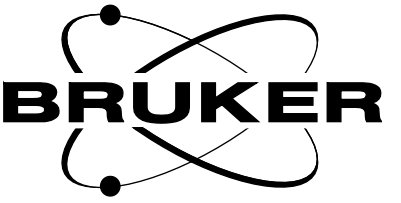


AQX

TCU
Technical Manual

Version 002

BRUKER

This manual was written by

J. M. Rommel, N. Kuntz, T. Eckert

Rheinstetten, Germany

Updated for BASH 2.0 by U. Roos - December 1996

P/N: Z31342
DWG-No. 1051 002

1 : AQX TIMING CONTROL UNIT

Kuntz, Eckert, Rommel

Contents

Figures

Tables

1. Technical Description

1. 1. General Informations

The TCU consists of the two boards *TCU main* and *TCU ext.*
There are the two versions H2558 and H5811/12 of *TCU main* and the two versions H2562 EC<20 and EC≥20 of *TCU ext* .Both versions of each board can be combined but the H5811/12 needs the software release *XWIN-NMR Version 1.0* or later.

TCU Versions	Board	Part No.	Layout No.	EC Level	Software Constrains
TCU0	TCU main	H2558	H3P1860B	EC ≥ 01	
TCU0	TCU ext	H2562	H3P2020A	EC t 20	
TCU0	TCU ext	H2562	H3P2020B	EC ≥ 20	
TCU1	TCU main	H5811/12	H3P1860E	EC ≥ 00	XWIN-NMR Version 1.0
TCU1	TCU ext	H2562	H3P2020A	EC t 20	XWIN-NMR Version 1.0
TCU1	TCU ext	H2562	H3P2020B	EC ≥ 20	XWIN-NMR Version 1.0

Table 1: TCU versions

1. 2. Features

- Minimal duration is 50 nsec
- Maximal duration is 1,6777216375 sec
- Resolution is 12,5 nsec
- 35 real time outputs are exclusively controlled by the real time state machine. They can switch in each system cycle (max. 20 Mhz) and full resolution
- Another 112 real time outputs are controlled by 16 outputs of the state machine. That means that max. 16 out of 112 outputs can switch in each system cycle but each additional group of 16 signals needs one preparation cycle to switch in the same system cycle.
- 64KByte (optinal 128 KByte) Real-Time Acquisition Memory.
- There are 4 trigger inputs, edge or level sensitive
- I80960 Microcontroller with internal 1KB cache and 1KB Data Ram operating at 33MHZ without wait states.
- 32 Bit Data/Address VME Bus Slave controller with interrupt capabilities
- fast local 256KB instruction Ram with 1 wait state and pipeline operation
- special VME-Bus register set (interrupt,status, and configuration)

- Bruker Identification System EEPROM BBIS

1. 3. Architecture

1. 3. 1. Block Diagram

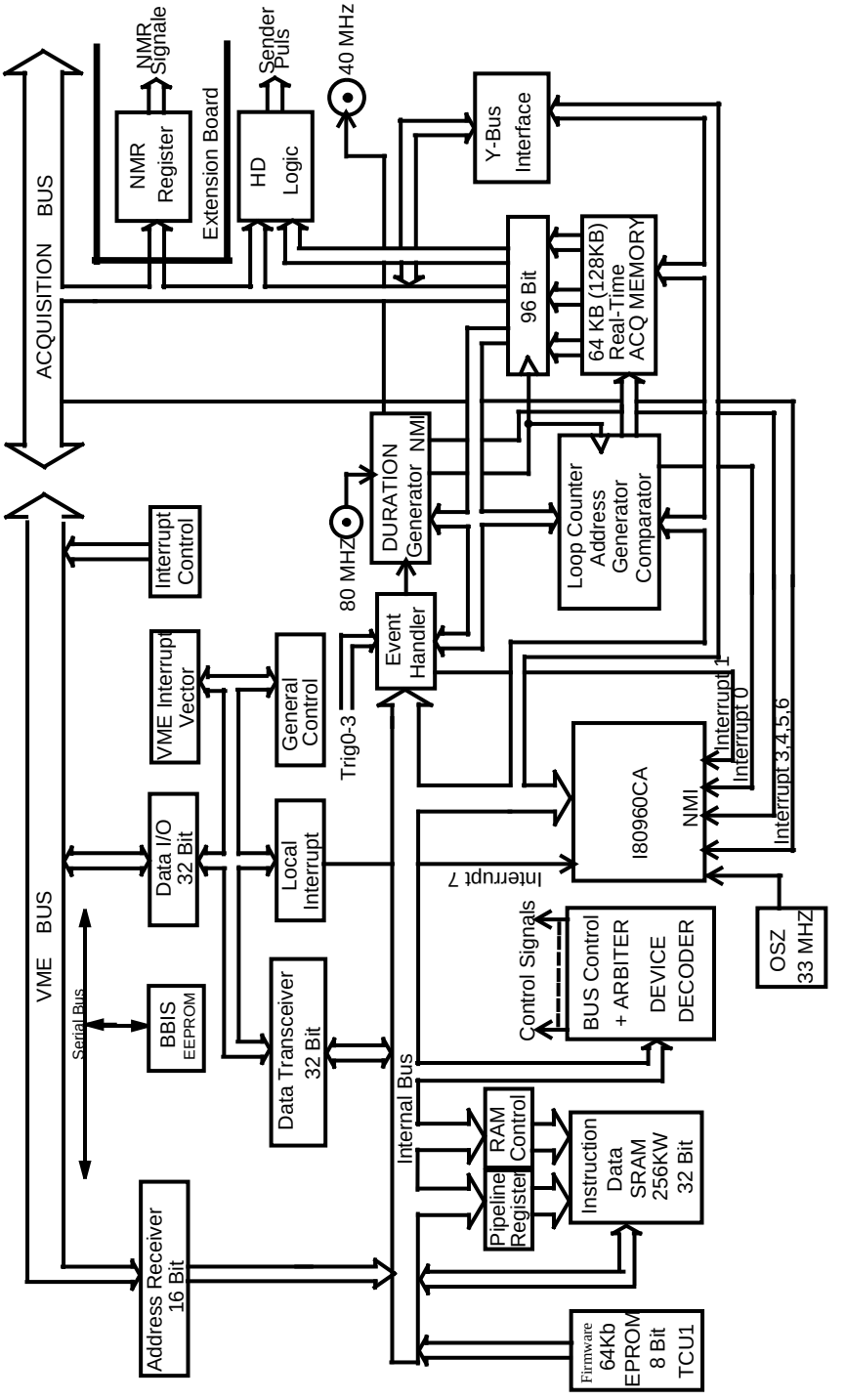


Figure 1: Block Diagram of TCU

1. 3. 2. VME-Bus Interface

The Interface to the VME-Bus is designed as a 32 bit address/data Bus Slave with Interrupt capabilities. To simplify the Interface only 32/16 bit data access are possible (no nibble or page access). The instruction and data ram and nearly all registers are read-/ writeable from the VME-Bus which is important for testing. All functions, witch normally are controlled by the I960, can also be activated from a VME-Bus Master.

1. 3. 3. Arbiter between I960 and VME-Bus

Any access from the VME Bus or I960 to the resources on the TCU are supervised by the Arbiter. Whereas I960 accesses has higher priority as VME-Bus accesses. There are two possible ways for the Arbiter to allocate accesses from the VME Bus to the TCU.

1. The ready signal of the I960 is controlled by the arbiter . An access from the VME- Bus to the data Ram or VME register set is delayed until the ready signal is inactive. In the other case the I960 waits for ready be active, when it access this region ,before it continues its current cycle.

2. The Hold Request signal on the I960 is set by the arbiter if a VME Master wants access the instruction Ram or TCU Register Set (Loop counter, Address generator, Duration generator). The I960 signals to the arbiter with the HOLDACK line that it has released the intern Bus and the VME-Bus access is granted.

1. 3. 4. Reset Control

The reset logic is used to initialize the Timing Control Unit and I960 processor. Three conditions will activate the reset logic:

1. Power up reset

2. VME-Bus System reset

3. Software reset

After RESET the control logic on the board is ready to work whereas the I960 will be stay in RESET STATE until a GO command is given by the Host CPU. In this state the CPU can load the I960 instruction ram with executable program code or the proper operation of the TCU can be tested via VME -Bus.

1. 3. 5. I960 Processor Kernel

An important part of the Timing Control Unit is the Intel I960 Microprocessor operating at 33MHZ with build around 256KB fast (25ns) SRAM and 64KB EPROM. The EPROM (8 bit data width) contains the initial boot record for the I960 after power-up and the GNU debugger software. The operation mode (i.e. Bus configuration, access speed, ready timing) of the I960 bus controller is programmable. For a detailed description see chapter ”Memory region configuration”. In this application the I960 operates with the external 32 bit instruction RAM in pipeline mode ,with 0 wait state on read and 2 wait state on write cycles. Fast PALs are used to generate the ram address for pipeline operation. The I960 Address Mapping is decoded by the TCU internal device decoder. The I960 has an on chip interrupt controller with 8 dedicated interrupt pins and a separate NMI input. The interrupt are low level detected.

1. 3. 5. 1. I960 Address Mapping

The i960 Processor has an linear address space from 0 to 2^(31) -1. Some of this address space is reserved or assigned to special functions. A memory address is a 32 bit value within 0 and FFFFFFFF Hex . It can be used to reference single byte , 2 bytes , 4 bytes , double word (8 bytes), triple word (12 bytes) or quad words (16 bytes) in memory, depending on the instruction being used. The i960 address space is shown below:

1. 3. 5. 2. i960 global Address Table

0000 0000 - 0000 0003 NMI VECTOR

0000 0004 - 0000 003F INTERNAL DATA RAM (OPT. INTERRUPT VECTORS)

0000 0040 - 0000 00BF INTERNAL DATA RAM (OPTIONAL DMA REGISTER)

0000 00C0 - 0000 03FFINTERNAL DATA RAM (USER WRITE PROTECTED)

0000 0400 - FEFF FFFF EXTERNAL MEMORY CODE OR DATA

FF00 0000 - FFFF FEFFRESERVED

FFFF FF00 - FFFF FF2CINITIALIZATION BOOT RECORD

FFFF FF2D - FFFF FFFFRESERVED MEMORY

1. 3. 5. 3. Memory region Configuration

The address space of the i960 can be mapped as read-write, read-only memory and memory mapped I/O. The whole memory space is divided into 16 regions each 256 MBytes in range. The upper four bits of the address (A31 - A28) indicate which of the 16 regions is currently selected. The memory configuration for the TCU1 is as follows. The regions 0 to 7 are reserved for the inst./data SRAM. The 1KB internal RAM of the i960 is mapped into region 0. The internal RAM of the I960 can only be accessed by loads, stores, or DMA instructions. The regions 8,9,A are reserved for user definable memory space (i.e. VME DMA I/O). Region B is used for VME I/O devices codes i.e VME interrupt vector register, VME address modifier register, General Control register , and local interrupt register. Region C is used for special AQ-Bus register the duration- and address generator. Region E is used for the 4 port Real-Time Ram whereas Region F is used for the EPROM boot sequence and for monitor code. The memory table entry is shown below:

1. 3. 5. 4. Memory Region Configuration of i960

0000 0000 - 0FFF FFFFREGION 0SRAM_CONF (internal 1KB data RAM)
1000 0000 - 1FFF FFFFREGION 1EXTERNAL 256 KB I/D SRAM CONF.
2000 0000 - 2FFF FFFFREGION 2 SRAM_CONF
3000 0000 - 3FFF FFFFREGION 3SRAM_CONF
4000 0000 - 4FFF FFFFREGION 4 SRAM_CONF
5000 0000 - 5FFF FFFFREGION 5 SRAM_CONF
6000 0000 - 6FFF FFFFREGION 6SRAM_CONF
7000 0000 - 7FFF FFFFREGION 7SRAM_CONF
8000 0000 - 8FFF FFFFREGION 8RESERVED FOR VME-BUS DMA
9000 0000 - 9FFF FFFFREGION 9 ” ”
A000 0000 - AFFF FFFFREGION A ” ”
B000 0000 - BFFF FFFFREGION BVME_DEVICE_CODE
C000 0000 - CFFF FFFFREGION CADD. and DURATION GEN., AQ-BUS, LOOP Counter
D000 0000 - DFFF FFFFREGION Dnot used
E000 0000 - EFFF FFFFREGION E4 PORT REAL-TIME RAM.
F000 0000 - FFFF FFFFREGION FEPROM_CONF

Table 1: Memory Region Table (256 Mbytes step)

Each region has independent software programmable parameters that define the data-bus width, ready control, number of wait states, pipeline read mode, byte ordering and burst mode. These parameter are stored in the memory-region configuration table. The purpose of configurable memory regions is to provide system hardware

interface support. Because of slow external memory devices the i960 must generate wait states for any region. Five parameters define the wait-state-generator operation:

NRAD - Number of wait cycles for Read Address-to-Data
NRDD - Number of wait cycles for Read Data-to-Data
NWAD - Number of wait cycles for Write Address-to-Data
NWDD - Number of wait cycles for Write Data-to-Data
NXDA - Number of wait cycles for X (read or write)Data-to-Address

Region 0-7 Inst. SRAM configuration table (region 0-7) with 25 ns access time Data-Bus Width 32 Bit

NRAD	1 wait states
NRDD	1 wait states
NWAD	2 wait states
NWDD	2 wait states
NXDA	0 wait states
Pipelining	YES
External Ready	NO
BURST	YES

Region B configuration table VME_DEVICE_ Code

Data-Bus Width	32 Bit
NRAD	3 wait states
NRDD	1 wait states
NWAD	3 wait states
NWDD	1 wait states
NXDA	0 wait states
Pipelining	NO
External Ready	YES
Burst	NO

Region C configuration table Address/Duration Gen. and AQ-Bus Register

Data-Bus Width	16 Bit
NRAD	32 wait states
NRDD	32 wait states
NWAD	32 wait states
NWDD	32 wait states
NXDA	0 wait states
Pipelining	NO
External Ready	NO
Burst	NO

Region E configuration table for 4 Port real-time Ram

Data-Bus Width	32 Bit
NRAD	2 wait states
NRDD	2 wait states
NWAD	2 wait states
NWDD	2 wait states
NXDA	0 wait states
Pipelining	NO
External Ready	NO
Burst	Yes

Region F EPROM_DEVICE Code configuration table

Data-Bus Width	8 Bit
NRAD	6 wait states
NRDD	3 wait states
NWAD	5 wait states
NWDD	3 wait states
NXDA	1 wait states
Pipelining	NO
External Ready	NO
Burst	NO

1. 4. Logical References

Table 2: Memory Map and Device Codes

Address of VME Acc.	Address of Local Acc.	Destination	name	Mode R/W	Size of Operand Byte 3 2 1 0
19100000 to 191FFFFC	10000000 to 100FFFFC	Instruction and Data RAM of ”i960”, 1M byte		R/W	b b b b
19200000 to 1921FFFC	E0000000 to E001FFFC	4-Port-RAM for Real Time Operands 16k words of 32 bit (optinal 32K words 32 bit)		R/W	b b b b
19220000	B0000000	Interrupt Vector of TCU on VME Bus	viv	R/W	x x x b
19220004	B0000004	VME Control Register	ctrl	R/W	x x x b
19220008	B0000008	Debug and local Interrupt Register of ”i960”	li	R/W	x x x b
19220010	no access	Software initiated Hardware Reset of the whole board	res	W	x x x x
19220014	no access	Go-Command to ”i960”	go	W	x x x x
19221010	C0000010	Set Interrupt Control Register1		W	x x x b
19221010	C0000010	Clear and disable Interrupt 1		R	x x x x
19221020	C0000020	Read Debugging Signals of duration generator and loop counter		R	x x x b
19221028	C0000028	Read Trigger Inputs		R	x x x b
19221030	C0000030	Clear and disable NMI		R	x x x x
19221040	C0000040	Enable Interrupt 0		W	x x x x
19221040	C0000040	Clear and disable Interrupt 0		R	x x x x
19221050	C0000050	Clear and disable WAIT		R	x x x x
19221060	C0000060	Set HMD1 Register		W	x x x b
19221068	C0000068	Set HMD2 Register		W	x x x b
19221080	C0000080	RUN Start of Clock Generator Run of Address Generator		W	x x x x
19221084	C0000084	STEP Clock Generator switch-off, (The effect of each STEP is the next clock edge, high>low>high>low)		W	x x x x
19221088	C0000088	LDREG Load Start Register of Address Generator, Clock Generator keeps running		W	x x b b
1922108C	C000008C	STOP Stop of Clock Generator, clock at ”High”		W	x x x x

Address of VME Acc.	Address of Local Acc.	Destination	name	Mode R/W	Size of Operand Byte 3 2 1 0
19221090	C0000090	START Start of Clock Generator Start of Address Generator at operand		W	x x b b
19221094	C0000094	Address Generator step/clock		W	xxxx
19221098	C0000098	LDADDR Load of Address Generator if Clock Generator is off, Clock Generator stays off		W	x x b b
1922109C	C000009C	DEVST Load Address Generator with value of Start Register (During RUN this address becomes valid after the actual duration		W	x x x x
192210C0	C00000C0	RDADDR Read of Address Generator Register		W	x x b b
19221100	C0000100	INIT Stop of Clock Generator Clear all registers of Address Generator		W	x x x x
19221104	C0000104	Read NMR1-Preregister into NMR-Reg.		R	x x b b
19221108	C0000108	Read NMR2-Preregister into NMR-Reg.		R	x x b b
1922110C	C000010C	Read NMR3-Preregister into NMR-Reg.		R	x x b b
19221110	C0000110	Read NMR4-Preregister into NMR-Reg.		R	x x x b
19221114	C0000114	Read NMR5-Preregister into NMR-Reg.		R	x x b b
19221118	C0000118	Read NMR6-Preregister into NMR-Reg.		R	x x x b
1922111C	C000011C	Read NMR7-Preregister into NMR-Reg.		R	x x b b
19221120	C0000120	Read NMR8-Preregister into NMR-Reg.		R	x x b b
19221124	C0000124	Read NMR1-Register into NMR-Reg.		R	x x b b
19221128	C0000128	Read NMR2-Register into NMR-Reg.		R	x x b b
1922112C	C000012C	Read NMR3-Register into NMR-Reg.		R	x x b b
19221130	C0000130	Read NMR4-Register into NMR-Reg.		R	x x x b
19221134	C0000134	Read NMR5-Register into NMR-Reg.		R	x x b b
19221138	C0000138	Read NMR6-Register into NMR-Reg.		R	x x x b
1922113C	C000013C	Read NMR7-Register into NMR-Reg.		R	x x b b
19221140	C0000140	Read NMR8-Register into NMR-Reg.		R	x x b b
19221150	C0000150	Read BLKTRA(8..15) into NMR-Reg. B1 Read BLKTR(1,2,5,6) and BLK-GD(X,Y,Z) into NMR-Reg. Byte0		R	x x b b
19221154	C0000154	Read BLKTRA(1..8) into NMR-Reg. B1 Read BLKTRC(1..8) into NMR-Reg. Byte0		R	x x b b

Address of VME Acc.	Address of Local Acc.	Destination	name	Mode R/W	Size of Operand Byte 3 2 1 0
19221158	C0000158	Read BLK_F(1..4) and SP_F(1..4) into NMR-Reg. Byte0		R	x x x b
1922115C	C000015C	Read SP_PA(1..4) and SP_FA(1..4) and OB-SCH(1..4) and TUNE,RCP0 into NMR.		R	x x b b
19221300	C0000300	Assert Interrupt to VME-Bus, Level defined by Jumper W8		W	x x x x
19221310	C0000310	Read Y-Bus Data back		R	x x x b
192213F0	C00003F0	Read NMR-Register into port 4		R	b b b b
192213F8	C00003F8	Read NMR-Buffer into port 4		R	b b b b
192213E0	C00003E0	Read AQ-Bus into port 4 Read SPP-Register into port 3		R	b b b b
19221400 to 192217FC	C0000400 to C00007FC	Read or write 8 bit data from or to Y-Bus		R	x x x b

1. 4. 1. Soft Reset

The Host CPU can reset the TCU1 via this device code and the I960 processor will go in the reset state until it will be started with the Go command. While the I960 processor is in reset state a VME-Bus Master can access the internal instruction RAM, for example to load a program or to test this RAM. In the reset state the front LEDs 'FAIL' and 'HOLD' are switched on.

1. 4. 2. Go Command

This Device Code is used to release the I960 processor from the reset state. The I960 will than execute the initial program located in the PROM and than it will branch to the instruction Ram.

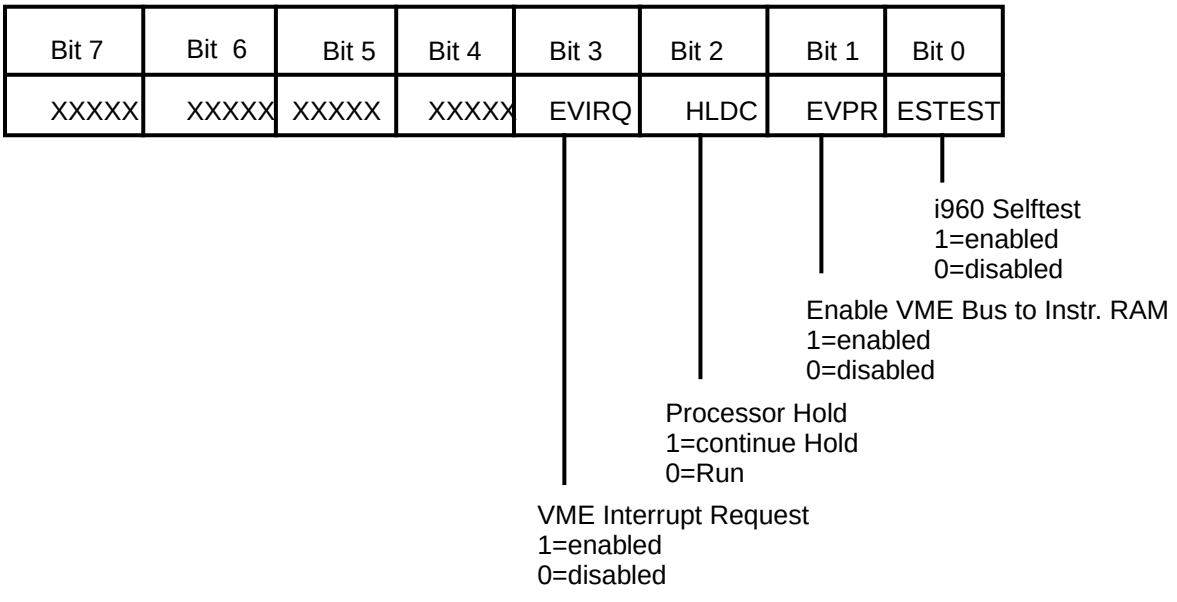
1. 4. 3. VME Interrupt Request

The I960 processor or the Host CPU can write to this Device code to generate an Interrupt request on the VME -Bus. Prior to this action the 'EVIRQ' bit in the General Control register must be set to enable this function.

1. 4. 4. General Control Register

The General Control register is an 8 bit read/writeable register used to control the global operation of the I960 processor from the host CPU.

Figure 2: General Control Register



ESTEST : If set (1) the internal selftest on the I960 is enabled . The I960 will execute the selftest operation after reset and test its internal register and bus structure and the external Bus for any contention. If the Test passes through without an error the front LED is switched off, otherwise the LED will be on.

EVPR : Setting this Bit to 1 allows a VME-Bus Master to access the I960 local instruction Ram without prior setting the I960 in the Hold or Reset State. Therefor the I960 will only be in Hold State for the duration of the VME access cycle.
If set to 0 the local instruction Ram is not accessible by a VME-Bus Master until the I960 is in reset or continue Hold State.

HLDC : If set to 1 the I960 will go into the continue Hold state. For example to load a new program or initialize the I960 . The I960 will leave the Hold state if this bit is set to 0.

EVIRQ : If set 1 the I960 can generate an interrupt request on the VME-Bus .When set to 0 the interrupt to the VME-Bus is disabled.

1. 4. 5. Interrupt Control Register

Interrupt 1 of ”i960” can be initiated by one of the 4 trigger inputs TRIG0, TRIG1, TRIG2 or TRIG3. Associated to interrupt 1 is the Interrupt-Control-Register. It can be accessed via VME-Bus or by the ”i960” (see Table 3)

Table 3: Binary Format of Interrupt Control Register

D7	D6	D5	D4	D3	D2	D1	D0	Meaning
x	x	x	0	0				TRIG0
x	x	x	0	1				TRIG1
x	x	x	1	0				TRIG2
x	x	x	1	1				TRIG3
x	x	x			0	0	0	INT if trigger signal at low level
x	x	x			0	0	1	INT if trigger signal at high level
x	x	x			0	1	0	INT if trigger signal goes from high to low
x	x	x			0	1	1	INT if trigger signal goes from low to high
x	x	x			1	0	0	INT at any transition of trigger
x	x	x			1	0	1	not defined
x	x	x			1	1	0	NO INT possible, masked
x	x	x			1	1	1	INT set independent of any trigger signal

The interrupt has to be cleared via VME-Bus or by local access (see Table 2).

Clearing the interrupt sets also the bits D2, D1and D0 to 1.

1.4.6.VME Interrupt Vector Register

An 8 Bit width register used to store the appropriate interrupt vector for an VME-interrupt acknowledge cycle. It should be written by the Host CPU before any interrupt action of the TCU1 is started.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VD7	VD6	VD5	VD4	VD3	VD2	VD1	VD0

Figure 3: Interrupt Vector Register

1. 4. 6. NMI Control Register

The NMI of ”i960” can be initiated by one of the 4 trigger inputs TRIG0, TRIG1, TRIG2 or TRIG3 and by the proper operand of the ”Real-Time-Program-Word” (see Table 4).
The necessary settings of the NMI-Control-Register can only be carried out by an instruction and operand out of the ”Real-Time-Program-Word” Port2 (D31..D22).

The NMI has to be cleared via VME-Bus or by local access (see Table 2)

Table 4: Binary Format of NMI control register instruction and operand

D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	Meaning
1	1	1	0	0						Set NMI control register with D26,...D22
					0	0				TRIG0
					0	1				TRIG1
					1	0				TRIG2
					1	1				TRIG3
							0	0	0	NMI if trigger signal at low level
							0	0	1	NMI if trigger signal at high level
							0	1	0	NMI if trigger signal goes from high to low
							0	1	1	NMI if trigger signal goes from low to high
							1	0	0	NMI at any transition of trigger
							1	0	1	not defined
							1	1	0	NO NMI possible, masked
							1	1	1	NMI set independent of any trigger signal

1. 4. 7. The WAIT Control Register

The WAIT logic provides the hold function of the real time control unit.
The hold time depends on the conditions in the ”Meaning” column. Reset or a special device code stop the WAIT situation (see Table 2). The Wait Control Register is loaded from the Port2 of the ”Real-Time-Program-Word” (D31..D22).

Table 5: Binary Format of WAIT control register.

D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	Meaning
1	1	0	1	1						Set WAIT control register with D26,...D22
					0	0				TRIG0
					0	1				TRIG1
					1	0				TRIG2
					1	1				TRIG3
							0	0	0	WAIT if trigger signal at low level
							0	0	1	WAIT if trigger signal at high level
							0	1	0	WAIT until trigger signal goes from high to low
							0	1	1	WAIT until trigger signal goes from low to high
							1	0	0	WAIT until any transition of trigger
							1	0	1	not defined
							1	1	0	Unconditional WAIT
							1	1	1	NO WAIT

1. 4. 8. Interrupt 0

Interrupt 0 of ”i960” is used as error interrupt. It is released by the run out condition in the real time program memory.

Interrupt 0 can be enabled, cleared and disabled via VME-Bus or by local access
(see Table 2).

1. 4. 9. The Real-Time-Program-Word

The real time state machine is built around a 4 -Port-RAM of 16k words by 32 bits.
Three ports (port 2,3 and 4) are read out in one step forming a 96 bits wide ”Real-Time-Program-Word”. The 32 bit words appearing at each port must be fed to different addresses at port 1.

1. 4. 9. 1. Address Generator of the Real Time State Machine

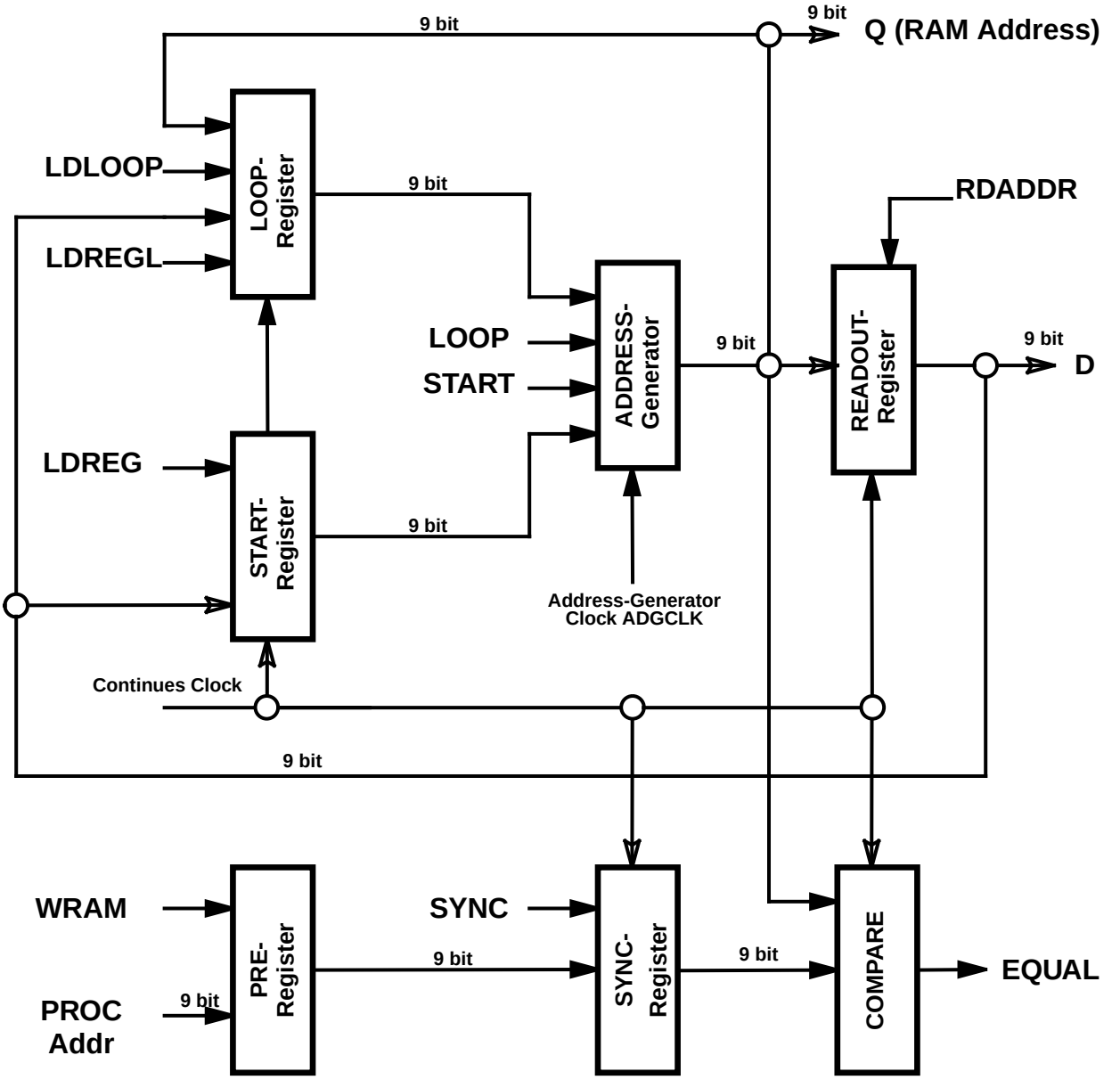


Figure 4: Address Generator

Table 6: Real-Time-Program-Addresses

Address of VME Acc.	Address of Local Acc.	Part of Real-Time-Program-Word	name	Mode R/W	Size of Operand Byte 3 2 1 0
19200xx0 to 1921xxx0	E0000xx0 to E0001xx0	Real Time Control Signals at port 2		w	b b b b
19200xx4 to 1921xxx4	E0000xx4 to E0001xx4	Output Signals at port 3		w	b b b b
19200xx8 to 1921xxx8	E0000xx8 to E0001xx8	Device Control Signals at port 4		w	b b b b
19200xxC to 1921xxxC	E0000xxC to E0001xxC	not used		w	b b b b

Table 7: Format of Output-Signal-Word at port 4 of the Real-Time-Program-Memory

Bit/ Data Lines	Signal	Function
31	BLKGDZ	Gradient Blanking Puls
30	BLKGDY	”
29	BLKGDX	”
28	AQRCUEN	Generate RCU-GO Pulse (high activ)
27	AQ-Bus ENABLE	initiate an AQ-Bus Cycle (high activ)
26	AQEXEC	AQ-Bus EXEC Signal starts action on the FCU or GCU
25	Write NMR-Reg.	Generate a write cycle to the NMR register (activ high)
24	APEXEC	
23	AQA(3)	AQ-Bus Address Bit
22	AQA(2)	”
21	AQA(1)	”
20	AQA(0)	”
19	AQS(3) or NMRA(3)	AQ-Bus sub. Add. or NMR Reg. Address
18	AQS(2) or NMRA(2)	”
17	AQS(1) or NMRA(1)	”
16	AQS(0) or NMRA(0)	”
15..0	AQD(15..0) NMRD(15..0)	AQ-Bus or NMR Reg. Data Bus

Table 8: Format of Output-Signal-Word at port 3 of the Real-Time-Program-Memory

Bit/ Data Lines	Signal	Homodecoupling
31	SP_F8	no
30	SP_F7	no
29	SP_F6	no
28	SP_F5	no
27	SP_F4	Homodecoupling switched together with BLNK_F1,...,BLNK_F4 by common bits in HMD1
26	SP_F3	
25	SP_F2	
24	SP_F1	
23	BLNK_F8	no
22	BLNK_F7	no
21	BLNK_F6	no
20	BLNK_F5	no
19	BLNK_F4	Homodecoupling switched together with SP_F1,...,SP_F4 by common bits in HMD1
18	BLNK_F3	
17	BLNK_F2	
16	BLNK_F1	
15	SP_PA8	no
14	SP_PA7	no
13	SP_PA6	no
12	SP_PA5	no
11	SP_PA4	Homodecoupling switched by specific bits in HMD1
10	SP_PA3	
9	SP_PA2	
8	SP_PA1	
7	BLNK_TR8	Homodecoupling switched by specific bits in HMD2
6	BLNK_TR7	
5	BLNK_TR6	
4	BLNK_TR5	
3	BLNK_TR4	
2	BLNK_TR3	
1	BLNK_TR2	
0	BLNK_TR1	

Table 9: Binary Format of the Real-time-program word Port2.

D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	Meaning
0	D	D	D	D	D	D	D	D	D	load long Duration
1	0	D	D	D	D	D	D	D	D	load loop counter
1	1	0	0	0	1	X	X	X	X	HMD on
1	1	0	0	0	0	X	X	X	X	HMD off
1	1	0	1	1	TS1	TS0	CW2	CW1	CW0	WAIT Control word
1	1	1	0	0	TS1	TS0	CI2	CI1	CI0	NMI Control Word
1	1	1	1	0	TS1	TS0	0	0	X	conditional loop back
1	1	1	1	0	X	X	1	1	0	uncond. loop back
1	1	1	1	1	X	X	X	X	X	Decr. loop counter and loop if > 0

* In all cases when Bit D31 is high a short Duration (D13..D2) is loaded.
* TS1,TS0 are the Trigger select bits, D are Data bits, CW2..1 are the Wait control bits and CI2..0 are the INT. control bits.

Table 10: Format of Output-Signal-Word at port 2 of the Real-Time-Program-Memory

Bit/ Data Lines	Signal	Function
31..22	DP2(31..22)	Set the Wait and NMI Control Reg. (see table 4,5)
29..14	DP2(29..14)	Loop counter input
30..4	DP2(30..4)	Long Duration Data Bits
13..2	DP2(13..2)	Short Duration Data Bits
5,4	DP2(5,4)	High resolution control of the Output switching in steps of 12,5ns-100ns
2	DP2(2)	shift the phase of the 40MHz Clock in steps of 12,5ns relativ to the rising etch of the AQSTROBE

1. 4. 10. Control Register HMD1 and HMD

Each bit of HMD1 and HMD2 switches the homodecoupling modification of specified signals on or off (see Table 8 and 11).
A ”0” at the corresponding bit position means, no homodecoupling of this signal, a ”1” switches the real time modification on.

Table 11: Format of Register HMD1 and HMD2

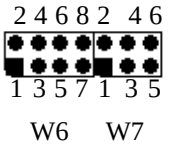
Bit/ Data Lines	HMD1	HMD2
0	BLNK_F1/SP_F1	BLNK_TR1
1	BLNK_F2/SP_F2	BLNK_TR2
2	BLNK_F3/SP_F3	BLNK_TR3
3	BLNK_F4/SP_F4	BLNK_TR4
4	SP_PA1	BLNK_TR5
5	SP_PA2	BLNK_TR6
6	SP_PA3	BLNK_TR7
7	SP_PA4	BLNK_TR8

1. 5. Operational Settings

1. 5. 1. Configuration

1. 5. 2. VME Interrupt Request (Jumper W6 and W7 on H5811)

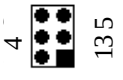
The VME Interrupt request lines are selected by these jumpers. Only one jumper should be set.



W6	W7	IRQ
1-2	---	1
3-4	---	2
5-6	---	3
7-8	---	4
---	1-2	5
---	3-4	6
---	5-6	7

1. 5. 3. VME Interrupt Level (Jumper W5 on H5811)

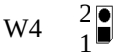
This jumper is used to set the appropriate VME Interrupt Level . The configuration should correspond to the setting of the Interrupt request jumper.



W5 5-6	3-4	1-2	INT. LEVEL
IN	IN	IN	X unused no valid conf.
IN	IN	OUT	1
IN	OUT	IN	2
IN	OUT	OUT	3
OUT	IN	IN	4
OUT	IN	OUT	5
OUT	OUT	IN	6
OUT	OUT	OUT	7

1. 5. 4. Clock Divide (Jumper W4 on H5811)

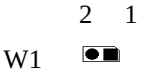
The system clock driven by the I960 can be divided by two with this jumper.



1-2	I960 Clock Frequenz
OUT	33 MHz
IN	16 MHz

1. 5. 5. Real-Time ACQ. Memory Size. (Jumper W1)

The addressable Range of the Address Generator can be selectect with this jumper.



1-2	Address Size
OUT	4K WORDS a 96 bits (standard)
IN	8K WORDS a 96 bits (optional if IDT7025 Memorys are inserted)

1. 5. 6. Extern/Internal 80MHz TTL Signal selection (Jumper W8)

In place of using the 80MHz (600mv) Signal from the PTS a 80MHz TTL Signal can be provided.The Translater on the TCU must then be disabled.

W8	Configuration
1-2	External 80MHz TTL Signalmust be provided
2-3	Internal 80Mhz Translater (Sinus to TTL) is used (default).

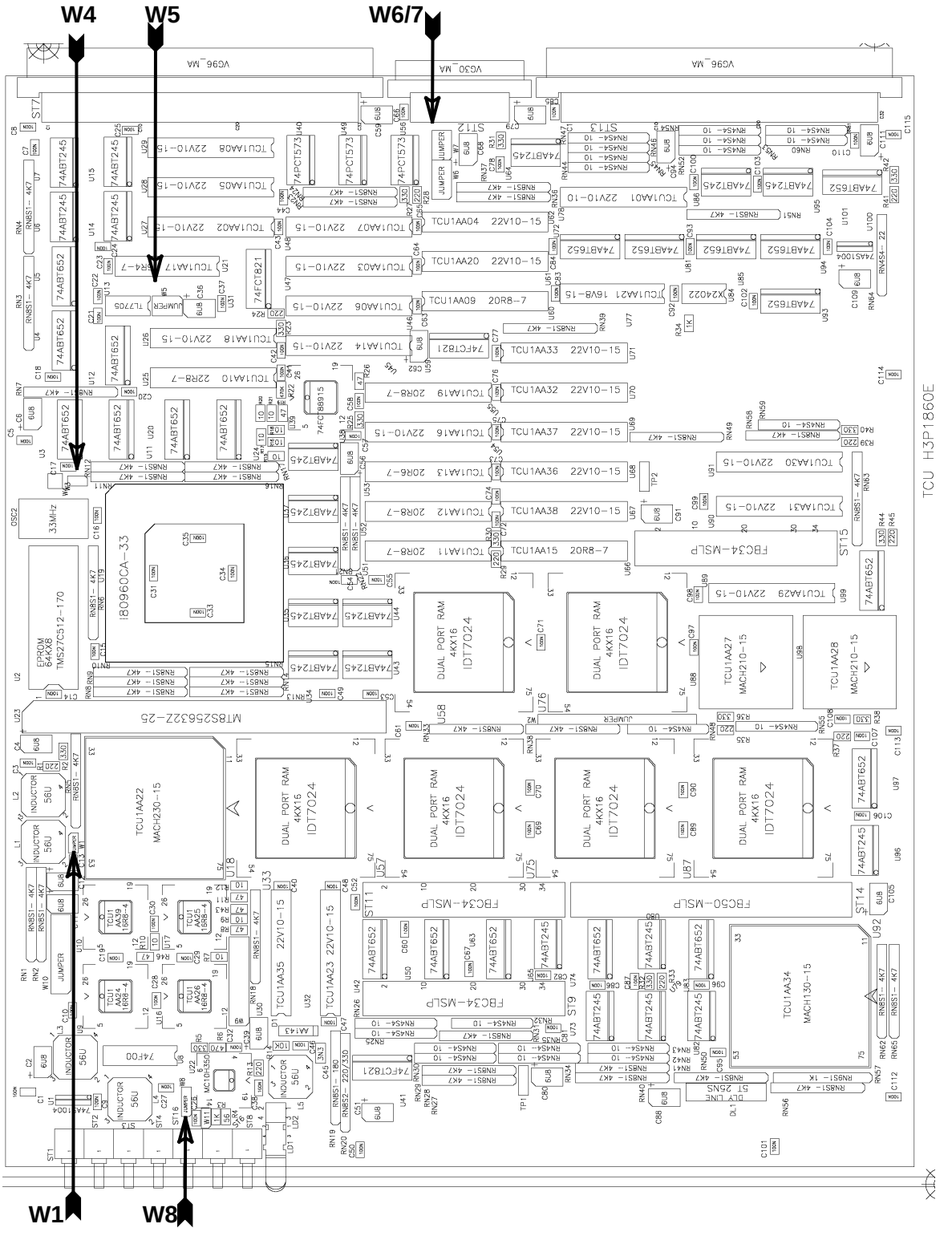


Figure 5: Location of Jumpers on H5811

1. 5. 7. The Bruker Identification System

An Identification-element consists of an interface- PAL and a group of 1 to 8 EEPROM's.Up to 64 interface-PAL's can be connected to the "Serial Bus". The bus consists of 1 data line and 2 clock lines.
ACLK Address clock Controls the communication between the Master and the interface PALs. It is connected to the VME connector pin J2-b3.
SCLK Serial clock -Controls the communication between Master and EEPROMs. It is connected to the VME connector pin J1-b21 .
SDA Serial data -Address, command and Data serial line. It is connected to the VME connector pin J1-b22 .

The data structure is defined in the Bruker BBIS norm from Aug. 16.1993.

The "Serial Bus address" - (SBA) is a 10 bit address and it is parted in three sections:

- SBA 0 - 23-bit EEprom address, inside of each EEprom group.
- SBA 3 - 86-bit Interface PAL address. Address of EEprom groups.
- SBA 91 protocol bit for Future extensions.

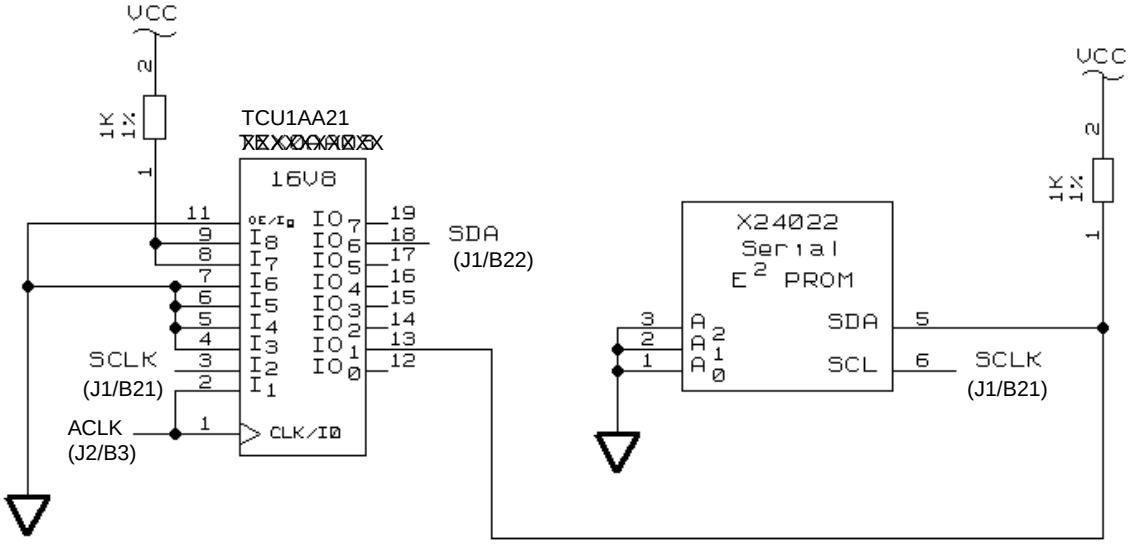


Figure 6: Schematic of the Identification Element on the TCU

The schematic shows following hardwired address inputs:

- Pin 1, 2, 3 of the EEPROM show the hardwired EEPROM address "0"
- Pin 4 (MSB) through 9 of the PAL device are wired to match with group address "03" (A5 through A0)

1. 5. 8. Selecting the input location of signal TRIG2 and TRIG3

There are 4 external trigger signals TRIG0,...,TRIG3. TRIG0 and TRIG1 have a fixed input at the front panel. The input of TRIG2 and TRIG3 is selectable by setting jumper W3 and W4 on the TCU Extension board H2562 as follows.

Jumper W3 on H2562	
connected pins	**Selection**
1 - 2	TRIG3 at Connector T1, pin 11
2 - 3	TRIG3 at Connector T3, contact NN

Jumper W4 on H2562	
connected pins	**Selection**
1 - 2	TRIG2 at Connector T1, pin 25
2 - 3	TRIG3 at Connector T3, contact LL

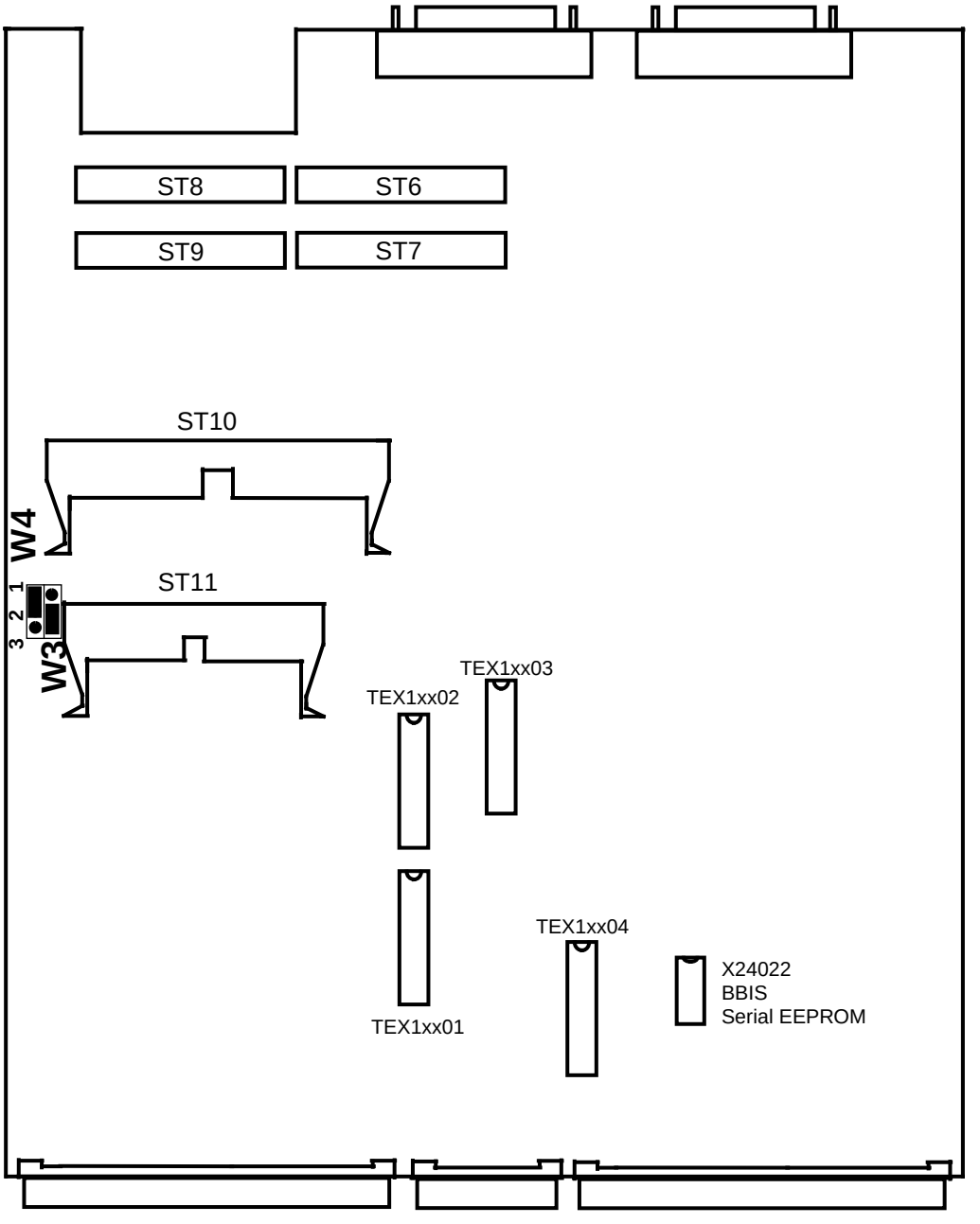


Figure 7: Location of Jumpers on H2562 TCU Extension

1. 5. 9. Controlling Elements

There are no other elements on both TCU boards.

1. 6. Specifications and Connections

1. 6. 1. Construction and Board Size

The TCU1 occupies 12 TE of an EURO-Norm-Rack. It consists of two boards of full plug-in-length, assembled in a distance of 8 TE and connected via two flat ribbon cables. These two boards are named as

TCU-MAIN at the left side and
TCU-EXT at right side of TCU.

The board size is 280mm by 235,35mm.

1. 6. 2. Location of Connectors and Controlling Elements

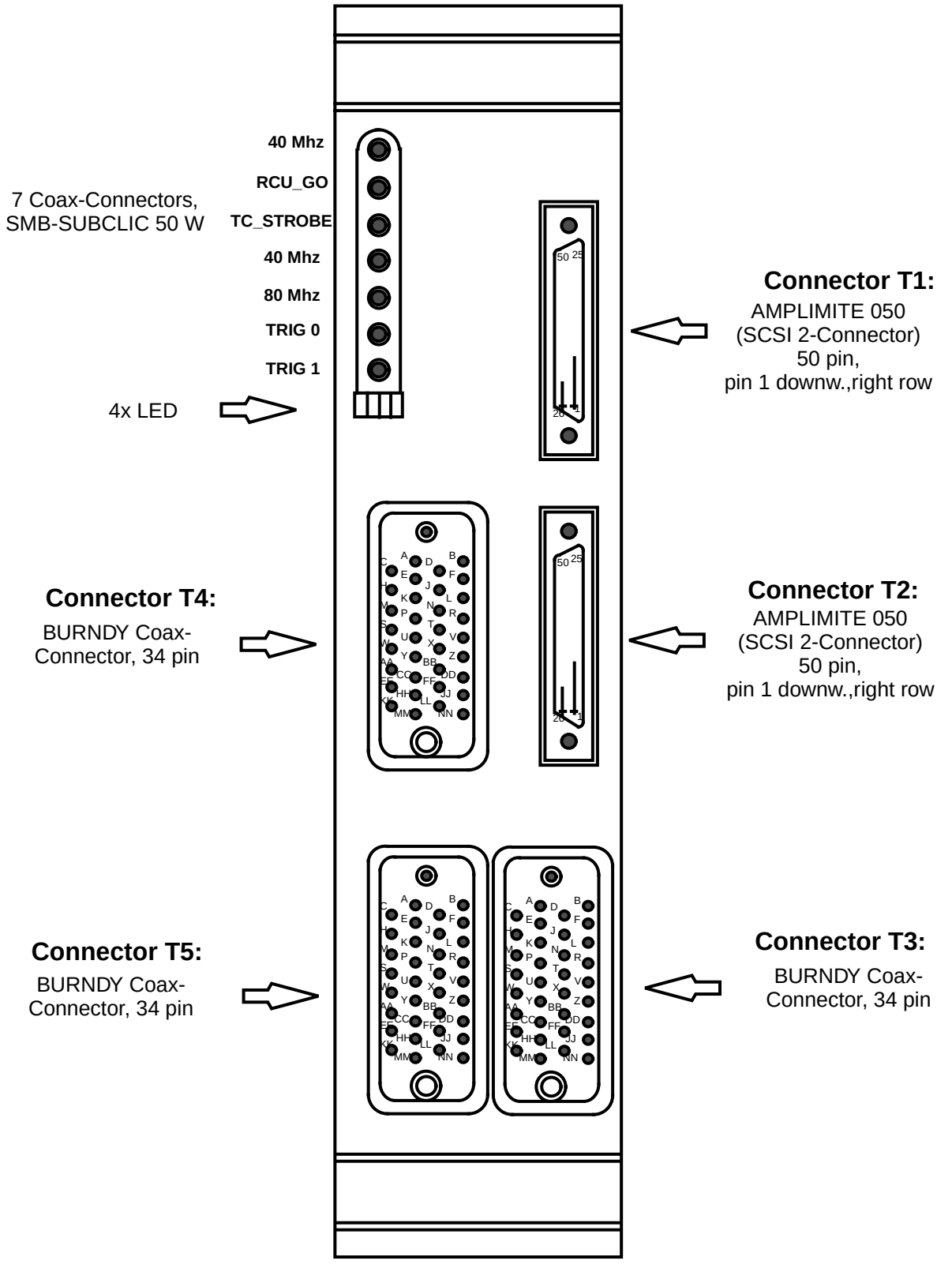


Figure 8: Front View of TCU

1. 6. 3. Connectors and Signal Allocations

All outputs switch between TTL-Levels. They are in a high impedance state after power-on and pulled up by a resistor of 4.7kΩ . The activated driver are able to drive 32 mA at ”High” and 64 mA at ”Low”.

Connector ”AMPLIMITE 050”:

The connector has got 50 contacts, arranged in one pin row numbered from 1 to 25 and one row numbered from 26 to 50. The mounting direction is pin 1 (right row) respectively pin 26 (left row) downwards (see Figure 7). It should be pointed out that, most computer manufacturers count different in their SCSI pin assignment tables (one row odd numbered and one even).
The standard cable (e.g. ”SCSI-Cable”) has always two ”side-by-side-lines” of each row twisted together. That means line 1 is twisted with line 26, line 2 with 27 and so on.

The BURNDY-Connector:

It contains 34 coaxial sockets resulting in 68 signal and ground lines to be connected. This will be managed via two 34-pin ribbon cable connectors labeled as ”Ti-A” and ”Ti-B”.
They have two 17-pin rows each, one of odd numbered pins and one of the even numbered. The odd numbered pins are allocated to the signals the even numbered are connected to ground.

The signal name ”word3_1” designates as signal source bit 1 of the output word at port 3 (see Table 8) of the Real-Time-Program-Memory.

1. 6. 3. 1. Connector T1 and T2

All signals are outputs except those remarked with note 1.

Signals with note ”sf” have ”STRAFI” as destination.

”Right row” and ”left row” have reference to the front view of TCU

Table 12: Pin Assignment of T1

Connector T1 (ST4, H2562) AMPLIMITE 050							
Pin	**Signal (right row)**	**Meaning**	**Notes**	**Pin**	**Signal (left row)**	**Meaning**	**Notes**
1	word3_0	BLNK_TR1		26	GND		
2	NMR5_0	GAIN0_TR1		27	NMR5_1	GAIN1_TR1	
3	NMR5_2	C/AB_TR1		28	GND		
4	word3_1	BLNK_TR2		29	GND		
5	NMR5_3	GAIN0_TR2		30	NMR5_4	GAIN1_TR2	
6	NMR5_5	C/AB_TR2		31	GND		
7	word3_4	BLNK_TR5		32	GND		
8	NMR5_6	C/AB_TR5		33	GND		
9	word3_5	BLNK_TR6		34	GND		
10	NMR5_7	RELAY_H		35	NMR5_8	RELAY_X	
11	TRIG3		1	36	GND		
12	NMR5_9	RELAY_Y		37	NMR5_10	RACK_ON/OFF	
13	NMR5_11			38	GND		
14	NMR5_12	RELAY_Z		39	GND		
15	NMR5_13			40	GND		
16	NMR5_14			41	GND		
17	NMR5_15			42	GND		
18	NMR6_0	STP1_DIR	sf	43	NMR6_1	LB_SEL	sf
19	NMR6_2	DCM_STRT	sf	44	GND		
20	NMR6_3	STP1_CLK	sf	45	GND		
21	NMR6_4	STP2_CLK	sf	46	GND		
22	NMR6_5	RES_STP1	sf	47	GND		
23	NMR6_6	DCM_RES	sf	48	GND		
24	NMR6_7	GO_POS	sf	49	GND		
25	TRIG2		1	50	GND		

”Direct Connection to High Power Router”

Table 13: Pin Assignment of T2

Connector T2 (ST5, H2562) AMPLIMITE 050							
Pin	**Signal (right row)**	**Meaning**	**Notes**	**Pin**	**Signal (left row)**	**Meaning**	**Notes**
1	NMR7_1	RSEL5_1		26	NMR7_3	RSEL5_3	
2	NMR7_0	RSEL5_0		27	NMR7_2	RSEL5_2	
3	NMR7_5	RSEL6_1		28	NMR7_7	RSEL6_3	
4	NMR7_4	RSEL6_0		29	NMR7_6	RSEL6_2	
5	NMR1_1	RSEL1_1		30	NMR1_3	RSEL1_3	1
6	NMR1_0	RSEL1_0		31	NMR1_2	RSEL1_2	
7	GND			32	GND		
8	NMR1_5	RSEL2_1		33	NMR1_7	RSEL2_3	
9	NMR1_4	RSEL2_0		34	NMR1_6	RSEL2_2	
10	GND			35	GND		
11	word3_0	BLNK_TR1		36	GND		
12	word3_1	BLNK_TR2		37	GND		
13	word3_2	BLNK_TR3		38	GND		
14	word3_3	BLNK_TR4		39	GND		
15	word3_4	BLNK_TR5		40	GND		
16	GND			41	GND		
17	NMR1_9	RSEL3_1		42	NMR1_11	RSEL3_3	
18	NMR1_8	RSEL3_0		43	NMR1_10	RSEL3_2	
19	word3_5	BLNK_TR6		44	GND		
20	word3_6	BLNK_TR7		45	GND		
21	word3_7	BLNK_TR8		46	GND		
22	NMR2_5	BLNK_TR9		47	GND		
23	NMR2_6	BLNK_TR10		48	GND		
24	NMR1_13	RSEL4_1		49	NMR1_15	RSEL4_3	
25	NMR1_12	RSEL4_0		50	NMR1_14	RSEL4_2	

1": 28 Way Router

1. 6. 3. 2. Connector T3

All signals are outputs.

Table 14: Pin Assignment of T3

Board Connector	GND Pin, connected to Outer Contact of T3	Signal Pin, connected to Inner Contact of T3	Signal	Meaning	Contact of T3	Destination	Notes
T3-A	2	1	NMR4_0	SELEC1 X/$\overline{H}$	A	AMX Router	
ST6	4	3	NMR4_1	BLNK_TR9	C		
H2562	6	5	NMR4_2	BLNK_TR10	E		
	8	7	NMR4_3	BLNK_TR11	H		
	10	9	NMR4_4	BLNK_TR12	K		
	12	11	NMR4_5	BLNK_TR13	M		
	14	13	NMR4_6	BLNK_TR14	P		
	16	15	NMR4_7	BLNK_TR15	S		
	18	17	NMR7_8	RSEL7_0	U	Router 3	
	20	19	NMR7_9	RSEL7_1	W	Router 3	
	22	21	NMR7_10	RSEL7_2	Y	Router 3	
	24	23	NMR7_11	RSEL7_3	AA	Router 3	
	26	25	NMR7_12	RSEL8_0	CC	Router 3	
	28	27	NMR7_13	RSEL8_1	EE	Router 3	
	30	29	NMR7_14	RSEL8_2	HH	Router 3	
	32	31	NMR7_15	RSEL8_3	KK	Router 3	
	34	33	NMR8_15	BLNK_TR15	MM		
T3-B	2	1	NMR8_0	RSEL9_0	B	Router 3	
ST7	4	3	NMR8_1	RSEL9_1	D	Router 3	
H2562	6	5	NMR8_2	RSEL9_2	F	Router 3	
	8	7	NMR8_3	RSEL9_3	J	Router 3	
	10	9	NMR8_4	SELH/L_TR1	L		
	12	11	NMR8_5	SELH/L_TR5	N		
	14	13	NMR8_6	SELH/L_TR7	R		
	16	15	NMR8_7		T		
	18	17	NMR8_8		V		
	20	19	NMR8_9	BLNK_TR9	X		
	22	21	NMR8_10	BLNK_TR10	Z		
	24	23	NMR8_11	BLNK_TR11	BB		
	26	25	NMR8_12	BLNK_TR12	DD		
	28	27	NMR8_13	BLNK_TR13	FF		
	30	29	NMR8_14	BLNK_TR14	JJ		
	32	31	TRIG2		LL		
	34	33	TRIG3		NN		

1. 6. 3. 3. Connector T4

All signals are outputs.

Table 15: Pin Assignment of T4

Board Connector	GND Pin, connected to Outer Contact of T4	Signal Pin, connected to Inner Contact of T4	Signal	Meaning	Contact of T4	Destination	Notes
T4-A	2	1	word3_16	BLNK_F1	A	ASU	
ST9	4	3	word3_24	SP_F1	C		
H5811	6	5	word3_17	BLNK_F2	E		
	8	7	word3_25	SP_F2	H		
	10	9	word3_18	BLNK_F3	K		
	12	11	word3_26	SP_F3	M		
	14	13	word3_19	BLNK_F4	P		
	16	15	word3_27	SP_F4	S		
	18	17	word3_20	BLNK_F5	U		
	20	19	word3_28	SP_F5	W		
	22	21	word3_21	BLNK_F6	Y		
	24	23	word3_29	SP_F6	AA		
	26	25	word3_22	BLNK_F7	CC		
	28	27	word3_30	SP_F7	EE		
	30	29	word3_23	BLNK_F8	HH		
	32	31	word3_31	SP_F8	KK		
	34	33	NMR2_10	TUNE_O/F	MM	SE451	
T4-B	2	1	NMR2_11	OBS_F1	B	FCU	
ST11	4	3	word3_24	SP_F1a	D	or	
H5811	6	5	NMR2_12	OBS_F2	F	SE451	
	8	7	word3_25	SP_F2a	J	or	
	10	9	NMR2_13	OBS_F3	L	LOT	
	12	11	word3_26	SP_F3a	N		
	14	13	NMR2_14	OBS_F4	R		
	16	15	word3_27	SP_F4a	T		
	18	17	word3_8	SP_PA1	V	HPPR	
	20	19	word3_9	SP_PA2	X		
	22	21	word3_10	SP_PA3	Z		
	24	23	word3_11	SP_PA4	BB		
	26	25	word3_12	SP_PA5	DD		
	28	27	word3_13	SP_PA6	FF		
	30	29	word3_14	SP_PA7	JJ		
	32	31	word3_15	SP_PA8	LL		
	34	33	NMR2_7		NN	RCP PA SWITCH	

1. 6. 3. 4. Connector T5

All signals are outputs.

Table 16: Pin Assignment of T5

Board Connector	GND Pin, connected to Outer Contact of T5	Signal Pin, connected to Inner Contact of T5	Signal	Meaning	Contact of T5	Destination	Notes
T5-A	2	1	word3_0	BLNK_TR1	A		
ST8	4	3	NMR3__0	CALIB_TR1	C		
H5811	6	5	NMR3_8	REGUL_TR1	E		
	8	7	NMR3_12	DROOP_TR1	H		
	10	9	word3_1	BLNK_TR2	K		
	12	11	NMR3_1	CALIB_TR2	M		
	14	13	NMR3_9	REGUL_TR2	P		
	16	15	NMR3_13	DROOP_TR2	S		
	18	17	word3_2	BLNK_TR3	U		
	20	19	NMR3_2	CALIB_TR3	W		
	22	21	word3_3	BLNK_TR4	Y		
	24	23	NMR3_3	CALIB_TR4	AA		
	26	25	word3_4	BLNK_TR5	CC		
	28	27	NMR3_4	CALIB_TR5	EE		
	30	29	NMR3_10	REGUL_TR5	HH		
	32	31	NMR3_14	DROOP_TR5	KK		
	34	33	word3_5	BLNK_TR6	MM		
T5-B	2	1	NMR3_5	CALIB_TR6	B		
ST9	4	3	NMR3_11	REGUL_TR6	D		
H2562	6	5	NMR3_15	DROOP_TR6	F		
	8	7	word3_6	BLNK_TR7	J		
	10	9	NMR3_6	CALIB_TR7	L		
	12	11	word3_7	BLNK_TR8	N		
	14	13	NMR3_7	CALIB_TR8	R		
	16	15	NMR2_0	HOLD LOCK	T	BSMS	
	18	17	NMR2_1	HOMOSPOIL	V	BSMS	
	20	19	NMR2_2	SELH_H/F	X	BLARH	
	22	21	NMR2_3	SELX_X/F	Z	BLARH	
	24	23	NMR2_4	Z0COMP_EN	BB	BSMS	
	26	25	NMR2_8	FXA	DD	BPII	
	28	27	NMR2_9	FXB	FF	BPII	
	30	29	word4_29	BLNK_GDX	JJ		
	32	31	word4_30	BLNK_GDY	LL		
	34	33	word4_31	BLNK_GDZ	NN		

1. 6. 4. Power Requirements

The TCU1 needs only 5 Volts as VCC and requires about 7 Ampere.

	Part-No.	+5 V	+12 V	-12 V	+5 V analog J3: C8	-5 V analog J3: C1,...,C5
TCU1	H5811/ H5812 H2562	6,3 A + 10 A	0	0	0	0

Remarks:

- The additional TCU power reqirement of 10 A reflect the worst case in which all TCU outputs are terminated with 50 Ohms to ground.

2. Manufacturing Informations

2. 1. Manufacturing Data

Table 17: Table of Assembly Groups

Amount	Title	Function	Part-Nr.
1	TCU-MAIN 4k	Assembled PCB	H5811
1	TCU-MAIN 8k	Assembled PCB	H5812
1	TCU-MAIN	Layout	H3P1860E
1	TCU-MAIN	Plain PCB	H5810
1	TCU-MAIN PAL set		H5840
1	TCU-EXT	Assembled PCB	H2562
1	TCU-EXT	Layout	H3P2020B
1	TCU-EXT	Plain PCB	H2563
1	TCU-EXT PAL set, EC<20		H3362
1	TCU-EXT PAL set, EC≥20		H3481
1	Front Panel		HZ2434
1	Front Panel Assembly Set		18878
1	Front Panel Ident		HZ2435
1	Cable 34P70	Interconnection H2558/H2562	HZ2934
1	Cable 50P80	Interconnection H2558/H2562	HZ2933
3	Burndy Connection	PCB to Burndy Adapter	H5563
1	TCU Test Board	For production and tests	H5663

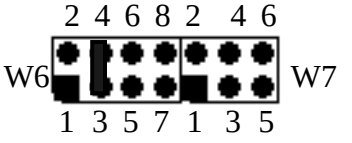
2. 2. Introduction Status

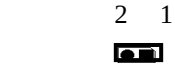
2. 2. 1. Configuration

Jumper W4 on H5811
not set

Jumper W5 on H5811
Interrupt Vector 0x2

Jumper W6 and W7 on H5811
Interrupt request on IRQ2

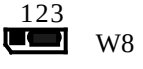




Jumper W1 on H5811

1-2	Address Size
not SET	4K WORDS a 96 bits (standard) H5811
SET	8K WORDS a 96 bits (optional if IDT7025 Memorys are inserted) H5812

Jumper W8 on H5811



W8	Configuration
1-2	External 80MHz TTL Signalmust be provided
2-3	Internal 80Mhz Translater (Sinus to TTL) is used (default).

Jumper W3 and W4 on H2562
TRIG3 at Connector T3, contact NN
TRIG2 at Connector T1, pin 25

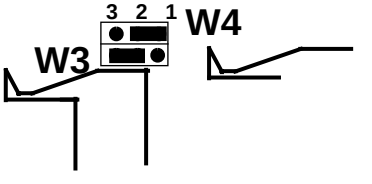


Figure 9: Introduced jumper setting

2. 2. 2. Modifications of the introduced layout

2. 2. 3. Service Informations

2. 3. History of Modifications

2. 3. 1. TCU main board

EC No.	Date	Part Number	Description of Bugs, Changes and Modifications	Ser.No.	New EC-Level
2096	12.12.94	H5811/ H5812	Includes all former modificcations of the TCU main board H2558 Provides 4k words instead of the former 512 adds the identification EEPROM	20	00
2158	18.5.95	H5812	Includes all former modificcations of the TCU main board H2558 Provides 8k words instead of the former 512 adds the identification EEPROM		00
2195	4.9.95	H5811/ H5812	Improvement of the AQ strobe signal to avoid write failures on the FCU		01

2. 3. 2. TCU extension board

EC No.	Date	Part Number	Description of Bugs, Changes and Modifications	Ser.No.	New EC-Level
1757	21.7.93	H2562	Introduction of TCU Extension Board as Layout Version H3P2020A	0010	EC01
1883	25.2.94	H2562	Modification for DPX and High Power Units: The NMR2, NMR5 andNMR6 drivers are prevented from turning off by Reset after measurement	0131	EC02
1918	19.4.94	H2562	Spike on output enable		EC03
1957	13.6.94	H2562	Introduction of layout H3P2020B, including all former modifications and adding more output signals needed in DMX and DRX This layout needs the PAL set H3481 instead of H3362		EC20

3. Testing

3. 1. Testprograms of AQX devices

3. 1. 1. Usage

3. 1. 1. 1. Where to use the testprograms

On AMX spectrometers (`amx, arx, asx`**)**

`aqtest`	Tests the AQI interface to Aspect3000, Aspect 30001
`gctest`	Tests the Gradient Controller
`gcutest`	Tests the Gradient Controller

On spectrometers of the DMX series (`dmx, drx, dsx`**)**

`fcutest`	FCU test (frequency control unit)
`tcutest`	TCU test (timing control unit)
`gcutest`	GCU test (gradient control unit)
`rcutest`	RCU test (receiver control unit)

On all spectrometers

`memtest`	Memory test. This test runs only stand alone
`siotest`	Tests the serial interfaces on the CCU and the SIO board. This test runs only under UNIX
Note:	If the board to be tested is not present, the test will print an error message and exit. The `gcutest` decides by itself which hardware is available and has to be tested.

3. 1. 1. 2. How to start a test program

`device` has to be specified as a choice out of the following device names `fcu, tcu, gcu, rcu, gc, aq, mem, sio`
The test programs have to be started on the AQX CCU of the spectrometer. Otherwise it warns you and exits. To log in at the spectrometer enter

```
telnet spect
root
```

Start a test using UNIX with

```
cd /u/systest/device
./devicetest
```

During execution the *`device`*`.firm` is loaded to the board or device under test and executed by the local processor. To run a test stand alone (without UNIX) shutdown the CCU with

```
/etc/init 5
```

On the console which is connected to the CCU enter

```
boot -f bfs()/usr/diskless/clients/spect/root
                    \/u/systest/device/devicetestsa
```

Exception: `memtest` is started standalone without the extension `sa`
`siotest` cannot be started standalone

Normally you should enter `auto`, when the testprogram prompts you for an input

3. 1. 1. 3. Special files used by the test programs

To use the driver and the full functionality of the test programs it is necessary that the following special files of each device had been created and are avalable:

File name	major#
`/dev/AQI`	55
`/dev/gc`	56
`/dev/rcu`	59
`/dev/fcu`	60
`/dev/tcu`	51

Such a special file is created with `mknod`, for example:

```
mknod /dev/AQI c 55 0
```

The major number can be checked with :

```
ls -l /dev/aq
crw -rw -rw 1 root bin 55 0 Jun14 1993 /dev/aq
```

3. 1. 1. 4. Main features of the test programs

1. Get program version

Start the test program with:

```
device test -v
```

The test will print its version number and exits.

Note: This paper applies to program version 950901.1 and the newer ones

2. auto-command

Start the test and enter the command `auto`. All tests are executed automatically. Errors found are printed on your terminal and listed in the file

```
/u/systest/device/errorfile
```

This error file is rewritten each time you exit and restart the test program.

3. help-command

When you enter `h`, you will get a list of all available commands with a short description.

4. protocol

When you enter the command `prot` for the first time, all subsequent input and output is written into a protocol file until you enter `prot` for the second time. You can write several protocol files while the test is running. The name is to your choice.

5. command file

Instead of entering commands directly to the test, you can put them into a file, then start with:

```
devicetest -c cmd
```

where `cmd` is the name of that file. The test program will execute the commands and if the last command is not `quit` or `q` it will continue with reading more commands from the keyboard.

6. shell

With the command `sh` you get a shell without leaving the test program. You can exit that shell and return to the test program by entering `exit` or `crtl-d`. This feature does not work in the stand alone tests.

7. terminate the test program

If you leave the test program by the commands `q` or `quit` , the program resets the i960 on this board (if there is one) and restores registers that may have been modified during the test. If you leave with the command `l` , nothing is changed or reset.

8. loops

The most tests can be started in a loop. See the section titled ‘‘parameter setting’’.

9. registers

The names of on-board registers can be found with the command `rname`. An information for each register is given with the command `rinfo`.

10. debug print’s

The accesses by the CPU or i960 to memory can be made visible by the command `sw` (switch). The second time `sw` is used, it makes the accesses invisible.

11. DELETE

Any command can be interrupted with `DELETE`. This feature may be delayed in stand alone programs.

Note: If the i960 is just executing a command, only the program running on the main CPU notices your `DELETE`. Before the i960 can execute a new command, you must reset it.

12. execution of a command

At first the processor will be started, if the command has to be executed on the i960. Then the user is asked for the necessary parameters. If necessary, they are transferred into i960-memory. During an execution of a command by the i960, the CPU polls the i960-memory to check for completion. All communication is done via the mail box located at offset 0x3600 in the i960-memory.

For RCU and AQ, the physical page addresses for the VME-memory to be used with a DMA start at offset 0x4000 in i960-memory.

13. Load (and execute) another program

Use the command `load`, then enter the name of the program to be loaded to the i960-memory. All subsequent commands for the i960 will load and use this program. You can directly start it with the command `run`.

14. List these manual pages

Enter the command

```
man
```

to the test and select amanual page.
It will be listed on the screen and can be saved in a file.

3. 1. 1. 5. Parameter setting

Defaults

Each value or string of the console print out written in brackets [] is a default setting. If you enter `RETURN`, this default value is kept and not modified. Use `gpar` to get the values of all available paramters. Use `spar` to set them (or part of them).

`start/lstart`	If there is an i960 on the board, `start` is the VME-address used by the CPU. `lstart` is the corresponding local start address used by the i960. If you enter the test start address by `spar`, you must always use the local address.
`number of loops`	Affects memory tests, register tests, read and write memory, read and write registers, and board specific tests.
Test `mode`	`mode` is for memory tests started on the i960. `f`: read/write words forward `r`: read/write words in reverse order `q`: read/write quad words forward `s`: read/write quad words in reverse order
`continue on error`	If this parameter is set and an error is found, the tests prints out the error message and continues. The total error count is printed out when the whole test finished. If this parameter is not set, the test terminates after the first error has been found.
`print on mem-access`	Use the command `sw` to switch on/off printing on memory access.

To switch on for CPU-memory accesses enter:

```
sw
c
```

To switch on for i960-memory accesses enter:

```
sw
l
```

3. 1. 1. 6. Overview of tests

Device memory test executed by the CCU

(a) `tim`	test instruction memory
(b) `tdm`	test data memory (if present)
(c) `tcm`	test combox memory (if present)
(d) `tms`	test memory and set param's

(e) `tmv` test memory with value

(f) `tmiv` test memory with incr. value

(a), (b) and (c) test the whole memory region present. (d), (e) and (f) use the parameters for start address and size which have to be set before with the command ‘‘`spar`’’.

(e) tests with one constant value set by `spar`,

(f) increments this value during the test.

(a) - (d) test in subsequent passes with the following values :

1.Pass: 0
2.Pass: 1, 2, 4, 0x10, ..., 0x80000000
3.Pass: value == address
4.Pass: value incremented by 0x10001
5.Pass: -1
6.Pass: 0
7.Pass: 0xaaaaaaaa and 0x55555555 alternativly

Device memory test executed by the local processor (i960)

These tests are not applicable on the FCU’s.

(a) `timl` test instruction memory local

(b) `tdml` test data memory local

(c) `tcml` test combox memory local

(d) `tmsl` test mem, set param’s local

(e) `tmvl` test memory with value local

These commands operate in the same manner, except that the i960 instead of the CCU accesses the device memory.

Register tests

(a) `tr` The registers are accessed by the CPU

(b) `trl` The registers are accessed by the i960.

Parameters:

`Name`: Select a register name or enter `all`.
If you enter `all`, all registers are tested for which this is possible.

`Value`: Select a number in hexadecimal, ‘‘`bits`’’ or ‘‘`all`’’.
If you enter ‘‘`bits`’’, the register will be tested with the values 1, 2, 4, 0x10, ...
If you enter ‘‘`all`’’, the register will be tested with the values 0, 1, 2, 3, 4, ...

Interrupt tests

(a) `int` Interrupt from i960 to CPU

(b) `intl` Interrupt(s) from CPU to i960

Basic tests for the i960

If the `auto` command in any test running on the local i960 does not work properly check the following basic functions:

1. Reset the i960

   `res`

2. Test if the device memory is accessible

   `tim`

3. Load the test program

   `load` *`device`*`test`

4. Start the i960 without any command

   `run`

5. Run a command on the i960

   `hello` (prints ”hello” on the screen)

3. 1. 1. 7. Special TCU test features

1. Wait operation test

   `wait` The CPU fills 4-PORT RAM with the instructions `WAIT`, `CLEAR WAIT, NMI` and tests them.

2. Duration test

   `dur`

3. Loop counter test

   `lpcnt` The i960 checks loop counter, decrement counter and unconditional loop back.

4. Address generator test

   `tagen`
   1. Interrupt INT0 Test
   2. Pre-register Test
   3. Address generator Bit Test, value = 0,1,2,4,8...0x100
   3. Address generator value Test, 0 <= value <=0x1ff
   4. Address generator Test with ’Astep’ register

5. Blanking register test

   `nmr`

6. Create RCU GO pulse

```
rcugo
```

ACQ bus test between TCU-FCU

7. ACQ bus data line test

`acq`	The CPU writes 4-PORT RAM and reads ACQ Bus from FCU register

8. ACQ - FCU Pointer test

   The i960 fills 4-Port Ram with FCU board number 0-7, FCU Pointer 0-255 and data. The CPU reads FCU-Memory pointer and compares them.

   The commands `aqincr`, `aqstep` and `aqreload` consist of the two components: command-name and pointer-number.

`aqfcu`	Increment, reload and step fcu instructions test. (from acq bus)
`aqincr`*nnn*	Fcu increment test (from acq bus)
`aqstep`*nnn*	Fcu step test (from acq bus)
`aqreload`*nnn*	Fcu reload test (from acq bus)
Note:	*nnn* is the FCU pointer number

   Examples:

```
aqincr0
aqstep1
aqreload255
```

ACQ bus test between TCU-GCU

1. Test of some GCU functions initiated by the TCU via the AQ bus

`gcu`	The CPU strts each of these tests below
`ng`	NG pulse test
`rtf`	gcu real-time AQ data test
`aqctl`	AQY Bus - Data and Control flags test
`aqdat`	AQY Bus Data flags test
`aqst`	AQY Bus Interrupt and Status register test
`aqf0`	AQY Bus Control Flag0 test
`aqf1`	AQY Bus Control Flag1 test
`aqint`	AQY Interrupt Flag test

3. 1. 1. 8. Special FCU test features

`conf`	Fcu board and memory configuration test
`fcun`	Set fcu board number
`durg`	duration test
`dds`	Test of DDS interface

The commands `lddp`, `ldp`, `incr`, `step` and `reload` consist of the two components: command-name and pointer-number

`lddp`*nnn*	load data via pointer test
`ldp` *nnn*	load pointer test
`incr`*nnn*	Fcu increment test
`step`*nnn*	Fcu step test
`reload` *nnn*	Fcu reload test

Examples:

```
lddp0
ldp12
incr255
```