The QPLL uses the LHC bunch-crossing clock as the reference frequency. This signal can be feed to the ASIC either in CMOS or LVDS levels (please refer to Figure 1). Selection of which input to use is simply done by forcing the unused clock input to logic level "0" (notice the use of the OR function in the reference clock signal path in the block diagram). The three clock outputs are LVDS signals and their frequency depends on the "*mode*" input. When "*mode*" is set to "0" the output clock frequencies are: 40 MHz, 60 MHz and 120 MHz otherwise the frequencies are: 40 MHz, 80 MHz and 160 MHz. Since the highest clock frequency is obtained directly from the Voltage Controlled Crystal Oscillator (VCXO), different crystals are required for operation in one of the two frequency multiplication modes. A crystal is provided by CERN with each QPLL for operation in the 160MHz mode.

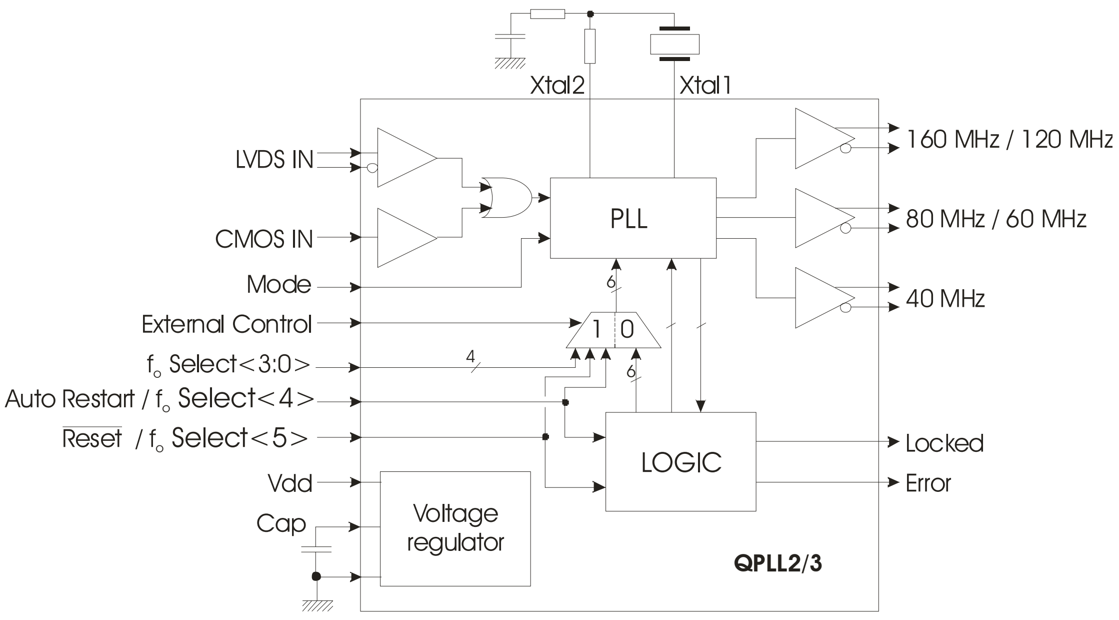


Figure 1 QPLL2 block diagram

The use of a VCXO in the QPLL allows to achieve low jitter figures but imposes the limitation of a small frequency lock range. To cope with crystal cutting accuracy, process, temperature and power supply variations, upon reset or loss of lock, the ASIC goes through a frequency calibration procedure. In principle, this is an automatic procedure that in most applications should be "transparent" to the user. However in some situations, like for example chip or system testing, the user might want to control it. The signals that are relevant to this function are: "*externalControl*", "*autoRestart*" and "*f_oSelect<5:0>*". If the "externalControl" signal is set to "1" then the automatic calibration procedure is disabled and the VCXO centre frequency is set by the signals "*f_oSelect<5:0>*" otherwise, the free running VCXO frequency is automatically determined. Please note that when the "*externalControl*" signal is set to "1" the signals "*autoRestart*" and "*~reset*" become *f_oSelect<4> and f_oSelect<5>* respectively.

The QPLL contains a lock detection circuit that monitors the lock state of the phase-locked loop. If the PLL is detected to be unlocked, a frequency calibration cycle is initiated to lock the PLL. This feature can be disabled by forcing the signal "*autoRestart*" to "0". In this case, a frequency calibration cycle is only started if a reset is applied to the IC. When "*externalControl*" is forced to "0" the "*locked*" signal reports the locked status of the PLL. In this case, the lock detection logic filters the random behaviour of the (internal) PLL lock indication. However, if the "*externalControl*" signal is set to "1" the "Locked" signal will have a random behaviour during loss-of-lock and lock-acquisition.

The logic circuits controlling the PLL use redundant logic techniques to cope with Single Event Upsets (SEU). The “error” flag indicates (momentarily) that one SEU has occurred. These errors are dealt with automatically requiring no action from the user.

QPLL operation modes

The QPLL operation modes are controlled by the state of the signals “externalControl” and “autoRestart” as indicated on Table 1.

externalControl	**autoRestart**	**Mode**
0	0	0
0	1	1
1	x	2

Table 1 QPLL operation mode selection

Mode 0:

In this mode the QPLL frequency calibration logic is active but a frequency calibration cycle is only executed after a reset.

<u>Mode advantages:</u> Once a first frequency calibration cycle has been executed (with the reference clock present) the QPLL will keep the frequency calibration settings until another reset is applied. This allows the QPLL to acquire lock relatively fast (~250 μs) when compared with “*mode 1*” where a frequency calibration is executed every time lock is lost (~180 ms). This mode can be particularly useful in radiation environments where both the reference clock and the QPLL analogue circuits can be subject to single event upsets.

<u>Mode disadvantages:</u> Because the frequency calibration settings are maintained during operation, only the QPLL analogue range is available to cope with the reference clock drifts and changes in the power supply voltage and temperature (please see Figure 2 for clarification of the terms used). In mode 0, the system where the PLL is integrated must guaranty that the QPLL lock signal is constantly monitored. In the case of loss of lock and if the QPLL does not regain lock after a pre-established delay a reset must be applied so that a new calibration cycle is executed.

Mode 1:

In this mode the QPLL frequency calibration logic is active, a frequency calibration cycle is executed after a reset or each time lock is lost.

<u>Mode advantages:</u> This mode requires minimum monitoring from the system in where the QPLL is integrated. The QPLL constantly monitors its lock state and executes a frequency calibration cycle every time loss-of-lock is detected. This mode displays the largest tracking range in relation to the reference frequency drifts and the largest tolerance to power supply and temperature variations.

<u>Mode disadvantages:</u> In principle, this should be the preferred mode of operation. However in radiation environments this mode can lead to relatively large “dead times” (~180 ms) since a calibration cycle will be executed each time the reference clock or the analogue circuitry of the QPLL will be disturbed by a single event upset. In those circumstances “*mode 0*” might be preferable.

Mode 2:

In this mode the QPLL frequency calibration logic is inactive. The QPLL can be used as a PLL or as a standalone clock generator:

Operation as a PLL:

The user must centre the VCXO operation range around the reference clock frequency by setting the bits f_0Select<5:0>. As shown in Figure 2 the settings should be such that the centre of the analogue range is as close as possible to the operation frequency.

Mode advantages: None. Mode mainly used for chip characterization and production testing.

Mode disadvantages: Requires the user to constantly keep track of any changes of circuit characteristics (for example crystal aging) and operation conditions like power supply voltage and temperature.

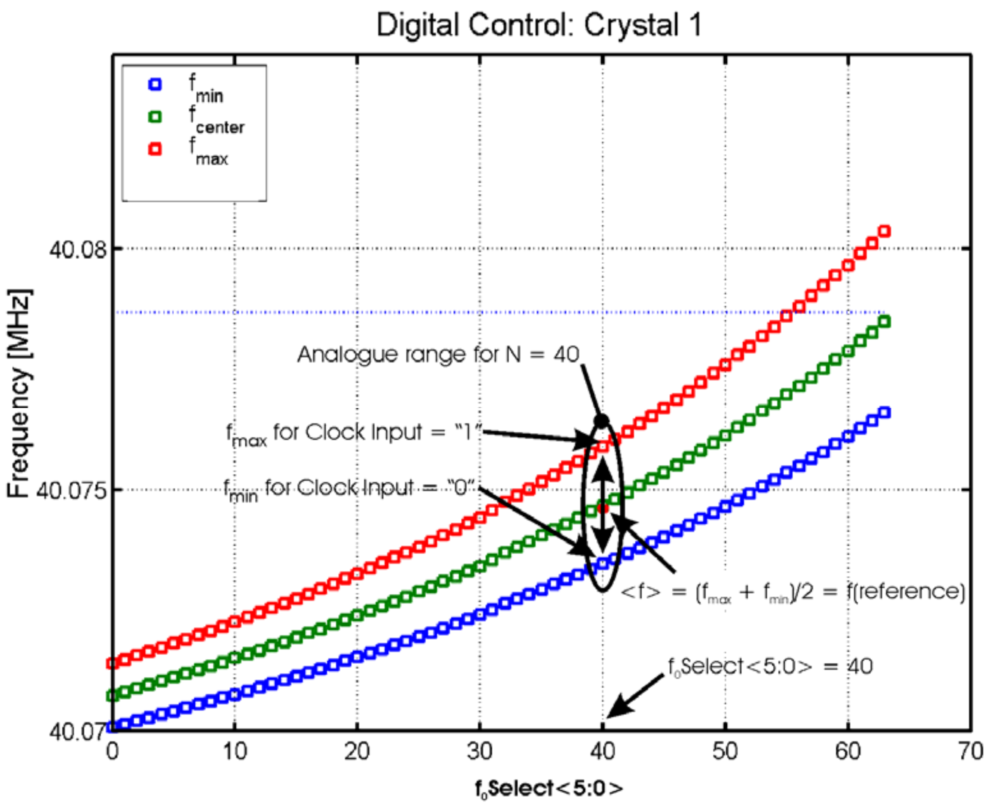


Figure 2. VCXO frequency as function of the digital control bits (f_0Select<5:0>)[1].

Operation as a Clock Source

The QPLL can be used as a standalone crystal oscillator whose frequency can be tuned by the control bits f_0Select<5:0>. In this case the QPLL is simply operated without a reference clock.
In this mode, the clock input can be used as an "extra frequency select bit". That is, after a given range is selected by the bits f_0Select<5:0> (see Figure 2), the clock input can be used to choose between the maximum and minimum oscillation frequencies of

[1] This picture will be updated once the final quartz crystals will be available. It is used here only as an example. The frequency range is not the target range.

that range. Setting the clock input to a “1” selects the maximum frequency while setting it to “0” selects the minimum frequency.

Warning: Mode 1 should never be used for standalone operation (QPLL as a simple clock generator). In that mode and in the absence of a reference clock, the QPLL is constantly executing frequency calibration cycles and its clock outputs are constantly having frequency “jumps”. Any QPLL trying to lock to such a signal will never achieve a stable lock. Mode 2 is thus the only mode recommended to implement a standalone clock source using a QPLL.

QPLL Signals

autoRestart – 5V compatible CMOS input with internal pull-up resistor. The functionality of this signal depends on the state of the "*externalControl*" signal:

if externalControl = "0"

autoRestart = "0": Automatic restart of the PLL is disabled. A frequency calibration cycle will only occur after a reset.

autoRestart = "1": Automatic restart is enabled. A frequency calibration cycle will occur each time the PLL is detected to be unlocked or after a reset.

if externalControl = "1"

"autoRestart" becomes "f_o*Select<4>"*.

cap – VCXO decoupling node:

A 100 nF capacitor must be connected between this pin and ground. Inductance of the interconnection must be minimized.

error – 2.5V CMOS output:

This signal indicates that an SEU has occurred. Since SEU events are dealt with automatically by the ASIC logic, this signal will be active only during the period in which the error condition will persist. A SEU on the QPLL logic circuits will not affect the operation of the PLL

externalControl – 5V compatible CMOS input with internal pull-down resistor:

externalControl = "0": The VCXO centre frequency is set by the automatic frequency calibration procedure.

externalControl = "1": The VCXO free running frequency is set by the input signals f_o*Select<5:0>.*

f_oSelect<3:0> - 5V compatible CMOS inputs with internal pull-down/pull-up resistors (f_o*Select<3>* ← pull-up, f_o*Select<2>* ← pull-down, f_o*Select<1>* ← pull-down, f_o*Select<0>* ← pull-down):

These signals (including f_o*Select<5:4>)* control the VCXO free running oscillation frequency when the signal "*externalControl*" is set to "1". If "*externalControl*" is set to "0" these signals have no influence on the IC operation.

inCMOS – 5V compatible CMOS clock input with internal pull-down resistor:

This is the CMOS reference clock input. **When in use, "*inLVDS+*" and "*inLVDS*"- must be set to logic levels "0" and "1" respectively**.

inLVDS+ and inLVDS- – LVDS clock inputs:

These signals are the LVDS reference clock inputs. **When in use, "*inCMOS*" must be held at logic level "0"**.

locked – 2.5V CMOS output:

This signal reports the PLL locked status.

if externalControl = "0"

In this case the lock indication is filtered by the QPLL lock detection logic, giving a stable indication during loss-of-lock, lock-acquisition or during lock.

if externalControl = "1"

In this case the lock indication state reflects the instantaneous lock indication provided by the PLL. The signal will display a random behaviour during loss-of-lock or lock-acquisition.

Lvds40MHz+ lvds40MHz- – LVDS output:

40MHz clock output

lvds80MHz+ lvds80MHz- – LVDS output:

mode = *“0”*: 60 MHz clock signal (with 120 MHz quartz crystal).

mode = “1”: 80 MHz clock signal (with 160 MHz quartz crystal).

lvds160MHz+ lvds160MHz- – LVDS output.

mode = *“0”*: 120 MHz clock signal (with 120 MHz quartz crystal).

mode = *“1”*: 160 MHz clock (with 160 MHz quartz crystal).

mode – 5V compatible CMOS input with internal pull-up resistor:

mode = “0”: 120 MHz frequency multiplication mode (120 MHz quartz crystal required).

mode = “1”: 160 MHz frequency multiplication mode (160 MHz quartz crystal required).

~reset – 5V compatible CMOS input:

if externalControl = “0”

Active low reset signal. It initiates a frequency calibration cycle and lock acquisition.

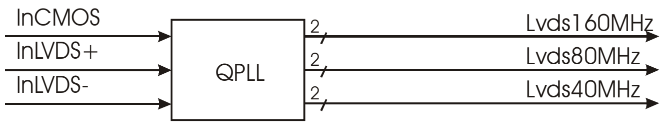
if externalControl = “1”

“~reset” becomes “*f_oSelect<5>*”.

xtal1, xtal2 – Quartz crystal connections pins

Timing

The QPLL timing diagram is illustrated in Figure 1. The values for the several clock outputs are given in Table 2 and Table 3.



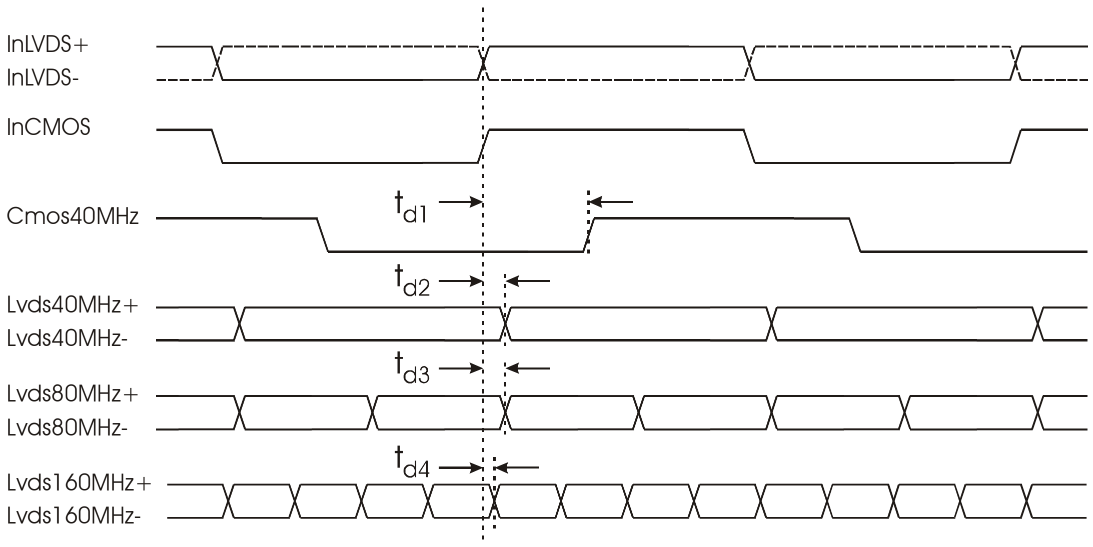


Figure 3 QPLL timing definitions

CMOS clock reference input

	Min [ns]	Typical [ns]	Max [ns]
t_{d1}	1.2	1.6	2.5
t_{d2}	1.3	1.7	2.6
t_{d3}	1.3	1.7	2.6
t_{d4}	0.8	1.2	2.1

Table 2 Output delays referenced to the CMOS clock input

LVS clock reference input

	Min [ns]	Typical [ns]	Max [ns]
t_{d1}	1.1	1.7	3.0
t_{d2}	1.2	1.8	3.1
t_{d3}	1.2	1.8	3.1
t_{d4}	0.7	1.3	2.6

Table 3 Output delays referenced to the LVDS clock input

QPLL PINOUT

The QPLL is packaged in a 28-pin 5 mm × 5 mm Leadless Plastic Chip Carrier (LPCC-28) with 0.5 mm pin pitch. Additional package information can be obtained from the "ASAT" web site (http://www.asat.com).

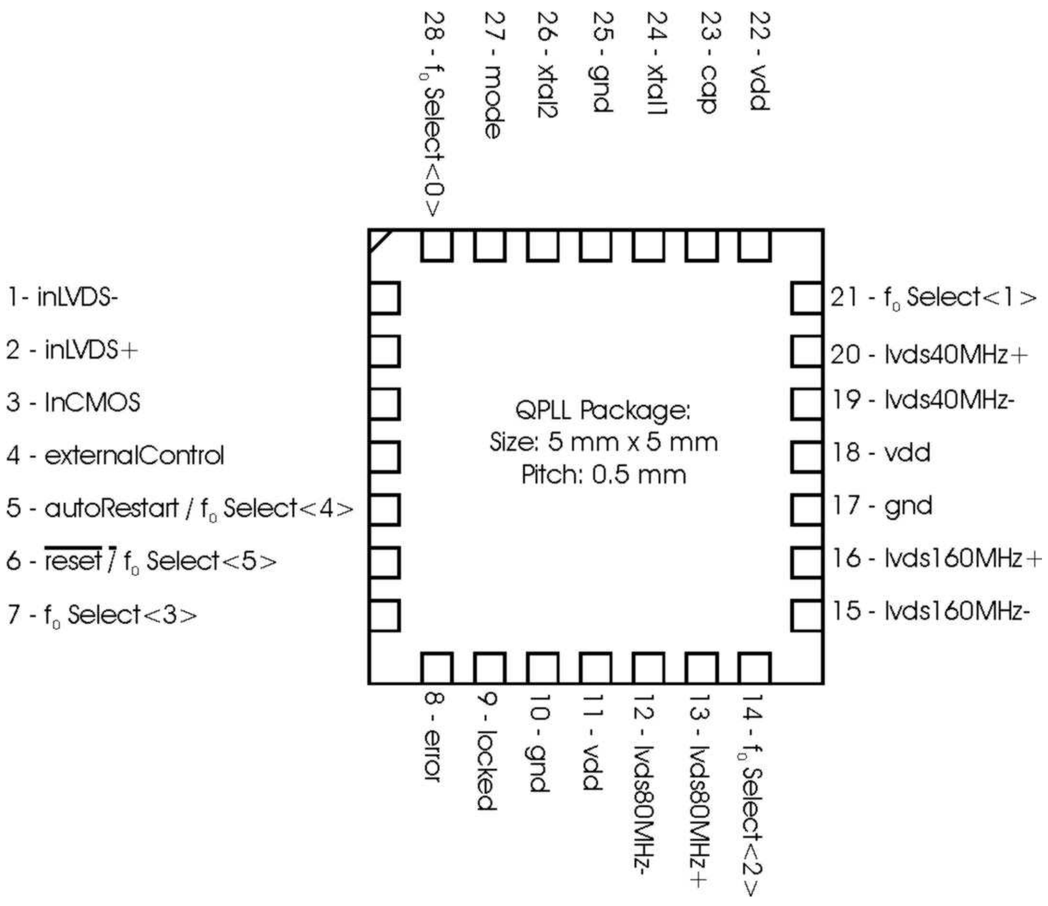


Figure 4 QPLL2 pinout

Pin assignments

Pin Number	Signal Name	Signal type
1	inLVDS-	Input, LVDS
2	inLVDS+	Input, LVDS
3	inCMOS	Input, CMOS 5V compatible
4	externalControl	Input, CMOS 5V compatible
5	autoRestart / f_0Select<4>	Input, CMOS 5V compatible
6	~reset / f_0Select<5>	Input, CMOS 5V compatible
7	f_0Select<3>	Input, CMOS 5V compatible
8	error	Output, 2.5 V compatible
9	locked	Output, CMOS 2.5 V

10	gnd	Power
11	vdd	Power
12	lvds80MHz-	Output, LVDS
13	lvds80MHz+	Output, LVDS
14	f_0Select<2>	Input, CMOS 5V compatible
15	lvds160MHz-	Output, LVDS
16	lvds160MHz+	Output, LVDS
17	gnd	Power
18	vdd	Power
19	lvds40MHz-	Output, LVDS
20	lvds40MHz+	Output, LVDS
21	f_0Select<1>	Input, CMOS 5V compatible
22	vdd	Power
23	cap	Power
24	xtal1	Analogue, Quartz crystal
25	gnd	power
26	xtal2	Analogue, Quartz crystal
27	mode	Input, CMOS 5V compatible
28	f_0Select<0>	Input, CMOS 5V compatible

CRYSTAL SPECIFICATION

A quartz crystal will be provided with each QPLL. The main characteristics of the crystal are given on the following table.

Pos	Description	Symbol	Typ.	Min.	Max.	Unit
1	Crystal type	Inverted mesa AT-Cut				
2	Resonance mode	Fundamental				
3	Load Frequency[3]	**FL**	160.314744			MHz
4	Load Capacitance	**CL**	5.5			pF
5	Frequency Tolerance at 25°C	**ΔFL/FL**		-18	18	ppm
6	Motional Capacitance	**C1**		4.2		fF
7	Static Capacitance	**Co**	2.8			pF
8	Drive Level	**P**			500	µW
9	Operating Temperature Range	**OTR**		0	60	°C
10	Series Resistance at 25°C	**Rs**			25	Ohm
11	Drift over OTR	**ΔFL/FL**		-10	10	ppm
12	Aging first year	**ΔFL/FL**			± 3	ppm
13	Package type	SMD ceramic				
14	Package surface				29.6	mm^2
15	Package height				1.75	Mm

Table 4 Quartz crystal specification

[3] This spec needs a 100% frequency verification over temperature range (5 °C intervals)

POWER REDUCTION NETWORK

QPLL excess jitter

It was observed that, depending on the operating temperature, the QPLL was generating at some frequencies within the locking range amounts of jitter well above the specification. The problem was investigated by CERN, NEVIS and Micro Crystal (the quartz crystal manufacturer) and it was established that the large jitter behaviour could be explained by activity dips in the crystal. Not all the crystals manufactured reveal the presence of activity dips but it turns out that they appear due to overdrive. The problem is solved by reducing the power delivered to the crystal by inserting an RC network between the QPLL and the crystal as discussed below.

Recommended network:

The network that should be used to reduce the power delivered to the crystal by the QPLL is represented in Figure 5. It is composed of two resistors (R1 and R2) and a capacitor (C). It should be inserted between the crystal and the QPLL as indicated. Notice that although the crystal is a symmetrical device the network must be connected to pin XTAL2.

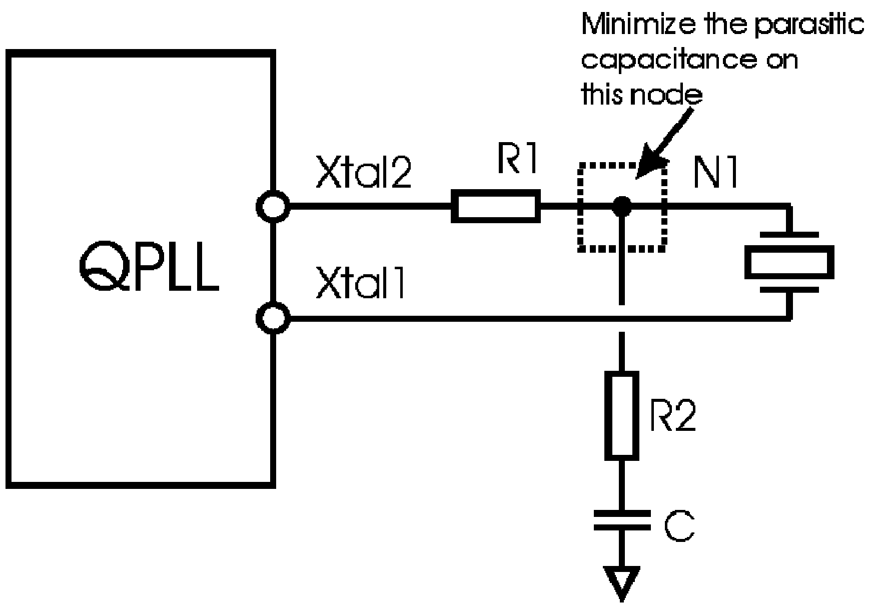


Figure 5 Power reduction network

Values for the circuit components are given on Table 5. If this values are used the QPLL performance remains basically unchanged in what concerns centre frequency, jitter performance and locking range. However, as explained in section "PCB Layout recommendations", to guaranty this some careful layout is mandatory.

Table 5 Component values for the power reduction network

R1 [Ω]	**R2 [Ω]**	**C [nF]**
62	240	10

It is not possible to predict which crystals will display activity dips so it is absolutely recommended the use of this network. Values in Table 5 must be respected.

POWER SUPPLY SENSITIVITY

Some of the QPLL characteristics are power supply voltage dependent: namely the VCXO free-running oscillation frequency and the static phase error. Both of these variables have an influence on the jitter performance. It is thus advisable to keep the noise levels on the power supply to the minimum possible. The numbers given below will help the user to form an opinion on how much noise can be tolerated on the power supply without incurring performance degradation.

Static phase error

The PLL, inside the QPLL ASIC, is a control loop that tries to maintain zero phase error between the reference clock and the internally generated VCXO clock. However, both these clocks propagate through clock buffers that introduce a non-zero delay between the two signals (see Timing). More over, since these buffers are external to the PLL control loop their power supply dependence is not compensated for. Variations in the power supply will result thus in a varying phase delay between the two clock signals.

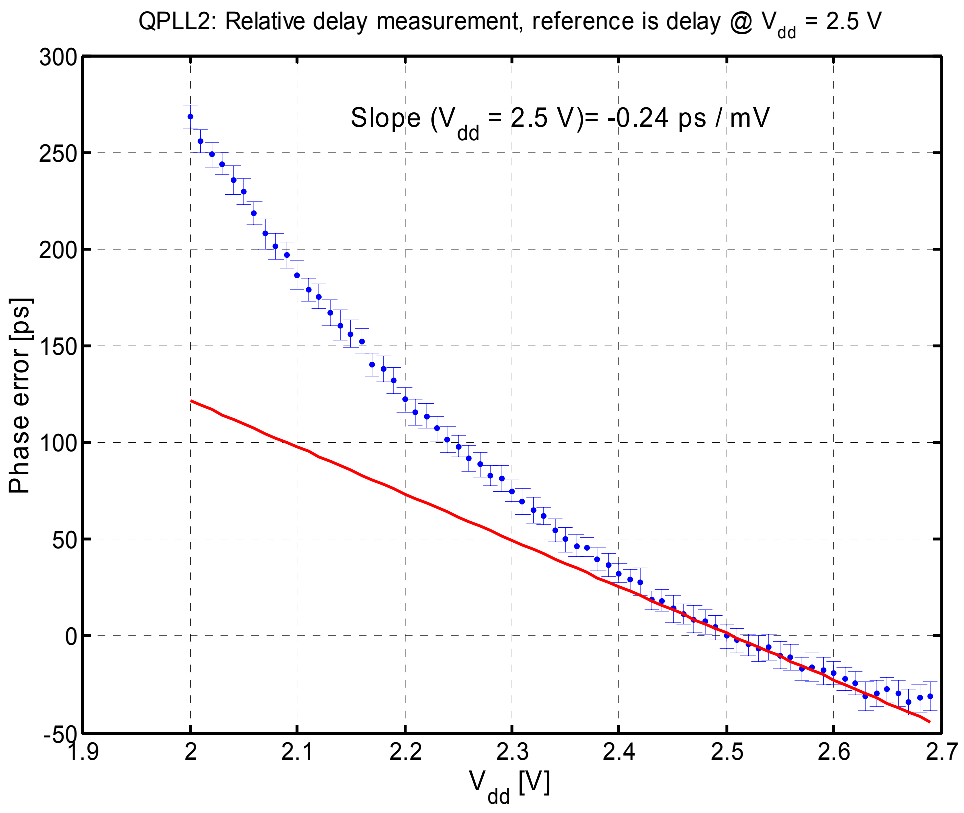


Figure 6 Phase error as function of the power supply voltage (relative measurement). In this measurement the reference clock is fed to the CMOS input

Figure 6 displays a typical curve for the static phase error as a function of the power supply voltage. In this case, the CMOS clock input is used as the clock reference input. The measurement is relative, that is, the phase error introduced by varying the power supply is measured relative to the static phase error when the power supply voltage is 2.5 V (the nominal power supply voltage). The curve displays a slope of -0.24 ps/mV at 2.5 V. Similarly, Figure 7 displays the static phase error when the LVDS input is used. In this case the slope of the curve is -0.72 ps/mV for the nominal power supply voltage.

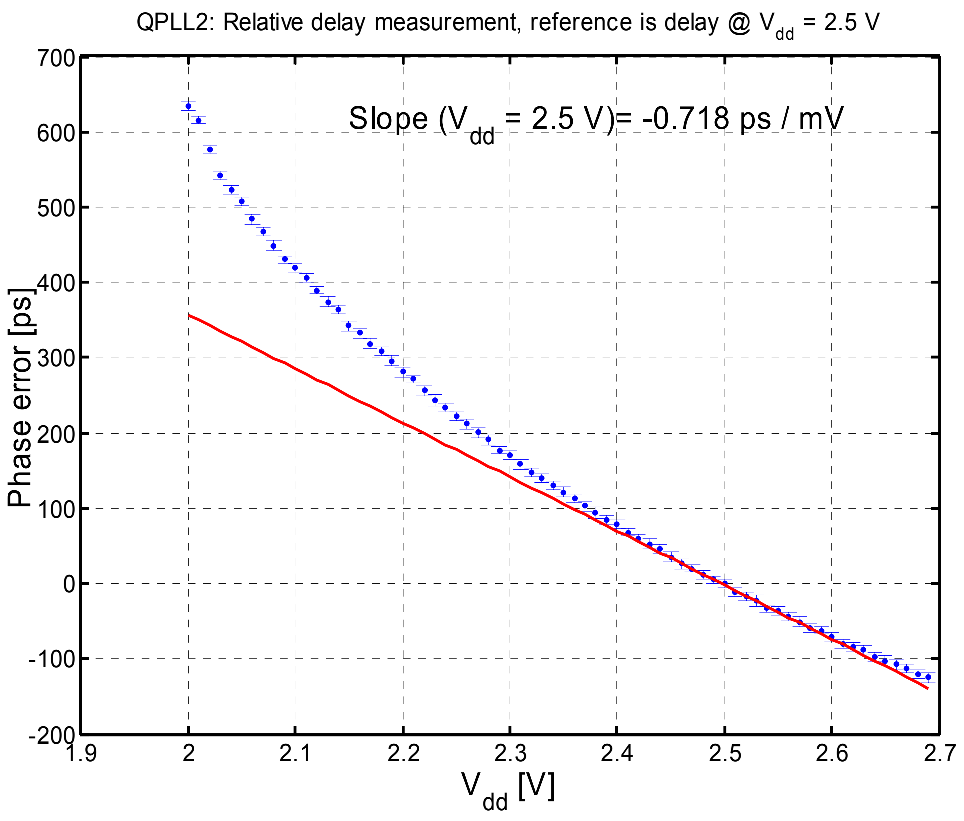


Figure 7 Phase error as function of the power supply voltage (relative measurement). In this measurement the reference clock is fed to the LVDS input.

VCXO free-running oscillation frequency

The QPLL can run stand alone as a clock source (see Operation as a Clock Source). When running in this mode there is no external reference to be tracked and the VCXO will produce a frequency which is essentially dependent on the quartz crystal being used and on the ASIC settings chosen. However, since no reference signal is being tracked, the power supply voltage will have some influence on the oscillator frequency.

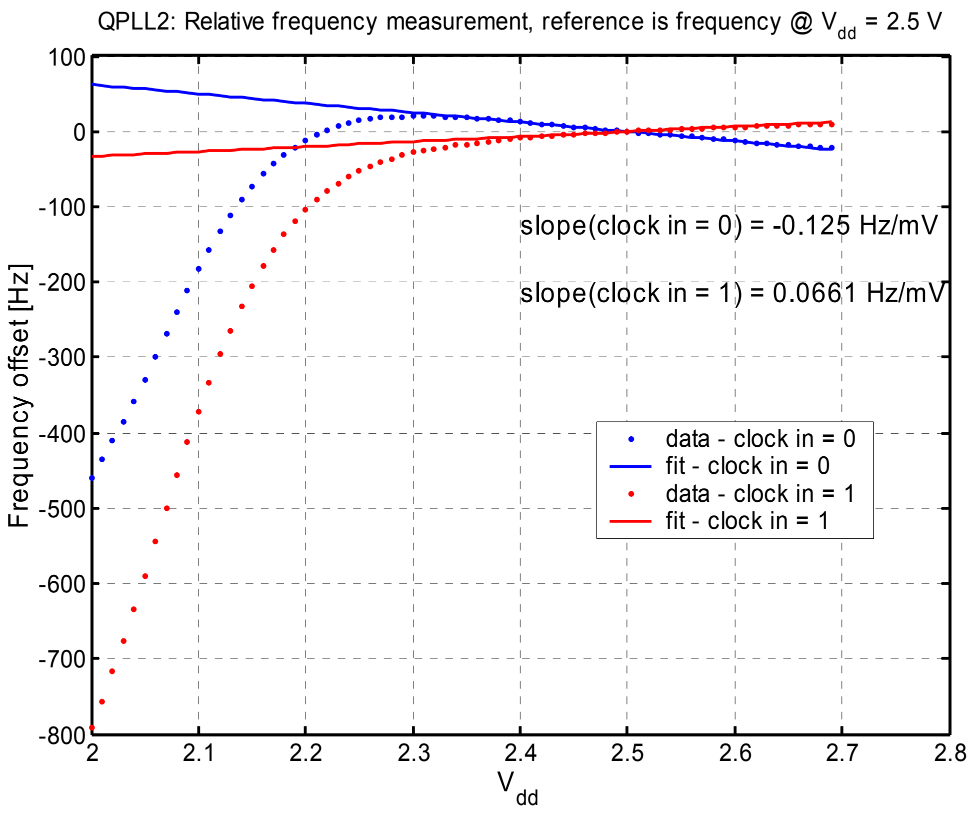


Figure 8 VCXO Oscillation frequency as function of the power supply voltage (relative measurement)

Figure 8 displays the VCXO oscillation frequency offset as function of the power supply voltage. The measurements are made relative to the VCXO frequency when the power supply voltage is 2.5 V. Notice that two curves are plotted: one corresponding to the reference clock in put set to “0” and the other one with the input set to “1”. In absolute value the slopes of both curves are less than 0.13 Hz/mV at V_{dd} = 2.5 V. Notice that the dependence on the power supply increases once the ASIC is powered with a voltage smaller than 2.3 V. Such a regime of operation should be avoided. In other to keep some operation margin it is recommended that the minimum power supply voltage should not be reduced bellow 2.4 V. This is valid for operation both in the PLL mode and on the Clock Source mode.

PCB LAYOUT RECOMMENDATIONS

Frequency pulling considerations

The QPLL is based on a Voltage Controlled Quartz Crystal Oscillator (VCXO). The frequency of oscillation of such circuit is essentially imposed by the quartz crystal resonance frequency. However, the circuit capacitance (which includes the layout parasitics) will also have an influence. In the case of a VCXO, this manifests itself in two ways: first, the oscillation frequency is not exactly the quartz crystal resonance frequency but higher (called the *loaded oscillation frequency*) and second, the frequency pulling capability of the circuit is affected by the total circuit capacitance, in particular by the minimum capacitance achievable.

To cope with any frequency uncertainty the crystal is specified for a given load capacitance. This gives the manufacturer the capability of tuning the crystal to a specific circuit.

Concerning the pulling range, one could be tempted to think that adding as much variable capacitance as possible would be a solution to increase the frequency pulling ability of the circuit. However, in the limit of an infinite load capacitance the oscillation frequency tends to the crystal resonance frequency. In this limit, the frequency sensitivity to capacitance variations is very small and the VCXO has thus a small pulling ability[4]. The solution is thus to work on the extreme of low capacitances where the frequency sensitivity is maximised. Figure 9 illustrates these concepts for a practical crystal (in this figure C12 represents the crystal package capacitance).

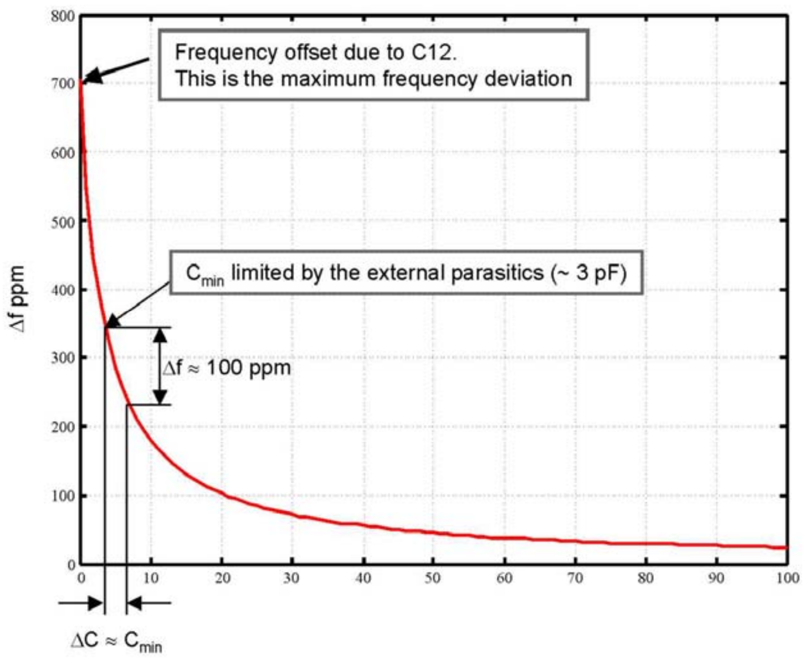


Figure 9 Circuit capacitance versus frequency pulling ability

We are thus faced with two problems: first, minimise the parasitic capacitances introduced by the circuit layout so that the pulling range does not get degraded and

[4] This would be a good solution if the capacitance could be strictly varied from a very small to a large value. However, in practical circuits a large maximum value also implies a relatively large minimum value. That is, the ratio between the minimum and maximum capacitance cannot be freely chosen.

second, make sure that the circuits built by the QPLL users display a load capacitance which is identical to the specified crystal load capacitance. It is thus strongly recommended that the users adopt the layout represented in Figure 10 for the interconnections between the QPLL and the quartz crystal. Failing to do so, it might result in the best case in a reduced or asymmetrical lock range and in the worst case in the impossibility to lock to the LHC frequency. It is thus the user responsibility to follow the recommended layout for the interconnections between the quartz crystal and the QPLL as close as possible.

From a simple parallel plate capacitance calculation the interconnection between Xtal1 and the crystal (crystal and IC soldering pads plus the PCB track) contributes a capacitance to ground of about 0.47pF. This value can be used as a guideline to design the PCB. However, in any case the user should check that the QPLL locking range is well centred around the LHC frequency using the procedure described in the following section.

Layout

As indicated in Figure 5, it is very important to minimize any stray capacitances on node 1 (N1) of the resistive divider. To have an idea of the parasitic capacitances in the circuit, just the crystal soldering pad alone represents about 0.43 pF if the dielectric has a thickness of 800 μm (0.49 pF for 700 μm and 1.72 pF for 200 μm). The parasitic capacitance on node N1 has two detrimental effects: First, it increases the power delivered to the crystal, and second it pulls low the resonance frequency of the circuit. It is thus important to reduce as much as possible the parasitic capacitance on this node. For that, the power and ground planes should be eliminated under the soldering pads of R1, R2 and the crystal soldering pad that is connected to this node. Signal routing must be avoided under this area. Please see Figure 10 for a suggestion on how the layout should be done.

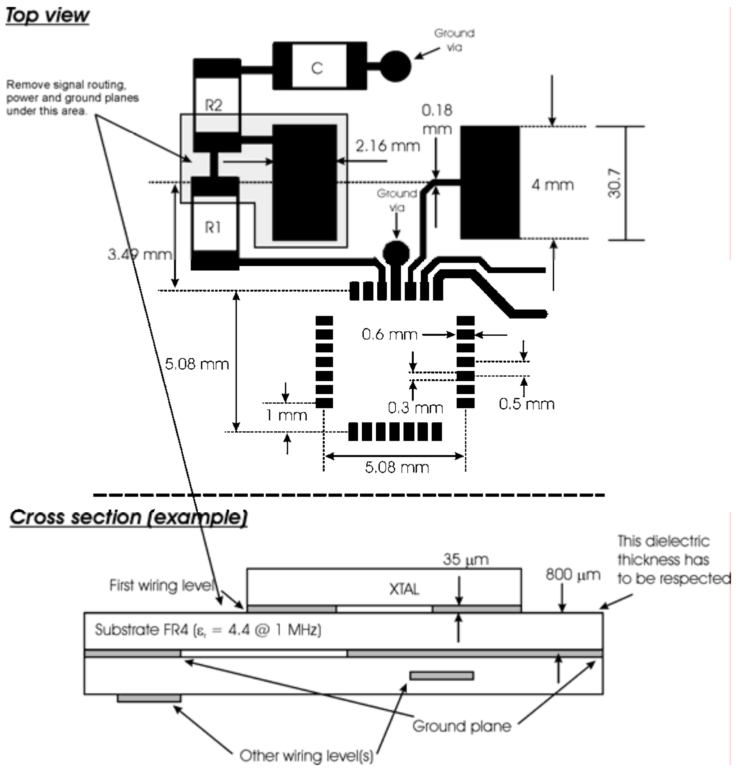


Figure 10 Recommend layout for the QPLL and crystal interconnection

To facilitate the CAD work a schematic capture symbol and the layout footprint of the ASIC are available in the CERN CADENCE library. The footprint is available in the library CNSPECIAL under the name QPLL. For the package type the LPCC option must be used.

As an example of a QPLL circuit the designer can refer to the TTCrq schematics and layout files. Links to these files can be found on the QPLL web site: http://www.cern.ch/proj-qpll.

Procedure to verify the PCB parasitic capacitance

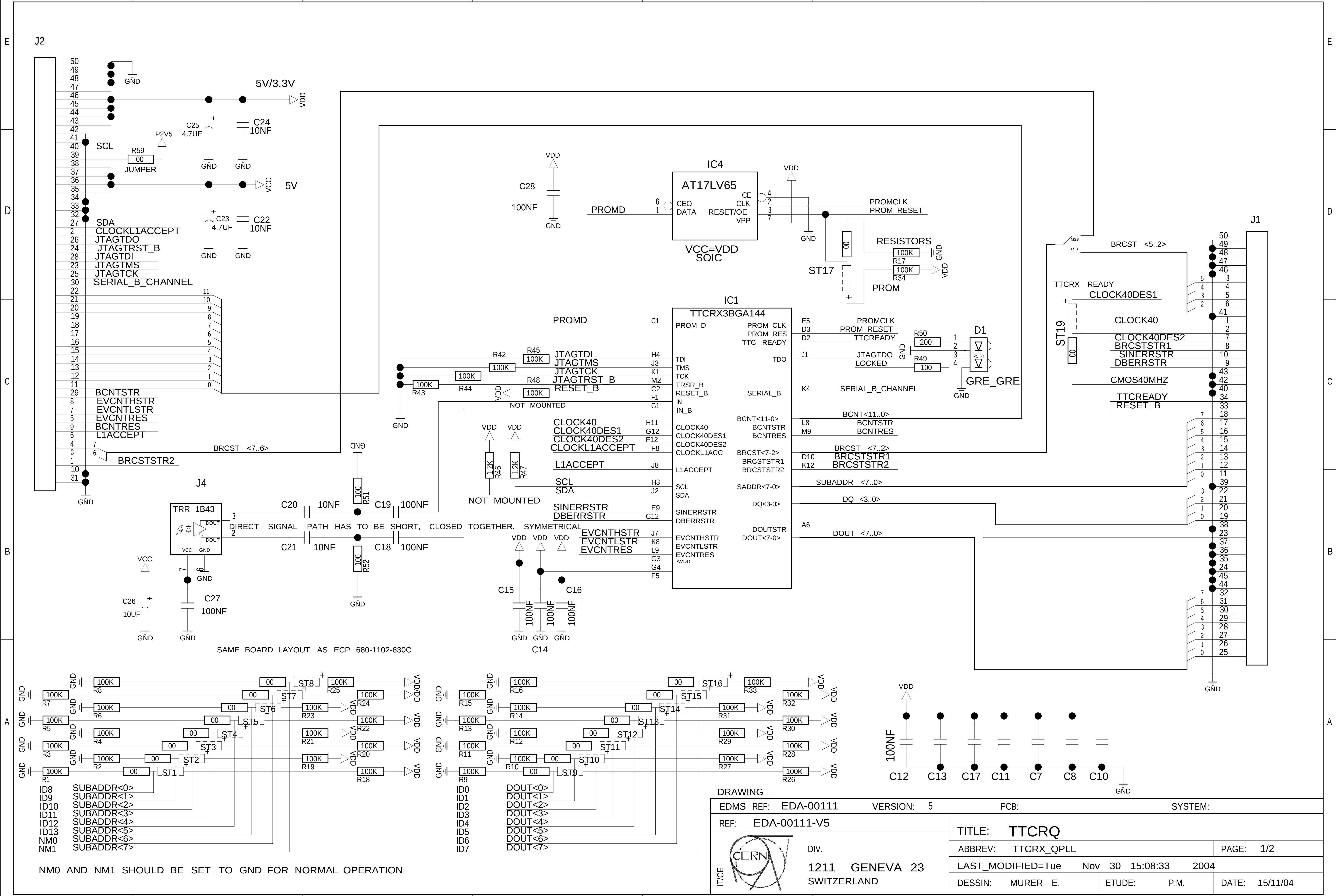
There are two alternative ways to verify that the PCB has been correctly designed:

The first (and simplest) is to check the circuit lock range. This can be done by sweeping the input frequency around the LHC frequency. By approaching the LHC frequency from below the lower locking frequency can be obtained. Similarly, by approaching the LHC frequency from above the upper locking frequency can be determined. The LHC frequency (f_{LHC}) must be well centred within these two limits.
For these measurements the frequency sweep should be done in “digital” steps allowing, at each frequency step, time enough for a calibration cycle to be executed.

The second method consists in measuring the free running oscillation frequency of the PLL. The following procedure needs to be applied: set the signal “*externalControl*” to “1” and the signals “*f_0Select<5:0>*” to “100111”[1] (binary). Then, measure the output frequency while the reference clock input is forced to “0”. This will give the minimum oscillation frequency (*f_0[min]*) for the selected frequency range. Then repeat the measurement forcing the reference clock input to “1”. This will give *f_0[max]* for that range. The average of these two frequencies (*f_0[min]* and *f_0[max]*) must be within: f_{LHC} ±25 ppm.

Please note all the frequency measurements mentioned above need to be done with an <u>absolute accuracy</u> of at least a few parts per million (ppm). Although most laboratory frequency meters are capable of providing such relative accuracy they are rarely that accurate in absolute terms. The solution in that case is to feed the frequency meter with a precise clock signal from a calibrated frequency standard (like for example a GPS based frequency source).
At CERN we are equipped to do such precise frequency measurements and we can help users that aren’t equipped to do so in their own labs.

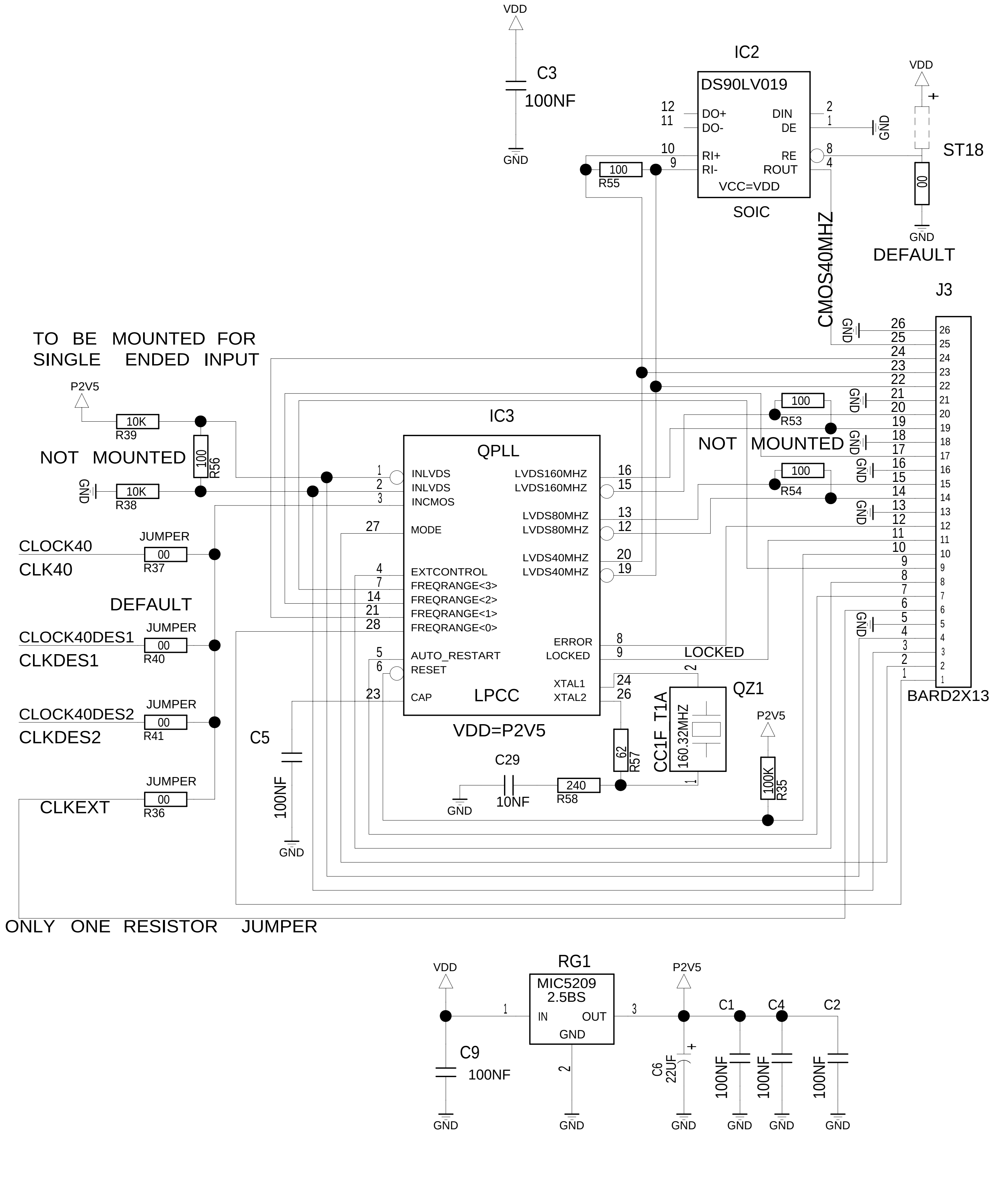
[1] Number to be confirmed once the crystals will be received from the manufacturer.



DRAWING

EDMS REF: EDA-00111	VERSION: 5	PCB:		SYSTEM:	
REF: EDA-00111-V5		TITLE: TTCRQ			
DIV.		ABBREV: TTCRX_QPLL		PAGE: 1/2	
1211 GENEVA 23		LAST_MODIFIED=Tue Nov 30 15:08:33 2004			
SWITZERLAND		DESSIN: MURER E.	ETUDE: P.M.	DATE: 15/11/04	

IT/CE



DRAWING

EDMS REF: EDA-00111 VERSION: 4 PCB: SYSTEM:

REF: EDA-00111-V5

TITLE: TTCRQ

DIV.

1211 GENEVA 23

SWITZERLAND

IT/CE

ABBREV: PAGE: 2/2

LAST_MODIFIED=Mon Jan 03 15:32:42 2005

DESSIN: MURER E. ETUDE: P.M. DATE: 03/01/05