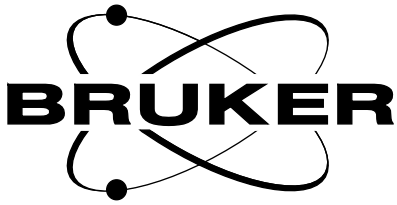


AQX

FCU
Technical Manual

Version 002

BRUKER

The information in this manual may be altered without notice.

BRUKER accepts no responsibility for actions taken as a result of use of this manual. BRUKER accepts no liability for any mistakes contained in the manual, leading to coincidental damage, whether during installation or operation of the instrument. Unauthorised reproduction of manual contents, without written permission from the publishers, or translation into another language, either in full or in part, is forbidden.

This manual was written by

Dr. J. M. Rommel, N. Kuntz, Th. Eckert

Rheinstetten, Germany

P/N: Z31340
DWG-Nr: 1049 002

1 : AQX FREQUENCY CONTROL UNIT

Kuntz, Eckert, Rommel

Contents

Figures

Tables

1. Technical Description

1. 1. General Informations

The FCU is a single board VME bus module.
Some output signals of two neighbouring FCU's can only adapted by one AQX FCU Adapter, *H2560*.
There are two FCU versions with equipped with 64k or 256k byte memory.

FCU Versions	Board	Part No.	Layout No.	EC Level	Software Constrains
FCU0 / 64k	FCU	H2556	H3P1940A	EC ≥ 01	
FCU0 / 256k	FCU	H2564	H3P1940A	EC ≥ 01	
FCU0 / 64k	FCU	H2556	H3P1940A	EC ≥ 05	
FCU0 / 256k	FCU	H2564	H3P1940A	EC ≥ 05	
TOMO FCU 64k	FCU	T5565	H3P1940A	EC ≥ 01	

Table 1: FCU versions

The FCU is a device for controlling the frequency, phase and the amplitude of one NMR-channel.
Frequencies in a range from 0 to 10 MHz can be generated with a "Direct Digital Frequency Synthesizer" (DDS).
For the control of an external frequency device (PTS) there is a PTS-register which can set PTS frequencies from 1 to 999 MHz in steps of 1 MHz.
Relative phases can be set for the DDS frequency by means of two adders. Absolute phases may be generated with an absolute phase reset (reset of the phase register and accu register of the DDS).
The amplitude of the frequency channel can be influenced by two DAC's called MOD and MULT. MOD is used for linear changes of the amplitude , while MULT is used to change the amplitude on a larger (logarithmic) scale and includes the setting of fixed attenuators.
All FCU's are equipped with memory from which all registers can be loaded with precalculated values.
The loading of registers is normally initiated by commands from the TCU which are received via the ACQ-bus.
But the FCU has also autonomous timing capabilities which make it possible to execute whole sequences of commands in proper timing , without TCU interaction. This is required for the execution of shaped pulses (CPD programs, etc.).

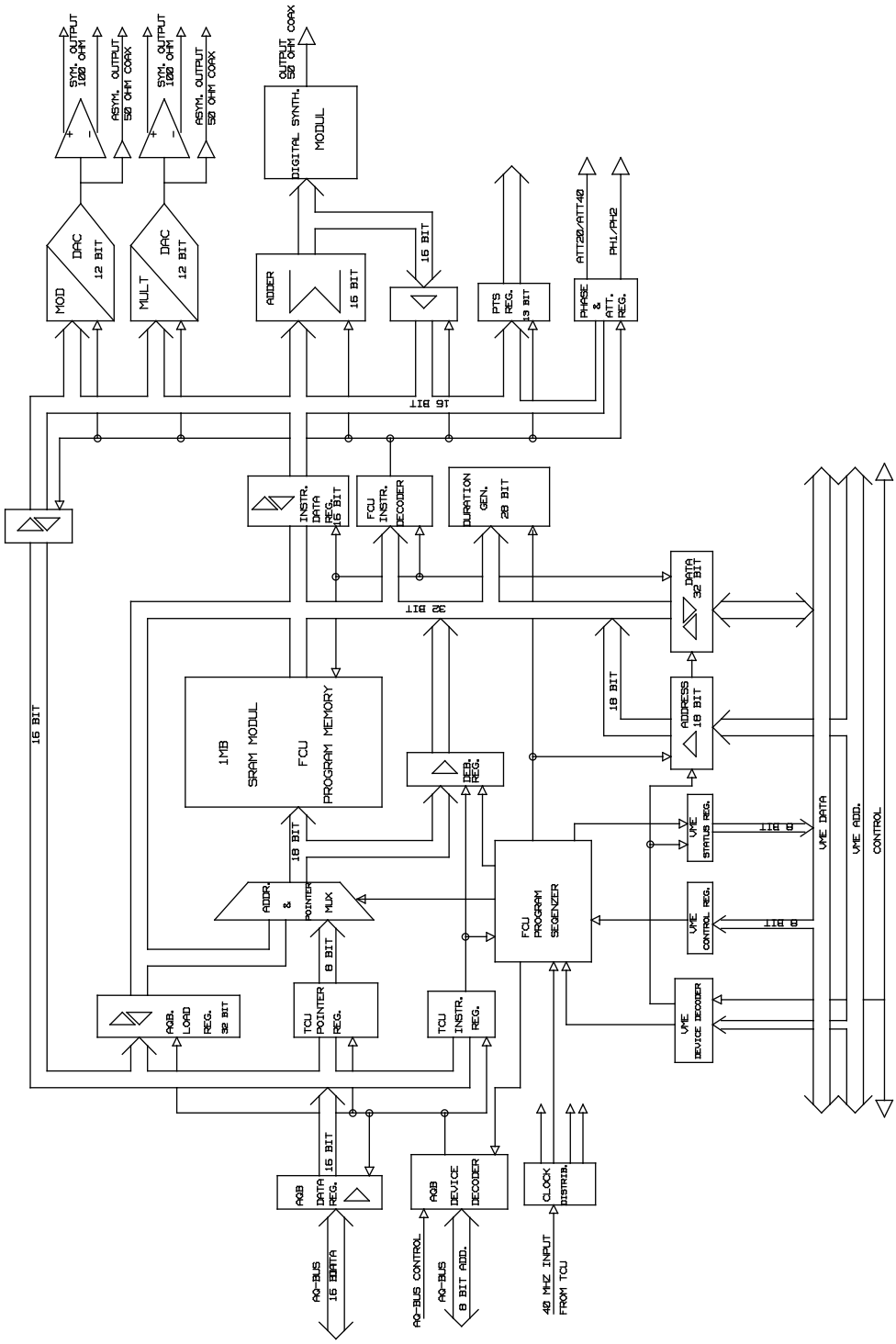
1. 2. Features

- Minimal duration 50 nsec
- Maximal duration 53 sec
- Minimal resolution 25 nsec
- Real time memory range max. 256 kWords
- Frequency synthesis from 0 to 10 MHz
- 256 sets of pointer registers

1. 3. Architecture

1. 3. 1. Block Diagram

Figure 1: FCU block diagram



1. 4. Logical References

Table 2: Memory Map and Device Codes

Address of VME-Bus Access	Operand included W-Addr. on D23-D16	Address of ACQ-Bus Access	Destination	name	Mode R/W	Size of Operand Byte 3 2 1 0
19300000 to 193FC000			1MB Real Time Program Memory		R/W	b b b b
193FFxxx	74		MULT-DAC	mult	W	x a b b
193FFxxx	70		MOD-DAC	mod	W	x a b b
193FFxxx	78		PTS Register	pts	W	x a b b
193FF004	not		PTS Register	pts	R	x x b b
193FFxxx	70;74;78		Instr./Data-Register	idr	W	x a b b
193FF018	not		Instr./Data-Register	idr	R	x x b b
193FFxxx	48		Add: A-Port + B-Port	add	W	x a b b
193FF008	not		Read Result of Adder	add	R	x x b b
193FFxxx	40		Load Adder, B-Port	iadd	W	x a b b
193FFxxx	44		Load Adder, A-Port	lp	W	x a b b
		x0	TCU Pointer & Instr.		W	x x b b
193FFxxx	4c		TCU Pointer & Instr.	tcup	W	x a b b
193FF010	not		TCU Pointer & Instr.	tcup	R	x x b b
193FF00C	not		TCU-Register	tcur	R	x x b b
193FF014	not		RAM Addr & Instr Status	as	R	b b b b
193FF01C	not		Board EC Level	ec	R	x x b b
193FF020	not		FCU Status & Contrl. Reg.	st	R/W	x x b b
193FF024	not		Read Status & Clear ST0	st0	R	x x b b
193FF028	not		Read Status & Clear ST1	st1	R	x x b b
193FF038	not		VME Address Reg.	vadd	R	b b b b
193FF03C	not		VME Data Reg.	vdat	R	x x b b
193FFxxx	50		ACQ Load Register	lacqr	W	x a b b
		x2	low word		W	x x x x
193FF02C	not		Software Reset	vres	W	x x x x
193FF030	not		Run FCU	vrun	W	x x x x
193FF034	not		Step FCU	vstep	W	x x x x
193FF038	not		Stop FCU	vstop	W	x x x x

Address of VME-Bus Access	Operand included W-Addr. on D23-D16	Address of ACQ-Bus Access	Destination	name	Mode R/W	Size of Operand Byte 3 2 1 0
193FF03C	not		FCU Output Enable	foen	W	x x x x
193FFxxx	04		Load low Frequency Storage Register 0	lfs0	W	x a b b
193FFxxx	08		Load middle Frequency Storage Register 0	mfs0	W	x a b b
193FFxxx	0C		Load high Frequency Storage Register 0	hfs0	W	x a b b
193FFxxx	14		Load low Frequency Storage Register 1	lfs1	W	x a b b
193FFxxx	18		Load middle Frequency Storage Register 1	mfs1	W	x a b b
193FFxxx	1C		Load high Frequency Storage Register 1	hfs1	W	x a b b
193FFxxx	24		Load Frequency Working Reg. from Storage Reg. 0	fws0	W	x a b b
193FFxxx	28		Load Frequency Working Reg. from Storage Reg. 1	fws1	W	x a b b
193FFxxx	60		Reset all internal registers and counters	ires	W	x a b b
193FFxxx	64		Reset Accu and counter (zero phase clear)	ares	W	x a b b
193FFxxx	68		Reset Phase Working Reg.	pres	W	x a b b
193FFxxx	74		Set ATT20/ATT40 Bit0/Bit1	att	W/R	x x x b
193FFxxx	3C		Set PH1/PH2	ph	W R	Bit17/18 Bit2/3

1. 4. 1. FCU Program

1. 4. 1. 1. FCU Instructions

Table 3: List of FCU Instructions

n| Instr.	|B31|B30|B29|B28|B27|B26|B25|B24|B23|	Kommentar
2| STOP	| 0 | 0 | 0 | x | x | x | 0 | 0 | x |	Halt und rette Pointer
4| DURA	| 0 | 0 | 0 | x | x | x | 0 | 1 | 1 |	Duradresse steht in B17..B0
4| WAIT	| 0 | 0 | 0 | x | x | x | 1 | 0 | 1 |	Warte bis Durat. fertig
4| DUR0	| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |	LREG0 -> Durationgenerator
4| DUR1	| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |	LREG1 -> Durationgenerator
4| DUR2	| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |	LREG2 -> Durationgenerator
4| DUR3	| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |	LREG3 -> Durationgenerator
4| DUR4	| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |	LREG4 -> Durationgenerator
4| DUR5	| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |	LREG5 -> Durationgenerator
4| DUR6	| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |	LREG6 -> Durationgenerator
4| DUR7	| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |	LREG7 -> Durationgenerator
1| CONT	| 0 | 0 | 1 | x | x | x | 0 | 0 | 0 |	inkrementiere Pointer
1,5| CONT75	| 0 | 0 | 1 | x | 1 | x | 0 | 0 | 0 |	wie CONT aber mit 75ns
4| RELD	| 0 | 1 | 0 | x | x | x | 0 | 0 | 0 |	INIT-Register -> Pointer
4| CRELDA	| 0 | 1 | 0 | x | x | x | 1 | 0 | 0 |	wenn A=0 -> RELD sonst CONT
4|CRELDB	| 0 | 1 | 0 | x | x | x | 1 | 1 | 0 |	wenn B=0 -> RELD sonstCONT
4I CALL	| 0 | 1 | 1 | x | x | x | 0 | 0 | 0 |	Call to Subroutine B17..B0
4| CCALLA	| 0 | 1 | 1 | x | x | x | 1 | 0 | 0 |	wenn A=0 -> CALL sonst CONT
4| CCALLB	| 0 | 1 | 1 | x | x | x | 1 | 1 | 0 |	wenn B=0 -> CALL sonst CONT
5| LUMP0	| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |	JUMP wenn LREGi < 2**18 - 1
5| LUMP1	| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |	sonst wie CONT und inkre-
5| LUMP2	| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |	mentiere LREGi
5| LUMP3	| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |	2**18-1 ist 11..111111
5| LUMP4	| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |	Bsp: 11..111101 wird
5| LUMP5	| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |	bei Schleife zurueck
5| LUMP6	| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |	2 mal durchlaufen
5| LUMP7	| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |	i = 0..7
4| JUMP	| 1 | 0 | 1 | x | x | x | 0 | 0 | 0 |	Jump to Adresse B17..B0
4| CJUMPA	| 1 | 0 | 1 | x | x | x | 1 | 0 | 0 |	wenn A=0 -> JUMP sonst CONT
4| CJUMPB	| 1 | 0 | 1 | x | x | x | 1 | 1 | 0 |	wenn B=0 -> JUMP sonst CONT
4| LDLP0	| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |	B17..B0 -> LREG0 beachte,
4| LDLP1	| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |	B17..B0 -> LREG1 dass die
4| LDLP2	| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |	B17..B0 -> LREG2 anderen
4| LDLP3	| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |	B17..B0 -> LREG3 Bits
4| LDLP4	| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |	B17..B0 -> LREG4 B31..B18
4| LDLP5	| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |	B17..B0 -> LREG5 nicht
4| LDLP6	| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |	B17..B0 -> LREG6 geloescht
4| LDLP7	| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |	B17..B0 -> LREG7 sind !!!

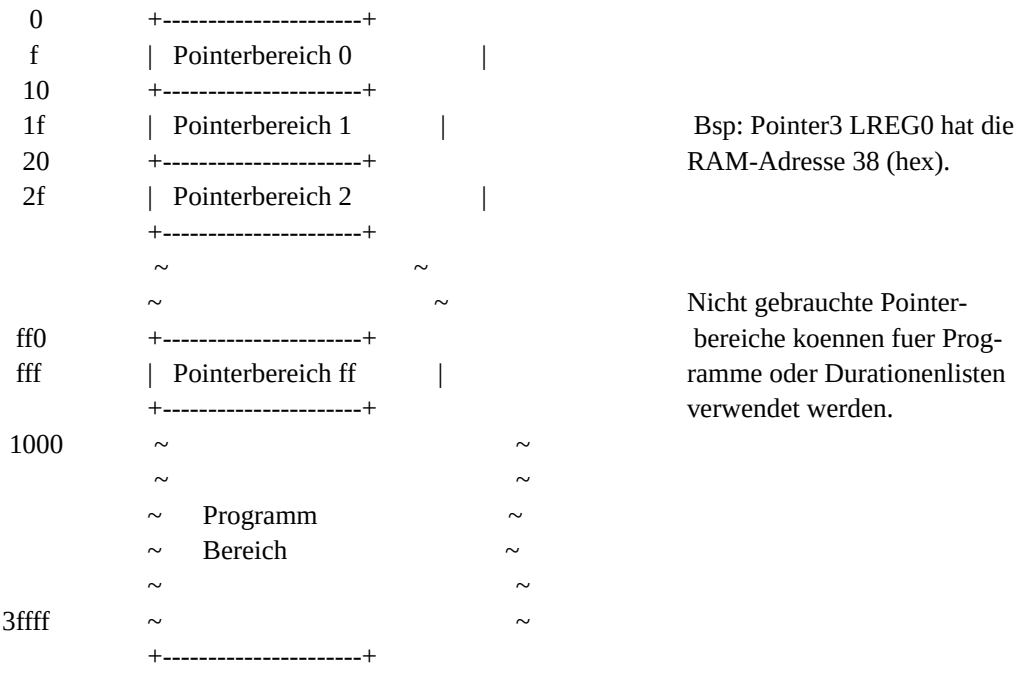
4| RET	| 1 | 1 | 1 | x | x | x | 0 | 0 | 0 |	RET
4| CRETA	| 1 | 1 | 1 | x | x | x | 1 | 0 | 0 |	wenn A=0 -> RET sonst CONT
4| CRETB	| 1 | 1 | 1 | x | x | x | 1 | 1 | 0 |	wenn B=0 -> RET sonst CONT

DURA,CALL,CALLA,CCALLB,LUMPi,JUMP,CJUMPA,CJUMPB und LDLPi benoetigen
das Datenfeld B17..B0. Alle anderen Befehle koennen eine Ausgabe taetigen.
n bedeutet die Zahl der 50ns Zyklen, die der Befehl braucht.
Bedingte Operationen dauern unabhaengig von der Bedingung immer 4 Zyclen.
WAIT kann auch dann gegeben werden, wenn keine Duration laeuft. WAIT
hat dann 4 Zyklen.

1. 4. 1. 2. Register and Pointer Assignment

B11..B4 sind die Adressen der Pointer. Jeder Pointer hat als
Unteradressen B3..B0. Die Unteradressen bedeuten im einzelnen:

B3..B0 = 0: aktueller Programmzaehler des FCU-Programms
B3..B0 = 1: INIT-Register (Anfangswert des Programmzaehlers)
B3..B0 = 2: Stack6 (Rueecksprungadresse des 6. Unterprgr.)
B3..B0 = 3: Stack5 (Rueecksprungadresse des 5. Unterprgr.)
B3..B0 = 4: Stack4 (Rueecksprungadresse des 4. Unterprgr.)
B3..B0 = 5: Stack3 (Rueecksprungadresse des 3. Unterprgr.)
B3..B0 = 6: Stack2 (Rueecksprungadresse des 2. Unterprgr.)
B3..B0 = 7: Stack1 (Rueecksprungadresse des 1. Unterprgr.)
B3..B0 = 8: LREG0 Register fuer Loop oder Duration
B3..B0 = 9: LREG1 Register fuer Loop oder Duration
B3..B0 = a: LREG2 Register fuer Loop oder Duration
B3..B0 = b: LREG3 Register fuer Loop oder Duration
B3..B0 = c: LREG4 Register fuer Loop oder Duration
B3..B0 = d: LREG5 Register fuer Loop oder Duration
B3..B0 = e: LREG6 Register fuer Loop oder Duration
B3..B0 = f: LREG7 Register fuer Loop oder Duration

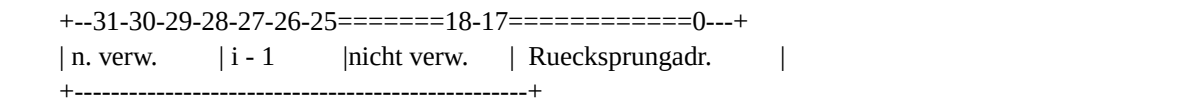


Format des Programmzaehlers:

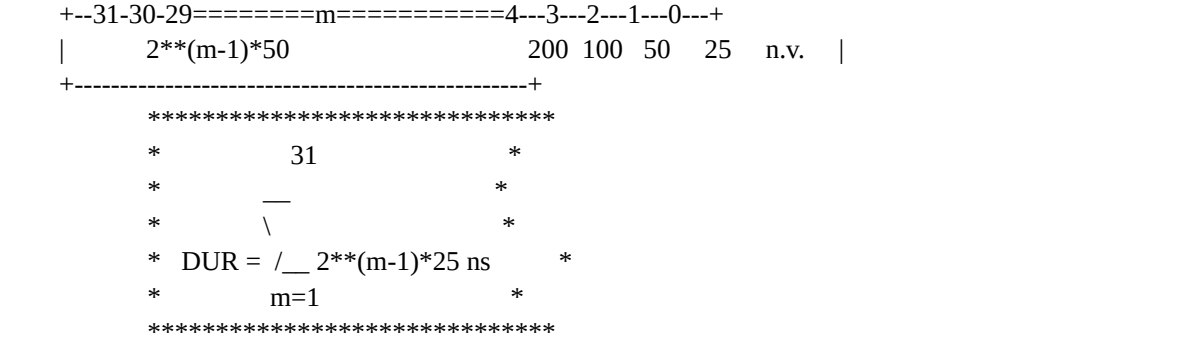
Der Unterprogrammlevel ist eingetragen in B28,...,B26

Format des Initialisierungsregisters INIT

Format des Stackregisters i



Format des Durationwortes:



1. 4. 1. 3. The Program Word

```
    +--31-30-29-28-27-26-25-24-23-22-21-20-19-18-17==========0---+
| I N S T     | i       |I N S T   |D E V I C E      |D A T E N
 ------------------------------------------------------------+
```

INST : siehe FCU-Instructionen
i : Nummer des Registers
DEVICE : Kode fuer Ausgabe des Datums

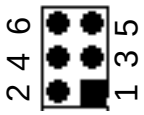
```
"   Device code table
"   -------------------------

"   The Devices  ( Digital synthesizer & D/A Converters )
"   is controlled by instruction bits 22 ... 18. The synthe-
"   sizer is controlled by an instruction pipeline to save
"   all instructions automatically.
"
"    1   ( 00001 ): Load  low   frequency storage register 0
"    2   ( 00010 ): Load middle frequency storage register 0
"    3   ( 00011 ): Load  high  frequency storage register 0
"
"    5   ( 00101 ): Load  low   frequency storage register 1
"    6   ( 00110 ): Load middle frequency storage register 1
"    7   ( 00111 ): Load  high  frequency storage register 1
"
"    9   ( 01001 ): Load frequency working register from
"                   frequency storage register 0
"   10   ( 01010 ): Load frequency working register from
"                   frequency storage register 1

"   15   ( 01111 ): Load Quad Phase

"   16   ( 10000 ): Load B - Input Adder

"   17   ( 10001 ): Load phase register
"   18   ( 10010 ): Add phase register

"   19   ( 10011 ): Set TCU-Pointer by instruction
"   20   ( 10100 ): Set TCU-Output register by instruction

"   24   ( 11000 ): Reset all internal registers and counters
"   25   ( 11001 ): Reset accu & counter ( zero phase clear )
"   26   ( 11010 ): Reset phase working register
"
"   27   ( 11011 ): Load frequency register from adder-result
"   28   ( 11100 ): Load D/A Converter 0
"   29   ( 11101 ): Load D/A Converter 1
"   30   ( 11110 ): Load frequency register ( PTS )
"   31   ( 11111 ): Load NMR - Register
"
"
"   Note !!! Instructions 1, 2, 3, 5, 6, 7, 16, 17 use the
"   Adder B - Port.  Restauration by Instruction 16
```

1. 5. Operational Settings

1. 5. 1. Configuration

1. 5. 1. 1. VME Device Code address space selection (Jumper W7)

It is possible to use eight FCU Boards in the Aquisiton System. The Device Code start address of any board depends on the setting of jumper W5. The diagram shows the correct setting of jumper W5 for various FCU Boards.



W5

3-4	3-4	1-2	Base Address	FCU Nr.
IN	IN	IN	0x19300000	1
IN	IN	OUT	0x19400000	2
IN	OUT	IN	0x19500000	3
IN	OUT	OUT	0x19600000	4
OUT	IN	IN	0x19700000	5
OUT	IN	OUT	0x19800000	6
OUT	OUT	IN	0x19A00000	7
OUT	OUT	OUT	0x19B00000	8

1. 5. 1. 2. Output voltage range selection for MULT and MOD (Jump. W15,W4)

The voltage range of the non differential (MULT,MOD) outputs can be set between 0V and +2,5V, or 0 and -2,5V at 50 Ohm load. W15 is the jumper for MULT and W4 the equivalent jumper for MOD. These two non differential outputs will be present even when the differential outputs are used.

default

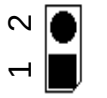
W15
(W4)

3-4	1-2	range
IN	IN	0V to -2.5V

2-4	1-3	range
IN	IN	0V to +2.5V

1. 5. 1. 3. Termination of the 40MHZ signal from the TCU (Jumper W2)

Jumper W2 is used to terminate the 40MHZ coax input from the TCU. This jumper should be inserted if the FCU is the last device receiving the 40MHZ signal.For all other FCU's this jumper should be out.

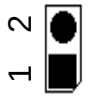


W2

1-2	
OUT	NOT TERMINATED
IN	TERMINATED

1. 5. 1. 4. Termination of the AQSTA signal from the TCU (Jumper W3)

Jumper W3 is used to terminate the AQSTA coax input from the TCU. This jumper should be inserted if the FCU is the last device receiving this signal.For all other FCU's this jumper should be out.



W3

1-2	
OUT	NOT TERMINATED
IN	TERMINATED

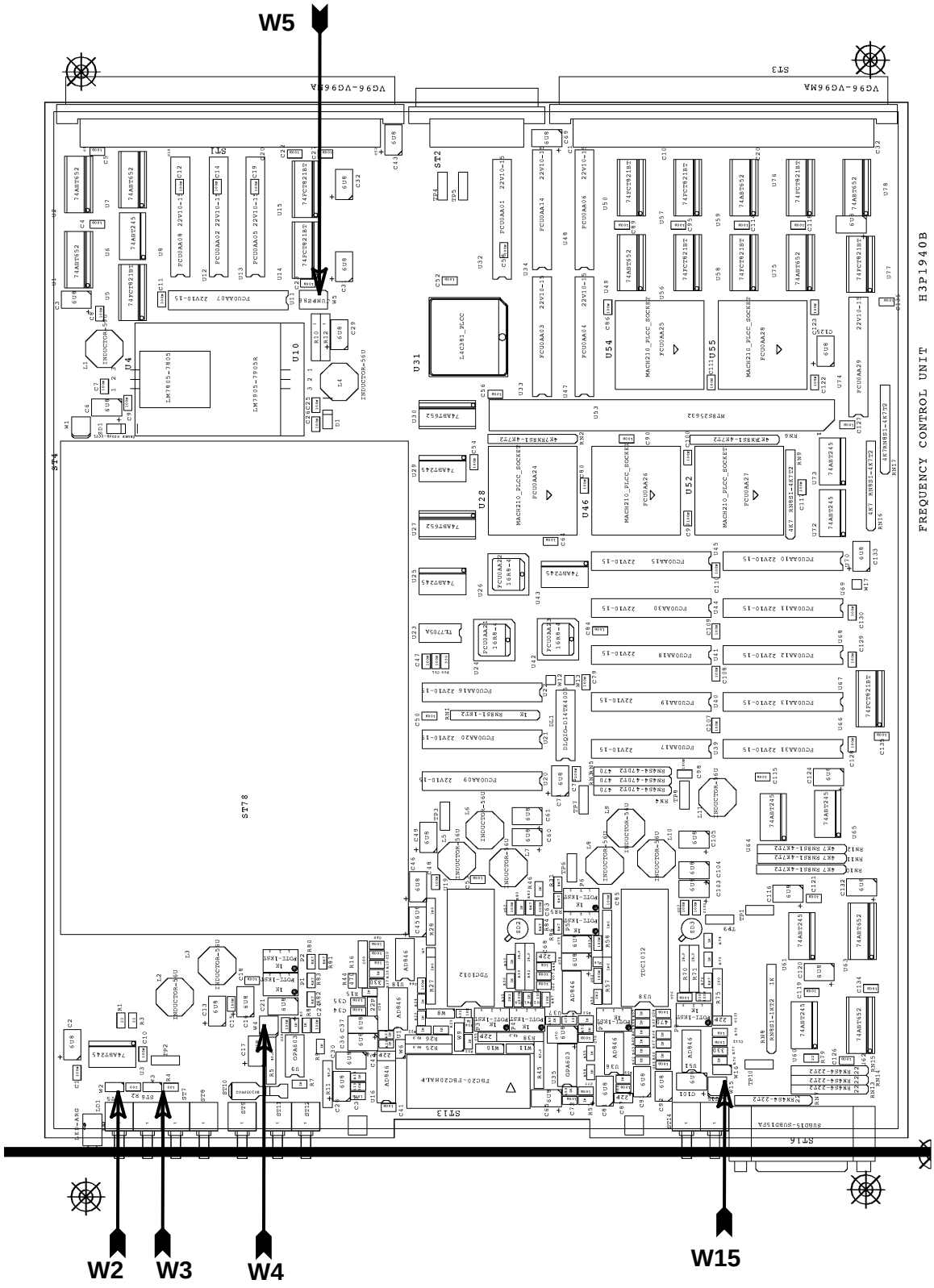


Figure 2: Location of Jumpers

1. 6. Specifications and Connections

1. 6. 1. Construction and Board Size

The FCU has a with of 4 TE and contains one board of full plug in length.
The board size is 280mm by 233.35mm.

Mounted at the front panel of two FCU's respectively is a so called "FCU-Adapter" (see Figure 1:). It collects all signals with the same destination ("MMA") from these FCU's and some from the TCU and lead them through one cable.

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

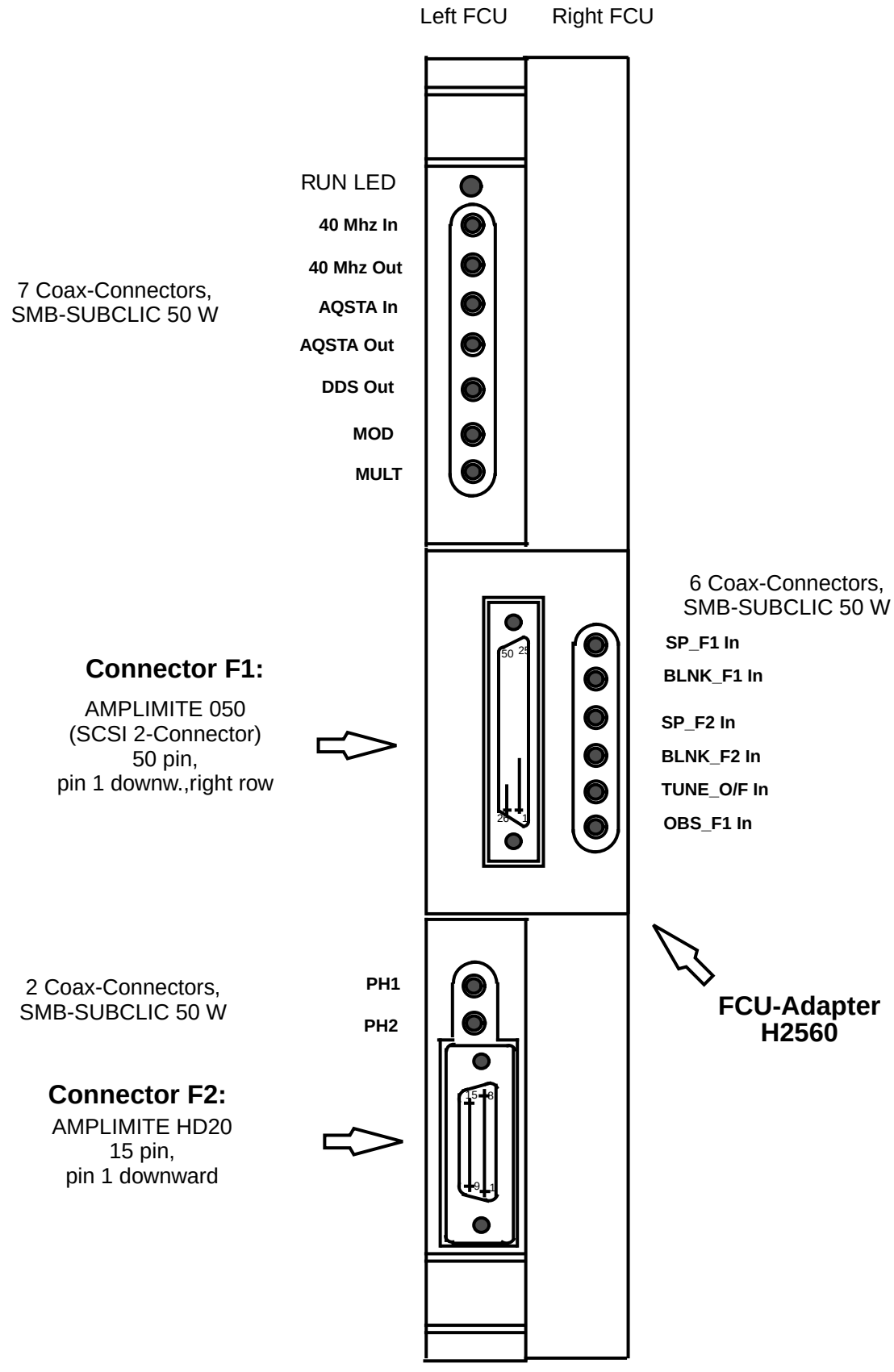


Figure 1: Front View of FCU

1. 6. 3. Connectors and Pin Assignments

All outputs (except those with note 1) switch between TTL-Levels. They are in a high impedance state after power-on and pulled up by a resistor of 1kΩ . 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 odd numbered pin row and one row of the even numbered. The mounting direction is pin 1 (right row) respectively pin 2 (left row) downwards (see Figure 1:).
The standard cable (e.g. ”SCSI-Cable”) has always an odd numbered line twisted with the next even numbered line. That means line 1 is twisted with line 2, line 3 with 4 and so on

1. 6. 3. 1. Connector F1

All signals are outputs .

Signals with extension ”_A” have its source at ”left” FCU, those with ”_B” at the ”right” FCU (see FigureFigure 1:).

Note 1: These outputs deliver differential signals with a voltage range between
+1 Volt and -1 Volt. The impedance is 50 Ω.

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

Table 4: Pin Assignment of F1

Connector F1 AMPLIMITE 050				
Pin	**Signal (right row)**	**Pin**	**Signal (left row)**	**Notes**
1		2		
3		4		
5		6		
7	GND	8	GND	
9	-MUL_A	10	+MUL_A	1
11	GND	12	GND	
13	-MOD_A	14	+MOD_A	1
15	AT20_A	16	GND	
17	AT40_A	18	GND	
19	BLNK_F1	20	GND	
21	SP_F1	22	GND	
23	OBS_F1	24	GND	
25	GND	26	GND	
27	-MUL_B	28	+MUL_B	1
29	GND	30	GND	

Connector F1 AMPLIMITE 050				
Pin	**Signal (right row)**	**Pin**	**Signal (left row)**	**Notes**
31	-MOD_B	32	+MOD_B	1
33	AT20_B	34	GND	
35	AT40_B	36	GND	
37	BLNK_F2	38	GND	
39	SP_F2	40	GND	
41		42		
43		44		
45	TUNE_O/F	46	GND	
47		48		
49		50		

1. 6. 3. 2. Connector F2

Table 5: Pin Assignment of F2

Connector F1 AMPLIMITE HD20	
Pin	**Signal**
1	PTS_08
2	PTS_1
3	PTS_2
4	PTS_4
5	PTS_8
6	PTS_10
7	PTS_20
8	PTS_40
9	PTS_80
10	GND
11	PTS_100
12	PTS_200
13	PTS_400
14	PTS_800
15	GND

1. 6. 4. Power Requirements

	Part-No.	+5 V	+12 V	-12 V	+5 V analog J3: C8	-5 V analog J3: C1,...,C5
FCU0	H2556 H2564	5,0 A	0,2 A	0,3 A	0,1 A	0,4 A

2. Manufacturing Informations

2. 1. Manufacturing Data

Table 6: Table of Assembly Groups

Amount	Title	Function	Part-Nr.
1	FCU 64k	Assembled PCB	H2556
1	FCU 256k	Assembled PCB	H2564
1	FCU	PAL set	H3290
1	FCU	Layout	H3P1940A
1	FCU	Plain PCB	H2557
1	FCU	Frontpanel	HZ1560
1	FCU	Frontpanel Assembly Set	HZ0943
1	FCU	Frontpanel Ident	HZ2381
1 for 2 FCU’s	FCU-Adapter	Assembled PCB	H2560
1 for 2 FCU’s	FCU-Adapter	Layout	H4P2010
1 for 2 FCU’s	FCU-Adapter	Plain PCB	H2561
1 for 2 FCU’s	FCU-Adapter	Assembly-Set	HZ2463

2. 2. Introduction Status

2. 2. 1. Configuration

2. 2. 2. Service Informations

2. 3. History of Modifications

EC No.	Date	Part Number	Description of Bugs, Changes and Modifications	Ser.No.	New EC-Level
1748	5.7.93	H2556 H2564	Introduction with layout version H3P1940A including a list of modifications	010 010	01
1762	14.7.93	H2556 H2564	Introduction of layout version H3P1940B, which includes all modifications that had been carried out so far on layout H3P1940A	089 010	02
1775	16.8.93	H2556 H2564	Turning capacitor C105 to its correct direction because of an incorrect label on both layout versions		03
1780	2.9.93	H2556 H2564	Terminating the IDC clock line because of an incorrect loading of the Instruction Data Register	115 010	04
2054	6.9.94	H2556 H2564	Introduction of a separate device code for setting PH1/PH2 (necessary for DSX)	710 105	05
2074	9.11.94	H2556 H2564	Modify floating DDS input FRCLR to a driven one	822 105	06
2103	16.1.95	T5565	Introduction of the special so called TOMO FCU 64k		01
2133	5.4.95	H2556 H2564	Functionality of automatic phase error compensation by CORTAB		07

3. Testing

Extended FCU hardware tests can be carried out with the special designed FCU Test Interface *TIF0* (H5808).

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:

device`test -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 devicetest
```

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`

3.1.1.8.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
```

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
```