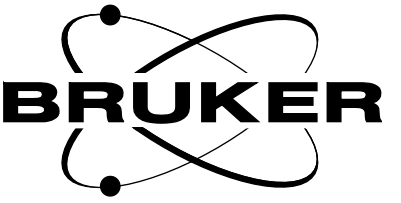


AQX

GCU
Technical Manual

Version 002

BRUKER

The information in this manual may be altered without notice.

This manual was written by

T. Eckert and Dr. J. M. Rommel

Rheinstetten, Germany
Updated for BASH 2.0 by U. Roos - December 1996
P/N: Z31343
DWG-Nr: 1052 002

1 : AQX Gradient Control Unit

Eckert/Rommel

Contents

Figures

Tables

1. Technical Description

1. 1. General Information

The Gradient Control Unit (GCU) is an intelligent VME-BUS Slave controller Board with the ability to generate interrupts on the VME-BUS. A essential part of the GCU is the I80960 microprocessor. The basic function of the Gradient Control Unit is to calculate on-line the X,Y and Z Gradient values and to control the timing of the D/A conversion. This firmware program is load by the CPU via the VME-Bus. A analog preamphasis Unit connected to the Gradient D/A converter outputs finally controls the Spectrometer Gradient Field. Because of digital noise, the D/A converters board is located in a separate unit. A cable is used to interface the GCU with the D/A converter board (50 way max. length should be 4 meters). The Timing Control Unit (TCU) witch generates the Master timing, requests the GCU via the AQ-Bus to calculate a new data set for X,Y and Z Gradient. Further the TCU provides the GCU with the 40 MHZ reference clock (via coax) for precise Gradient switching .

1. 2. Features

- 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 0 wait state and pipeline operation
- 8KB additional Data Ram with local Bus for communication between VME Master and Grad. Controller.
- special VME-Bus register set (interrupt,status, and configuration)
- 16 Data 8 Add/control Bit Interface to the Grad. D/A converter unit
- 32 Bit Timer with 25ns resolution to control the Gradient timing when producing a Ramp or other special Gradient waveforms.

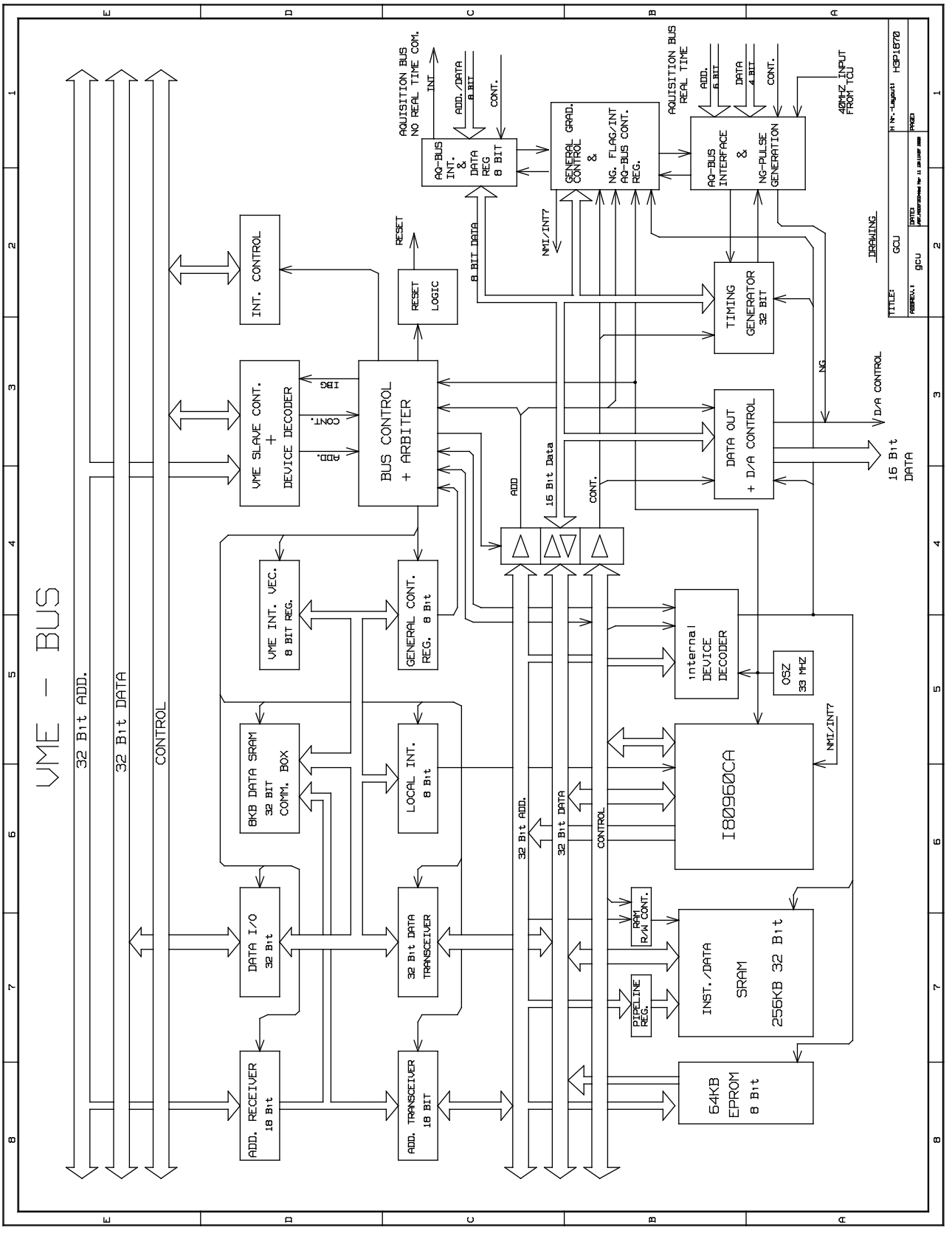
1. 3. Architecture

1. 3. 1. Block Diagram

The Gradient Controller can be distinguished in six groups.

- 1. VME - Bus Interface
- 2. I960 Processor Kernel
- 3. Arbiter between I960 and VME-Bus
- 4. Reset Control
- 5. Grad. Timer and AQ-Bus Interface
- 6. Grad. D/A converter Interface

Figure 1: GCU Block Diagram



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. The registers in the D/A converter board,witch are connected to the Gradient control unit via a 50 way cable are also testable from VME-Bus. For communication and data exchange between a VME-Bus Master and the I960 there is an 8KB data ram and the VME registers set, located on an separate internal Bus.. The advantage is ,that the VME-Master can access the local VME register or data RAM without interruption of the I960 program execution.

I960 Processor Kernel

An important part of the Gradient Control Unit is the Intel I960 Microprocessor operating at 33MHZ with build around 256KB fast (20ns) 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 GCU 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. The NMI is used to detect any real-time actions on the AQ-Bus such as the occurrence of an NG pulse. The INT7 is used to signal any timing error condition on the AQ-Bus interface meanwhile INT0-6 can be used by the VME Master to interrupt the I960.

Arbiter between I960 and VME-Bus

Any access from the VME Bus or I960 to the resources on the GCU 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 GCU.

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 Gradient Register Set (Timer,control reg,DAC reg.). The I960 signals the arbiter with the HOLDACK line that it has released the intern Bus and the VME-Bus access is granted.

Reset Control

The reset logic is used to initialize the Gradient Controller 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 GCU can be tested via VME -Bus.

Gradient Timer and AQ-Bus Interface

The Grad. Timer is a 32 bit programmable counter used to generate a constant periodical time base for the GCU to produce special Gradient shapes such as a Ramp or other waveforms. The Timer has a resolution of 25ns and a min. cycle time of 125ns. The Timing Control Unit (TCU) provides the Timer with the 40MHz clock. If enabled by the GCU, an device code (i.e. NG) given by the TCU over the real-time AQ-Bus starts the Timer. The occurence of the NG-pulse loads the D/A converter from the X,Y and Z register board with a new data set and the I960 is interrupted with the NG-FLAG set. Three sources can generate an NG-pulse :

- 1. The TCU over the real-time AQ-Bus
- 2. The Grad. Timer if enabled
- 3. The VME-Master (only for testing)

The AQ-Bus Interface consist of two parts, a real-time and a non real-time Bus. The real-time interface consist of an 8 bit address, 4 bit data Bus and 2 control signals. All GCU actions witch should be synchronic with the TCU timing are controlled via the real-time AQ-Bus (i.e. NG-pulse,dummy scan). The no real-time AQ-Bus is used for communication between the TCU and the GCU. It consist of an 8 bit add/data Bus, 4 control signals and an interrupt signal. The interrupt is used by the GCU to signal the TCU any error condition.

Grad. D/A converter Interface

The Interface between the GCU and the D/A converter unit consist of a 16 bit data Bus, 4 address and 4 control signals . The data for the X,Y and Z Gradient are transferred to the corresponding register, located on the D/A converter board, via a 50 way cable. An NG-Pulse from the GCU enables the data set to the D/A converter inputs whereby the respective analog signal is set. The registers on the D/A converter board are readable by the GCU witch is important for testing. The analog outputs (X,Y,Z) of the D/A converter are connected to the preamphasis unit located in the same box.

1. 4. i960-Processor Environment

1. 4. 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. 4. 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 00FFINTERNAL DATA RAM (USER WRITE PROTECTED)

0000 0100 - 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. 4. 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 GCU 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, Communication Box register, Control register , and local interrupt register. Region C is used for the Gradient Data Register SET (i.e. X,Y and Z Data reg.) . Region D is used for the external 8KB Data Ram (Gradient Data parameter) . Region E is used for the Gradient Control Registers (i.e Timer ,AQ-Bus Flags,NG-Flags,NMI) whereas Region F is used for the EPROM boot sequence and for monitor code. The memory table entry is shown below:

1. 4. 4. Memory Region Table (256 Mbytes step)

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 CGRAD_DATA_REGISTER
D000 0000 - DFFF FFFFREGION DEXTERNAL 8KB DATA RAM
E000 0000 - EFFF FFFFREGION EGRAD. CONTROL_REG.
F000 0000 - FFFF FFFFREGION FEPROM_CONF

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
Shown below is the Memory Region configuration for the GCU.

Region 0-7 Inst. SRAM configuration table (region 0-7) with 20 ns access time

Data-Bus Width	32 Bit
NRAD	0 wait states
NRDD	0 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 Grad.- Data REG. Device Code

Data-Bus Width 16 Bit
NRAD 6 (3) wait states
NRDD 3 wait states
NWAD 4 (3) wait states
NWDD 3 wait states
NXDA 0 wait states
Pipelining NO
External Ready NO
Burst NO

Region D configuration table Data/Parm SRAM with 25 ns access time

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 YES

Region E configuration table Grad.- Control REG. Device Code

Data-Bus Width	16 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	NO

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. 5. Detailed Register Set Description

The Register set can be general distinguished in two groups:

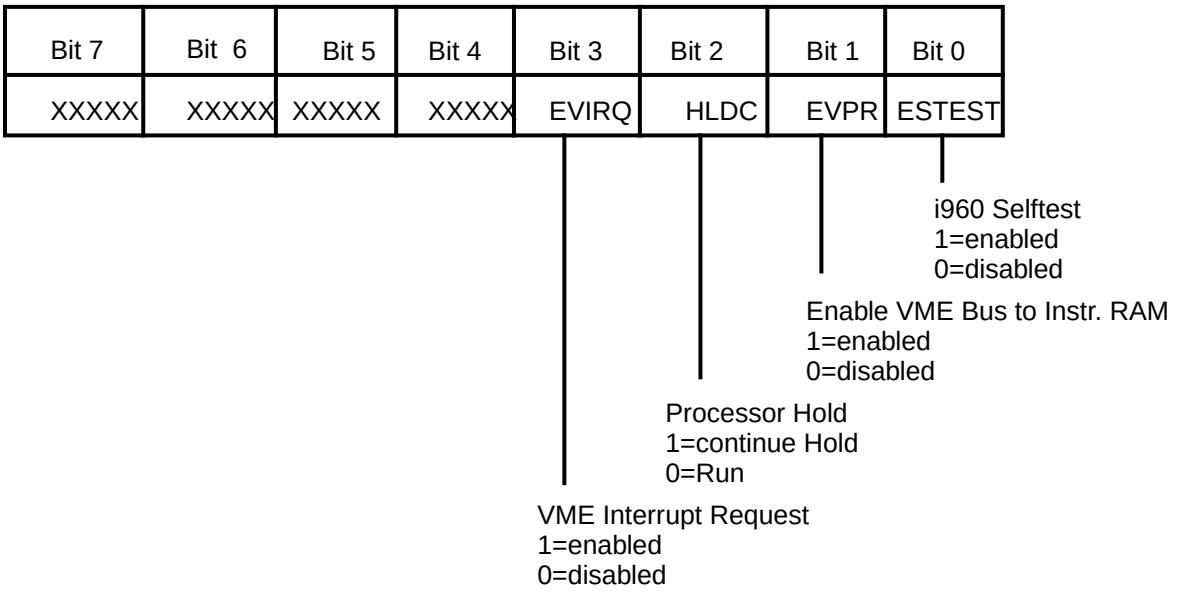
- 1. VME-Bus oriented Register
- 2. Gradients control/data oriented Register

Nearly all Register can be read and written from a VME_Bus Master and from the I960 processor (useful for test and debugging). From the VME-Bus all register are referenced as longword and from the I960 the register are addressed as long, short or byte. A detailed bit description of the different registers is given below.

1. 5. 1. VME Control Register

The VME 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: VME 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. This bit should only be set to 1 for test and debugging and not in an application program because the I960 execution time will dramatically increased ,when an VME-Bus cycle take place between the I960 instruction cycles .

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.

XXXX : The bits 4 - 7 are not used.

1. 5. 2. Local I960 Interrupt Register

This register is used to set a dedicated interrupt on the I960 processor. The Bits 0 - 6 are directly connected to the I960 interrupt pins XINT0 - XINT6. They are activ if set to LOW. Bit 7 is not used for interrupt, therefor it can be used for hardware debugging. Setting this register to FF all interrupts will be inactive. Normally the Host will set an interrupt bit to tell the I960 that a break condition has occurred. The I960 should clear this bit before it returns from the interrupt service routine.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
XXXXX	XINT6	XINT5	XINT4	XINT3	XINT2	XINT1	XINT0

Figure 3: Local Interrupt Register

1. 5. 3. 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 Gradient controller 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 4: VME Interrupt Vector Register

1. 5. 4. Communication Box Register

This is an 32 bit register which is physically located in the dual port data Ram and is used as a communication path between the VME-Bus Host and the I960 processor.

1. 5. 5. Gradient Control Register

There are several single bit registers (i.e. RAMP,DP,ENGI,TM,AQI) used to control the Grad. Timer operation as well as the detection and generation of an NG-Pulse. The control bits can be separately read and written (see Figure 4.3.3). The default value of all these register after power up is 0. A detailed bit subscription is given below.

Ramp control Register (RAMP)

This one bit register is used to alert the Grad. Timer. If set to 1 the Grad. Timer is started when an NG-pulse is detected. The Timer will then generate periodical NG.-Pulses and request the I960, with NMI or NG._Flag set, to issue new values for x,y, and z Gradient. In this way a ramp or any other waveform of the Gradient signal can be produced with constant time intervals. The Timer should be stopped (Ramp Bit set to 0) before a further NG.- Pulse is generated by the TCU. Otherwise an error interrupt will occur.

Digital Preamphasis control Register (DP)

If the GCU should calculate the preamphasis, needed to adjust the gradient current to the respective gradient coil, instead using the analog preamphasis control, the DP register should be set to 1. This alert the Timer to start on the next NG-Pulse . The Timer will then generate a periodical clock (GCLK) in addition to the NG.-Pulse from the TCU. On every falling edge of the GCLK signal the actual preamphasis data set for the X,Y,Z and the B0 correction, stored in the D/A register, is transferred to the D/A converters. The I960 should then calculate and load the D/A register with a new data set before the next edge of GCLK occurs. An NG.-Pulse ,generated by the TCU, signals the I960 by an NMI, that it has to issue a new data of nominal values for the X,Y,Z Gradient. If the DP register is set to low digital preamphasis is disabled and the timer is stopped.

Enable Grad. Interrupt (ENGI)

The Interrupt (I960 NMI) generated by an NG.-Pulse can be enabled if the ENGI register is set to high or disabled if set to low. In some time critical applications it can be better to poll the NG.-Flag instead to enter in the NMI service routine. For this reason it is possible to disable the NMI.

Test Mode Register (TM)

The TM register is provided to test some hardware register and control logic on the GCU board, without connecting the TCU or AQ-Bus. This can be useful to check the interface to the TCU is working correct. If the TM register is set to 1 the Grad. Timer is provided with the onboard 33MHZ system clock instead of the 40MHZ input from the TCU. Further the real time AQ-Bus flags (RAQD0-3) can than be set and cleared by the I960 and the NG.-Pulse, normally generated by the TCU, can be simulated by the I960 via device codes. Because the TM register is only interesting for tests it should not be used on normal operation. On default, after reset or power up, the Test Mode is off (TM = Low).

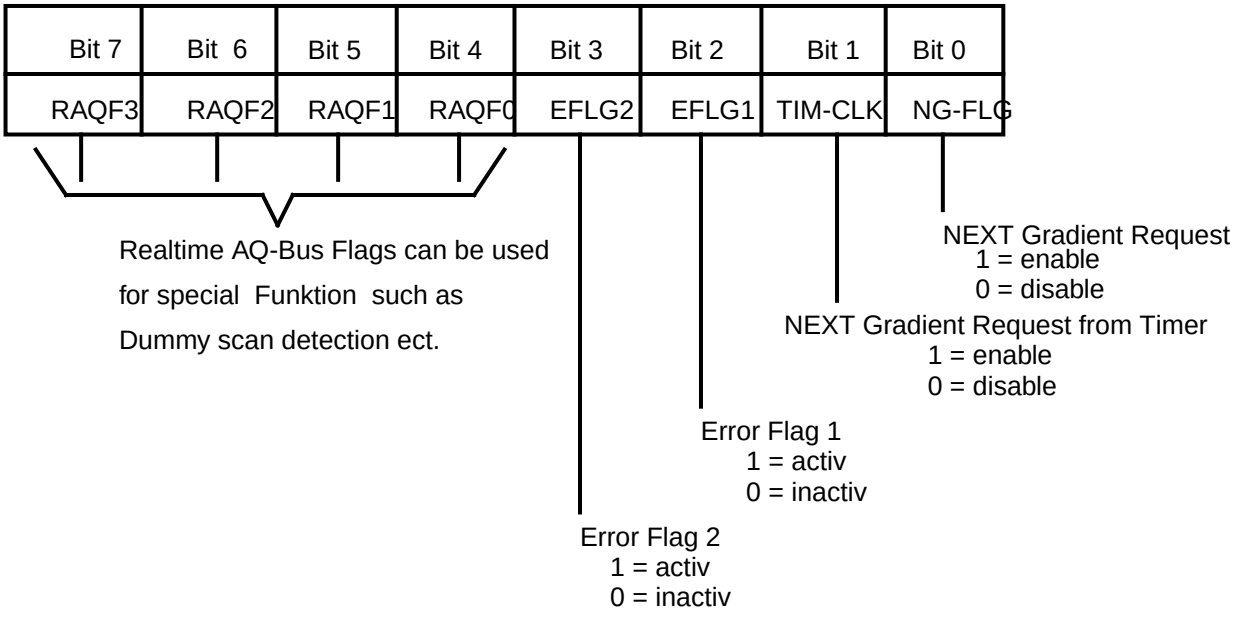
Acquisition Interrupt Register (AQI)

An interrupt (AQI) to the TCU over the AQ-Bus is provided to inform the TCU about an error occurred on the GCU. The TCU should read the AQ- Status register on the GCU when detecting AQI active (LOW) to obtain the error condition. The AQI is cleared (HIGH) when the TCU reads the AQ-Status register.

1. 5. 6. Gradient Status Register

This 8 bit register is used to monitor the commands from the TCU and the error conditions which arise when the timing of the gradient data transfer is not synchronic with the AQ-Bus operation. Its a special register where each bit can be separately written on a different address and all bits are readable in a single cycle.

Figure 5: Gradient Status Register



NG.-FLG:
If set to 1 the TCU requests the I960 for a new nominal set of X,Y and Z Gradient values. The I960 should clear (0) this bit when it has transferred the new data set.

TIM-CLK:
When set to 1 it signals the I960 that the timer has count to zero and the I960 should issue the next Grad. values witch are used to generate a ramp or any other function of the Gradient signal. If preamphasis mode (DP=1) is enabled the user program should issue the next data set for preamphasis control. This bit should be cleared by the I960 if it has transferred the new values.

EFLG1:
If set to 1 it signals the I960 that a NG-Pulse was generated by the Timing Control Unit and no new data set was available. For example if the time to calculate a new data set by the I960 processor is too short. Typically when this situation occurs the error interrupt XINT7 is active and the I960 can monitor this flag to obtain the error condition.

The bit should be cleared by the I960 after detection and initialized at the begin of a measure.

EFLG2:
A ”1” signals the user program that an NG.-Puls by the TCU was generated and the I960 has not finished to generate the ramp function. Typically when this situation occurs the error interrupt XINT7 is active and the I960 can monitor this flag to obtain the error condition. The bit should be cleared by the I960 after detection and initialized at the begin of a measure

RAQF0-3:
These bits can be set by the TCU over the real-time AQ-Bus. The GCU should read these bits to detect whether a dummy scan or any other real-time depended operation is needed to be executed by the GCU.

1. 5. 7. I960 NMI and XINT7

The NMI will be set by an NG.-Puls from the TCU or from the timer clock and should be cleared (set to 1) by the I960 before it returns from the interrupt service routine. The Interrupt 'XINT7' will be activated (0) if one of the above error conditions, where FLG1 or FLG2 are set, occurs. Specially for test or debugging the NMI/ XINT7 can be set and cleared by the Host CPU or the I960 itself .

1. 5. 8. AQY-Bus INT./Status Register

This 8 bit write only register is used to store the interrupt status information for the TCU. The TCU should read this register if it detects that the "AQI7" is active to obtain detailed information about the interrupt condition. The "AQI7" will be cleared after the TCU has read the Int/Status register.

1. 5. 9. AQY-Bus Data/Control Register

Via the AQY-Data register the TCU can send the GCU an 8 bit data word (AQY7-0) for general use (no real-time) such as special triggering methods or conditional gradient program execution. A software handshake mechanisms is used to insure that the GCU read valid data written by the TCU. The handshake is as follows: First the TCU write into the AQY- data register and then it sets the control flag "ACF0" equal LOW. The GCU, seeing the "ACF0" set, reads then the AQY-data register and clears (set to HIGH) the "ACF0". Further the TCU can send a 4 bit control word (AQF3-0) to the GCU over the AQY-Bus. The same handshake mechanisms as above , except the "ACF1" handshake flag, is used to signal the GCU that the "AQF3-0" are valid. The GCU reads the AQY-Data and control register in one cycle as a 16 bit word. The corresponding bit locations are as follows:

Figure 6: Acquisition Data/Control Register

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
ACF 1	ACF 0	XXX	XXX	AQF 3	AQF 2	AQF 1	AQF 0	AQY 7	AQY 6	AQY 5	AQY 4	AQY 3	AQY 2	AQY 1	AQY 0

Note: xxx not used

1. 5. 10. Gradient Data Register X,Y and Z

The X,Y and Z Gradient Data Register are 16 bit wide and are located in the D/A converter unit witch is connected to the Gradient Controller. These read- /writeable registers are used to store the actual Gradient value for the next D/A conversion.

The B0,B1 and B2 correction data register are each 16 bit register located in the D/A converter unit and are used to store the current values for the preamphasis correction, related to the data in the X,Y and Z Gradient register.

The PX,PY,PZ and PB0,PB1,PB2 are the equivalent register to those above. These registers are needed to preload a data set (X,Y,Z) in the case of the occurrence of a NG-Puls while the current preamphasis is not being finished.

A special read back register (see figure 4.3.2) is provided to test the Gradient data output operation without the connection of the external D/A converter unit (useful for test and debugging).

1. 5. 11. Gradient Timing Generator

This is an 32 bit counter witch generates the correct timing for the D/A conversion if digital preamphasis or ramp mode is enabled. The counter is clocked with 40MHz provided by the TCU and it has a resolution of 25ns with a minimal clock period of 125ns. The counter is loaded from the Gradient data I/O Bus , witch is 16 bit wide, therefore two access cycles are necessary to write or read the 32 bit counter. To calculate the value, needed by the counter to generate the appropriate clock rate use the following equation (V=T/25ns-1).
V = decimal value to load the counter.
T = Time in ns.
For example, if you want a clock rate of 1us than (1000ns/25ns -1 = 39 or 0x27 hex) you must load the counter with hex 27.

1. 5. 12. Description of special Commands

Soft Reset

The Host CPU can reset the Gradient Control Unit with 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 'STEST' and 'HOLD' are switched on.

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.

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 VME - Control register must be set to enable this function.

Generate NG.-Test Puls

This Device Codes is provided for test and debugging. It is used to simulate the NG Trigger Pulse normally generated by the TCU. The 'TM' bit must be set (HIGH) prior to generate the NG.-Pulse.

Clear DAC Register

To initialize the Gradient D/A converter at the begin of the measurement or after an error condition , this Device Code is provided to set the DAC output at 0 voltage.

Clear AQY-Bus control Flags

Two AQY-Bus control flags (ACF0,ACF1) are used for the handshake mechanisms between the TCU and the GCU. The 'ACF0' flag signals valid data in the AQY-Bus Data register and the 'ACF1' signals valid data in the AQY-Bus control word. The flags are set by the TCU and cleared if the GCU reads the corresponding device codes.

AQ-Bus short description

In the new acquisition system a special Bus (AQ-Bus) is defined as a communication path between the TCU ,FCUs,RCU and GCU. The AQ-Bus is located on the P2 connector of the VME-Bus and is distinguished in two groups:

- 1. Real-time AQ-Bus (unidirectional)
- 2. No real-time AQY-Bus (bidirectional)

The TCU witch generate the exact timing for the acquisition is master on the real-time AQ-Bus and controls the time relevant actions of all other AQ.- Units. For detailed timing and pin assignment see the sheet below. The no real-time AQY-Bus is used as a bidirectional communication path between the AQ.-Units. The TCU send the GCU no real-time depended data and control information needed for special triggering methods or conditional program execution. In the other direction the GCU use this bus to send the TCU a status information (i.e. interrupt status).

1. 6. Logical References

1. 6. 1. I960/VME-Bus Memory map I/O and Device Code space

The GCU Board has 256 kByte Static RAM for Instruction and Data code. This SRAM is mapped in the i960 address space from 1000 0000 to 1003 FFFF. The I/D RAM is byte addressable and its guaranteed to the i960 processor that an access within a 16 byte boundaries is indivisible for other masters until the operation is complete. The VME I/O devices space in region B is long word aligned. The Gradient I/O device space in region C and E is word aligned for I960 access and long word aligned for VME access. The last 256 MBytes region of the address space (F000 0000-FFFF FFFF) is reserved for the EPROM ,whereby only 64KByte are used. This EPROM is only 8Bit width and is byte addressed. The base address of the EPROM starts at address F000 0000 and ends at F000FFFF. The initial boot record block is located in EPROM an can be accessed bye the i960 at address FFFFFF00-FFFFFF2C. Further the EPROM includes a monitor and debug program. The address space is not fully decoded, therefor the EPROM will be seen by the i960 every 64KB steps between the address space of F000 0000 and FFFF FFFF. AT the same reason the I/D RAM will be found by the i960 every 265KByte step between the address space of 0000 0000 and 8000 000. A table of the i960 devices codes used in the Gradient Control Unit are shown below.

1. 6. 2. Global i960 Device code table

1000 0000	- 1003 FFFF	EXTERNAL I/D RAM 256 kByte
1003 FFFF	- 7FFF FFFF	unused EXTERNAL RAM SPACE
8000 0000	- AFFF FFFF	reserved for VME DMA I/O space
B000 0000	- BFFF FFFF	VME REGISTER SET
C000 0000	- CFFF FFFF	GRAD. DATA I/O
D000 0000	- DFFF FFFF	EXTERNAL DATA RAM 8KB
E000 0000	- EFFF FFFF	GRAD. CONT. REG.
F000 0000	- FFFF FFFF	EPROM 64 kByte

1. 6. 3. Global VME Device codes table

1840 0000 - 1843 FFFF	i960 Instr./Data RAM (256KB)
1847 0000 - 1847 1FFF	i960 Data only RAM 8KB (Grad. Parameter)
1847 8000 - 1847 800C	VME Register Set
1847 A000 - 1847 A1A4	Grad. Data I/O ,Timing Generator and Cont Reg.

1. 6. 4. Detailed i960 and VME Device code tables

Table 1: VME Register Set

i960 Address	VME Address	Length	Read Write	Function
B000 0000	1847 8000	BYTE	R/W	VME interrupt vector register
B000 0008	1847 8004	BYTE	R/W	VME control register
B000 0010	1847 8008	BYTE	R/W	local i960 Interrupt register
E000 00E0	1847 A0E0	LWORD	W	generates an interrupt toVME Bus
D0001FFC	1847 1FFC	LWORD	R/W	VME Communication Box
XXXXXXXX	1847 8010	LWORD	W	Soft Reset command
XXXXXXXX	1847 8014	LWORD	W	GO Command

Table 2: Gradient Data I/O Register Set

i960 Address	VME Address	Length	Read Write	Function
C000 0000	1847 A000	Word	R/W	X Gradient Data register
C000 0002	1847 A100	Word	R/W	Y Gradient Data register
C000 0004	1847 A004	Word	R/W	Z Gradient Data register
C000 0006	1847 A104	Word	R/W	B0 Correction Data register
C000 0008	1847 A008	Word	R/W	B1 Correction Data register
C000 000A	1847 A108	Word	R/W	B2 Correction Data register
C000 000C	1847 A00C	Word	R/W	PX X Gradient preload register
C000 000E	1847 A10C	Word	R/W	PY Y Gradient preload register
C000 0010	1847 A010	Word	R/W	PZ Z Gradient preload register
C000 0012	1847 A110	Word	R/W	PB0 B0 Correction preloadregister
C000 0014	1847 A014	Word	R/W	PB1 B1 Correction preloadregister
C000 0016	1847 A114	Word	R/W	PB2 B2 Correction preload register
C000 0020	1847 A020	Word	R	read back last Gradient Data only as Test Function
C000 001E	1847 A11C	Word	W	Clear DAC register (0)

Table 3: Gradient Timing /Control Register Set

i960 Address	VME Address	Length	Read Write	Function
E000 0040	1847 A040	Word	R/W	Timing Generator LOW Word
E000 0042	1847 A140	Word	R/W	Timing Generator HIGH Word
E000 0062	1847 A160	Word/Bit	W	NMI to I960 (signals NG or TIM-CLK)
E000 0064	1847 A064	Word/Bit	W	INT7 to I960 (signals error condition)
E000 0066	1847 A164	Word/Bit	R/W	NG. FLAG
E000 0068	1847 A068	Word/Bit	R/W	Timer Clock FLAG
E000 006A	1847 A168	Word/Bit	R/W	Error FLAG1
E000 006C	1847 A06C	Word/Bit	R/W	Error FLAG2
E000 006E	1847 A16C	Word/Bit	R/W	REAL TIME AQ-BUS FLAG 0
E000 0070	1847 A070	Word/Bit	R/W	REAL TIME AQ-BUS FLAG 1
E000 0072	1847 A170	Word/Bit	R/W	REAL TIME AQ-BUS FLAG 2
E000 0074	1847 A074	Word/Bit	R/W	REAL TIME AQ-BUS FLAG 3
E000 0080	1847 A080	Word/Bit	R/W	Digital Preamph. Control
E000 0082	1847 A180	Word/Bit	R/W	Ramp Control Bit
E000 0084	1847 A084	Word/Bit	R/W	Enable Grad. Int.(NMI) ENDI
E000 0086	1847 A184	Word/Bit	R/W	Test Mode (TM)
E000 0088	1847 A088	Word/Bit	W	NG. Test Pulse
E000 008A	1847 A188	Word/Bit	R/W	generate INT on AQ-Bus
E000 008C	1847 A08C	Word/Bit	R/W	Both Ramp and ENGI set or cleared
E000 00A0	1847 A0A0	Word	W	AQY-Bus INT Status Register
E000 00A0	1847 A0A0	Word	R	AQY-Bus Data & Cont. Flag
E000 00A2	1847 A1A0	Word	R	Test AQY-Bus Interrupt set
E000 00A4	1847 A0A4	Word/Bit	R	Clear AQY-Bus Cont. Flag0
E000 00A6	1847 A1A4	Word/Bit	R	Clear AQY-Bus Cont. Flag1

Table 4: Real-Time AQ-Bus Device Codes

AQ-Bus ADD	Write	AQ-Bus Data Bits	Function
0x90	W	D0=1	select NG-Pulse
0x94	W	D0 - D3	real-time AQ data dummy scan ect.

Table 5: No Real-Time AQ-Bus Device Codes

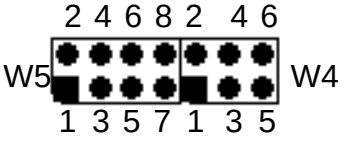
Y-Bus ADD	VME ADD TCU	Read/ Write	Data Bits	Function
0x90	19221640	W	D0 - D3	AQY control Reg.
0x94	19221650	W	D0 - D7	AQY data Reg.
0x98	19221660	R	D0 - D7	AQY data Reg.
0x98	19221660	W	D0=1	AQY cont. Flag0
0x9C	19221670	W	D0=1	AQY cont. Flag1
0x9C	19221670	R	D0 - D1	AQY cont. Flag 0-1
0x94	19221650	R	D0	AQY INT. Flag

1. 7. Operational Settings

1. 7. 1. Configuration

1. 7. 1. 1. VME Interrupt Request (Jumper W5 and W4)

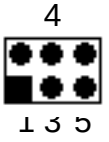
The VME Interrupt request lines are selected by these jumpers. Only one of them should be set.



W5	W4	IRQ
1-2	---	1
3-4	---	2
5-6	---	3
7-8	---	4
---	1-2	5
---	3-4	6
---	5-6	7

1. 7. 1. 2. VME Interrupt Level (Jumper W6)

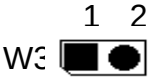
This jumper is used to set the appropriate VME Interrupt Level . The configuration should correspond to the setting of the Interrupt request jumper.



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

1.7.1.3. Clock Divide (Jumper W3)

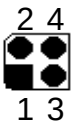
The system clock driven by the I960 can be divide by two with this jumper.



1-2	System Clock
OUT	33MHZ
IN	16MHZ

1.7.1.4. VME Device Code address space selection (Jumper W7)

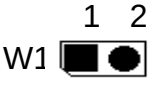
It is possible to use four Gradient Controller Boards in the ASX32 Computer. The Device Code start address of any board can be selected with jumper W7.



3-4	1-2	Base Address
IN	IN	0x18400000
IN	OUT	0x18500000
OUT	IN	0x18600000
OUT	OUT	0x18700000

1.7.1.5. Termination of the 40MHZ signal from the TCU (Jumper W1)

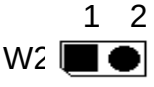
Jumper W1 is used to terminate the 40MHZ coax input from the TCU. This jumper should be inserted if the GCU is the last device receiving the 40MHZ signal.



1-2	
OUT	NOT TERMINATED
IN	TERMINATED

1. 7. 1. 3. Termination of the AQSTA signal from the TCU (Jumper W2)

Jumper W2 is used to terminate the AQSTA coax input from the TCU. This jumper should be inserted if the GCU is the last device receiving this signal.



1-2	
OUT	NOT TERMINATED
IN	TERMINATED

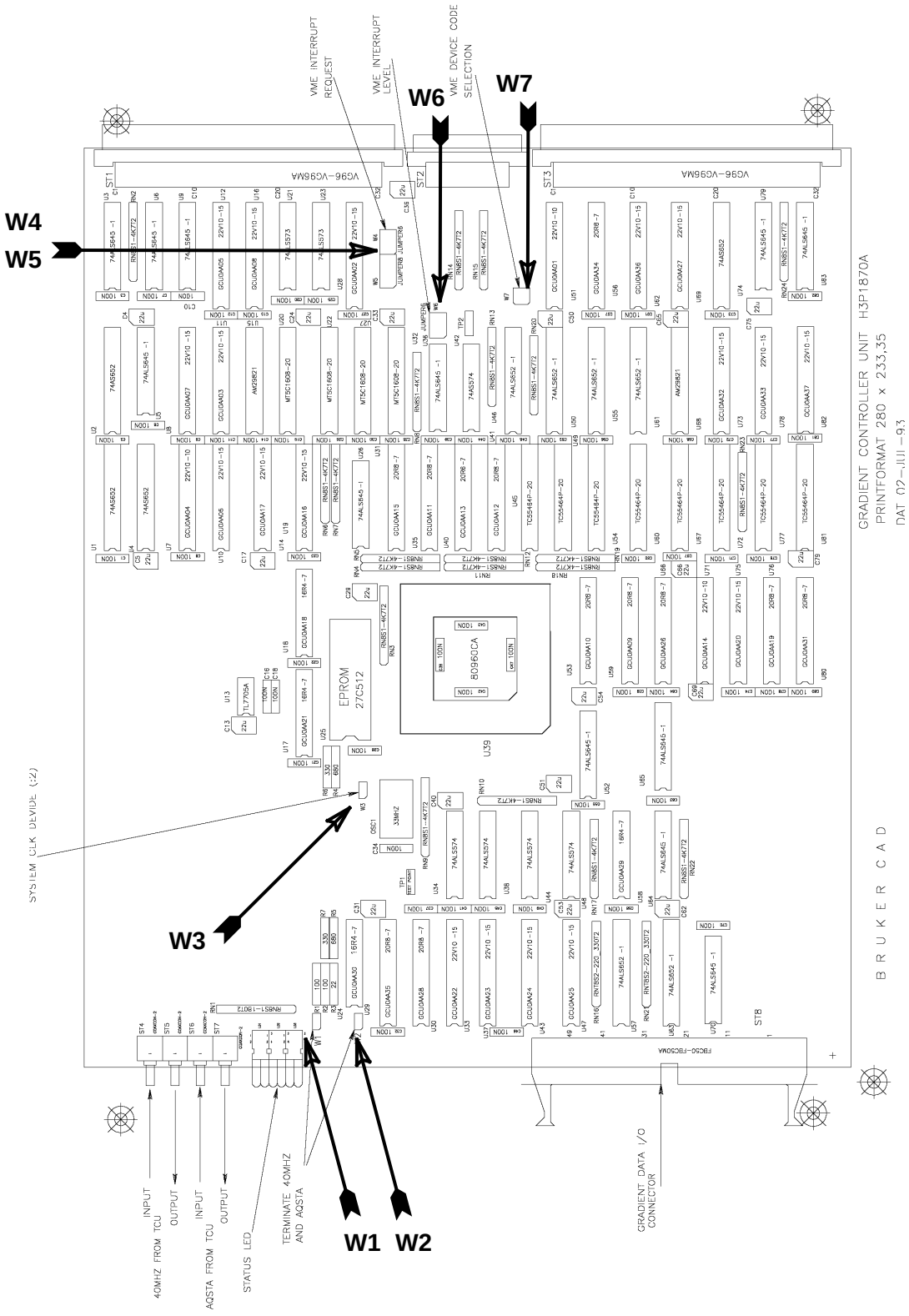


Figure 7: Location of Jumpers

1. 8. Specifications and Connections

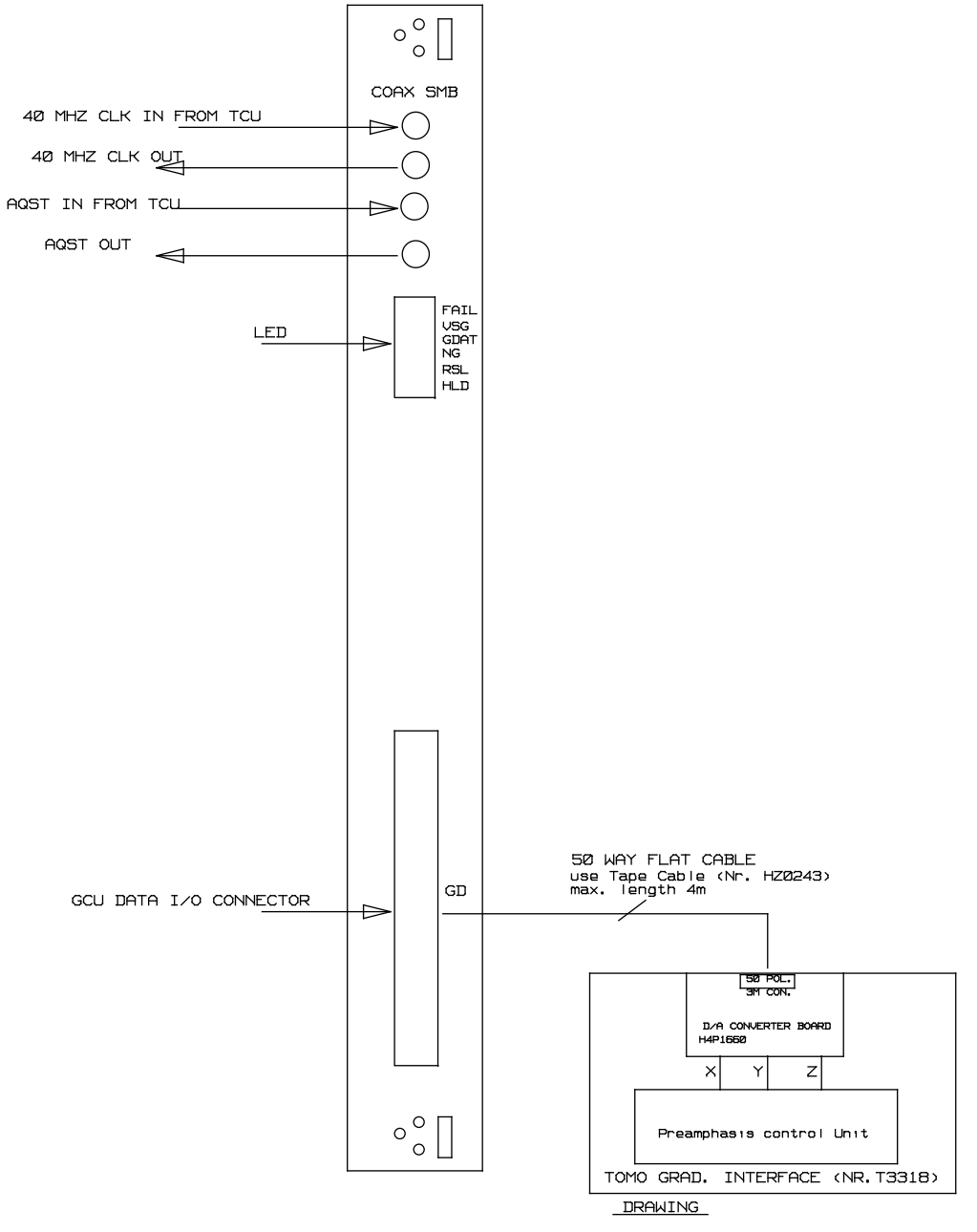
1. 8. 1. Construction and Board Size

The GCU is a VME module of 4TE and contains 1 PCB .
The real size is 233.35 mm by 280 mm . This is the so called ”Double European Standard” format with a nominal plug in depth of 280 mm.

Inputs of the GCU are the AQ-Bus located at the J2 VME connector and the AQ-Strobe and the 40Mhz-Clock located at the frontpanel.

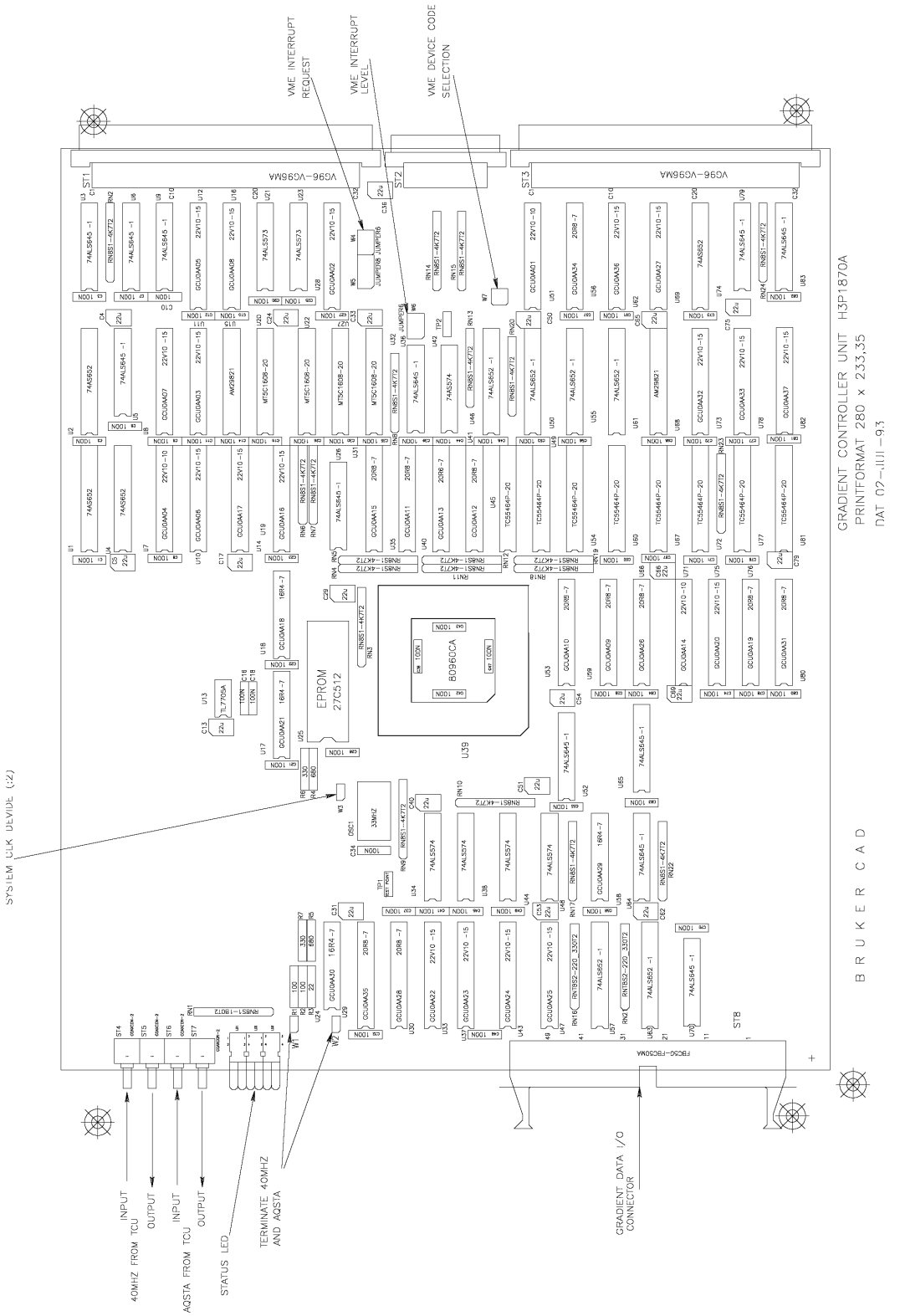
Data Inputs/Outputs of the GCU are the data lines to/from the D/A converter located on the 50-pin connector at the frontpanel.

Figure 8: GCU Front Panel



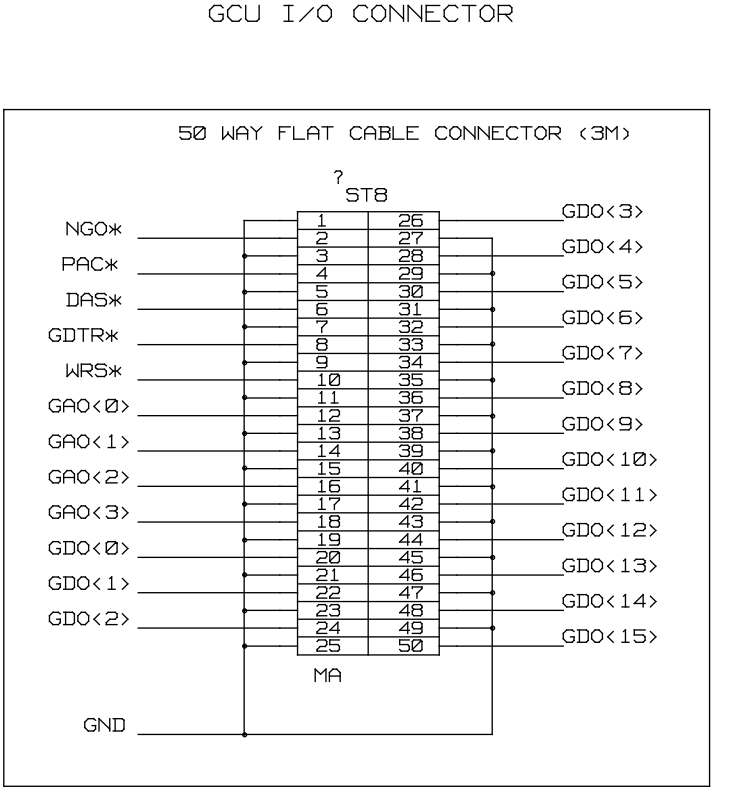
1. 8. 2. Location of Connectors and Controlling Elements

Figure 9: Location of Connectors and Elements



1. 8. 3. Connectors and Signal Allocations

Figure 10: Gradient Data I/O Connector



TTL OUTPUT TO GRAD. D/A CONVERTER BOARD

GAO<3..0> - ADDRESS FOR X,Y,Z GRADIENT REGISTER
GDO<15..0> - 16 BIT BIDIRECTIONAL DATA BUS
WRS* - WRITE STROBE (WRITE GRAD. DATA TO REGISTER)
GDTR* - GRAD. DATA READ/WRITE DIRECTION SIGNAL
DAS* - ADDRESS STROBE
NGO* - NEXT GRADIENT PULSE (TRANSFERS GRAD. DATA TO D/A CONVERTER)
PAC* - PREAMPHASIS CLOCK (OPTION IF DIG. PREAMPHASIS WILL BE USED)

Figure 11: AQ Bus signals on the VME J2-connector

ACQUISITION BUS ASSIGNMENT

	A	C
1	AQA<0>	AQI<0> 1)
2	AQA<1>	AQI<1> 1)
3	AQA<2>	AQI<2> 1)
4	AQA<3>	AQI<3> 1)
5	AQS<0>	AQI<4> 1)
6	AQS<1>	AQI<5> 1)
7	AQS<2>	AQI<6> 1)
8	AQS<3>	AQI<7> 1)
9	GND	GND
10	AQD<0>	AQY<0>
11	AQD<1>	AQY<1>
12	AQD<2>	AQY<2>
13	AQD<3>	AQY<3>
14	AQD<4>	AQY<4>
15	AQD<5>	AQY<5>
16	AQD<6>	AQY<6>
17	AQD<7>	AQY<7>
18	GND	GND
19	AQYWR	AQYAS
20	AQYACK 1)	AQYDS
21	GND	GND
22	AQD<8>	
23	AQD<9>	
24	AQD<10>	GND
25	AQD<11>	ADCON
26	AQD<12>	EPHD
27	AQD<13>	EP
28	AQD<14>	GND
29	AQD<15>	SPHD
30	GND	DWCLK
31		GND
32	AQSTROBE	AQEXEC

1. 8. 4. Power Requirements

	Part-No.	+5 V	+12 V	-12 V	+5 V analog J3: C8	-5 V analog J3: C1,...,C5
GCU0	H2546	7,8 A	0	0	0	0

2. Manufacturing Informations

2. 1. Manufacturing Data

(H-Numbers etc.)

Table 6: Table of Assembly Groups

Amount	Title	Function	Part-Nr.
1	AQX Gradient Control Unit	Assembled PCB	H2546
1		Layout	H3P1870A
1	PCB	Plain PCB	H2547
1	GCU0 PAL set		H3286
1	Front-Panel Assembly Set	Front panel	Hz3103 18878
1	Front-Panel-Ident	”GCU”	Hz3104

2. 2. Introduction Status

2. 2. 1. Configuration

2. 2. 2. Assembling

2. 2. 3. Modifications of the introduced layout

2. 2. 4. Service Informations

2. 3. History of Modifications

EC No.	Date	Part Number	Description of Bugs, Changes and Modifications	Ser.No.	New EC-Level
1838	2.12.93	H2546	Introduction of the AQX Gradient Control Unit	10	01

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:

```
devicetest -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-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 GC/GCU test features

1. Test of the D/A-Converter:

   `dac` Data are written into `xd`, `yd` or `zd`, then decremented and a Next-Gradient-Pulse is given. This is done in a loop until a lower bound is reached. The start value for data is `0xffff`.

2. Test of the `xd`,... registers

   If these registers are physically not present, a special test mode can be enabled that writes a value to the specified register and reads it back from `rbg` (read back gradient). This is done by entering "1" to the question "`read back Gr data from rbg?`".

   Example:

```
spar
read back Gr data from rgb? (1=yes, 0=no) 1
```

   At program start this parameter is set.

3. 1. 1. 9. Special MEM test features

1. The memory configuration

   Use `conf` to check the memory configuration

   `conf` prints out the actual memory configuration, the program `start address` and `size`, and the stack start address. Actually, the program is loaded to 0x200000 (= 2 Mega). On CPU/4 and CCU/5, the stack is at 0x800000 (= 8 Mega).

2. Test commands

   All following tests (except `auto`) use the current parameters `start`, `size` and `value` as set by the user (default is `start=value=0, size=0x40`).
   `auto` uses the total memory region found.
   In `auto`, the regions are split into blocks of at most 4 Mega, that are tested one after the other.
   `auto` calls the `value` test for 8 different values!

`bit`	Test the region with the `values` 1, 2, 4, 8, 0x10, ... 0x80000000
`value`	Test the region with the parameter `value`.
`incr`	Increment the parameter `value` during filling the buffer.
`pat`	Fill the buffer with the patterns 0, 80001, ... i*80001, ...
`comp`	Fill the buffer with `values` 0xaaaaaaaa and 0x55555555.
`addrw`	Set `value=address` for each `address` and write it to the address as a 32-bit-word.
`addrb`	Set `value=address` for each address and write it to the address as a byte. Do this for Bytes 0, 1 and 2 of the address.
`copy`	Copy memory regions byte per byte and compare the two regions.
`cache`	Check whether the memory contents of address 0 changes while the cache address 0 is used for writing.
`refr`	Fill the region with 0x5a, then wait 30 seconds and check it. Then fill the region with 0xa5, wait 30 seconds and check it.
`clear`	Write a pattern to one word in a region and clear the rest. Then check each word.
`tm`	Call all these tests with the current parameters.

4. Timing Diagrams

4. 1. The AQ Bus

Figure 12: The AQ Bus Description

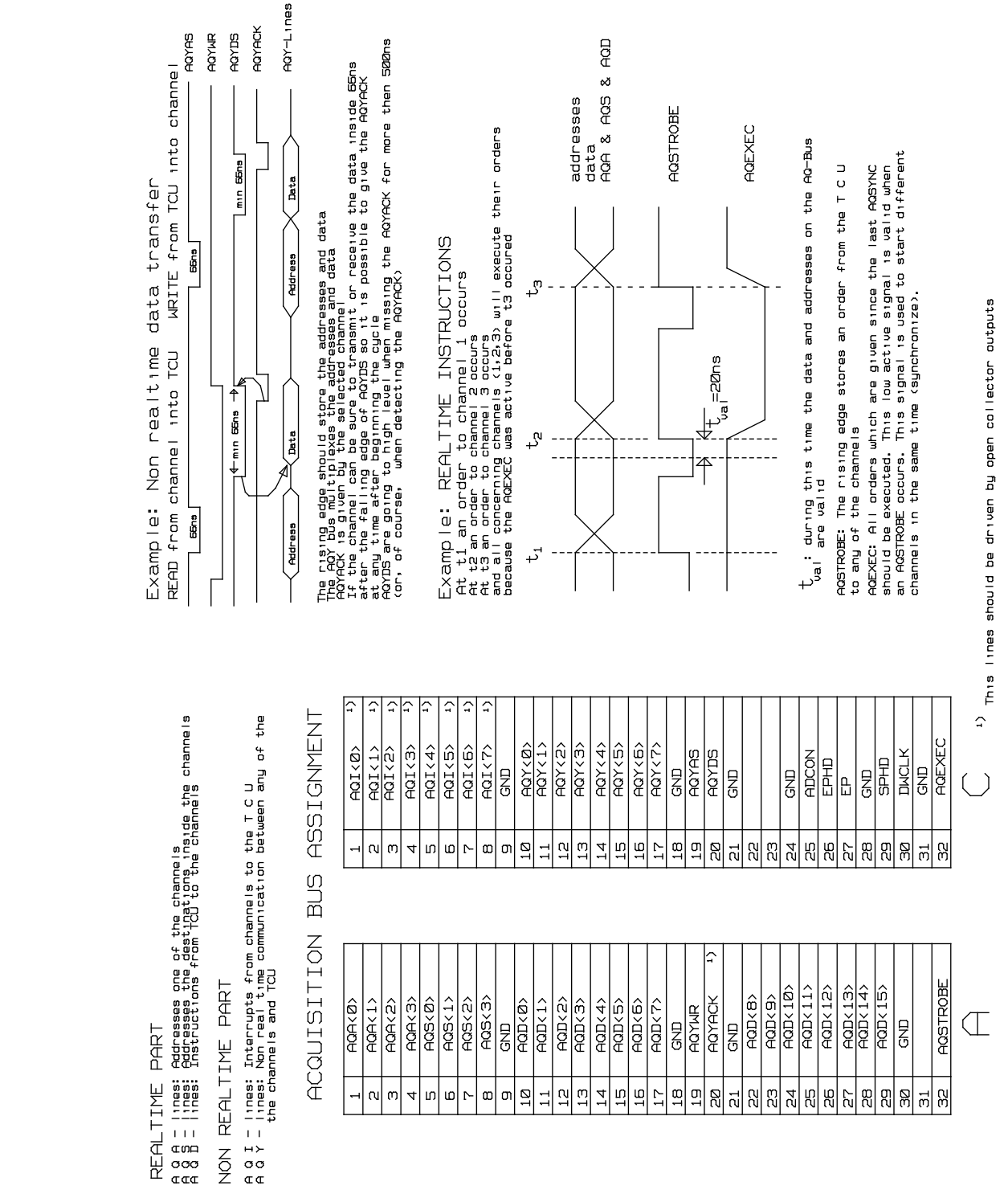
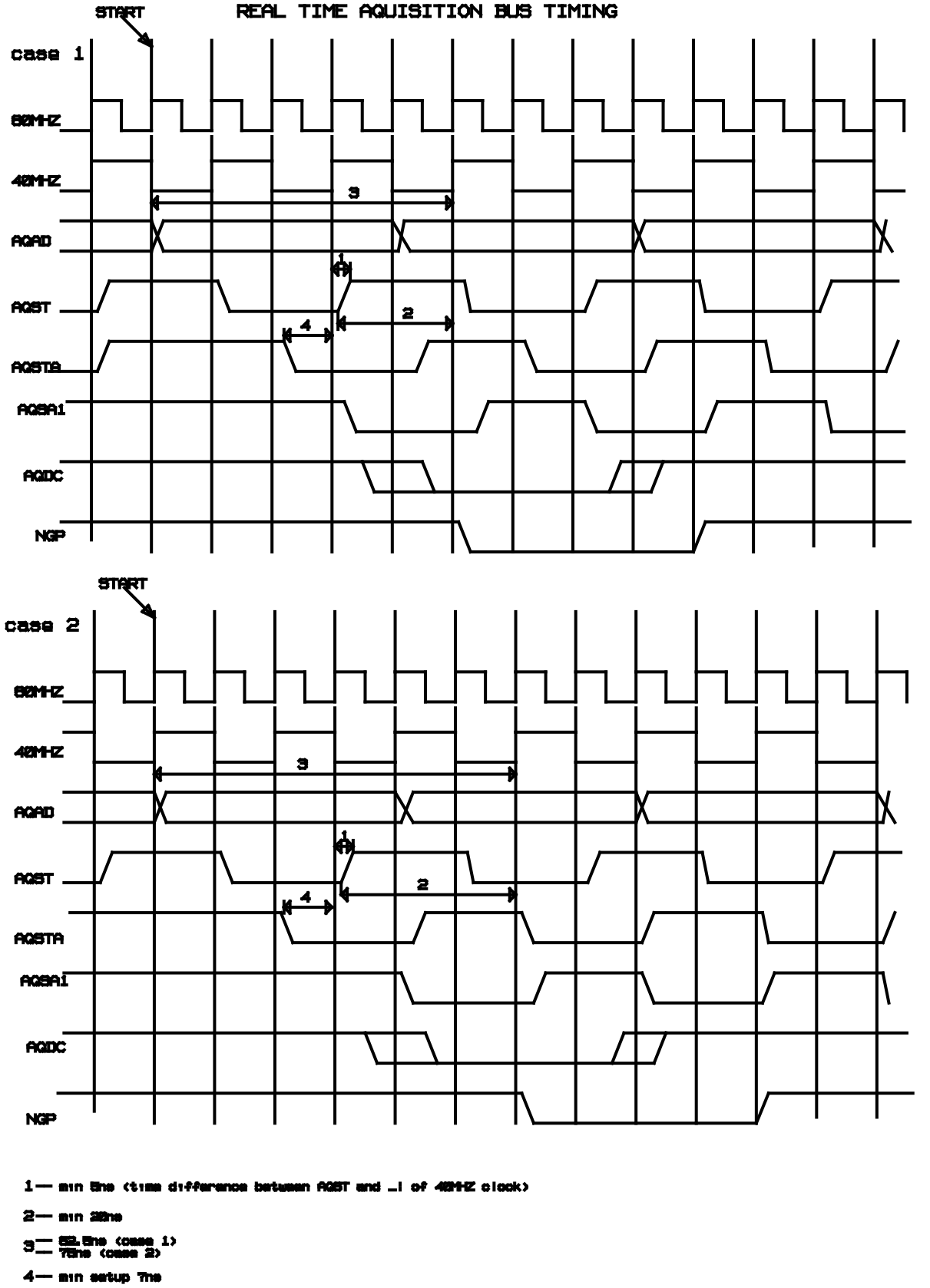


Figure 13: AQ Bus Timing



4. 2. The Gradient Data I/O Timing

Figure 14: Gradient Data I/O Timing