40 MS/s DUAL CHANNEL DIGITAL STORAGE OSCILLOSCOPE TRACE 8608A/8708

Programming Manual

(Version 1.0)

April 1989 **TRACE** Vienna

TABLE OF CONTENTS

1 INTRODUCTION

This interface is a general purpose bus line interface according to the IEEE-488/IC-625 and the RS232-C/V24 document by which the oscilloscope is adapted to make communication possible with any other IEEE-/IEC-bus and RS232-C/V24 compatible controller.

For more detailed information about the functioning of the hardware of the interface refer to the 8608A/8708 SERVICE MANUAL.

2 SETTING THE INTERFACE

For setting of the interface refer to 8608A/8708 Operating Manual Chapter 4.2.12.

3 IEEE-488/IEC-625 BUS INTERFACE

3.1 INTRODUCTION

The IEC-bus interface is designed for interconnecting several instruments, programmable or non-programmable, to form a measuring system. Internatinal agreements on such a system (using byte serial, bit parallel operation) are defined in IEC Publication 625-1.

To simplify cabling and to allow for extension, the interface is organized as a bus-line system. All instruments are interconnected via a common set of 16 lines.

As the bus-line is common to all instruments, to communicate effectively the problems of interfacing have to be solved within the instruments themselves; i.e. they must be designed with inbuilt IEC-bus facilities. This enables the user to select the most suitable instruments for the system, regardless of make, knowing that interfacing is not a major problem.

These IEC-bus facilities to be described are the functional,electrical and mechanical requirements of instruments designed for connection to the interface, as defined in IEC Publication 625.

Major characteristics

- Instruments are be of various manufacture.
- Different data rates can be adapted.
- Asynchronous data transfer (up to 1 Mbyte/s) is possible without a controller.
- System flexibility allows rapid interchange of simple and highly complex equipment set-ups.
- No cabling problems.

Variants:

One other standard conforms to the IEC 625-standard in all aspects except for the type of interface connector used (see Section 3.12. Mechanical specifications). This variant is the American standard IEEE-488, sometimes referred to as:

HP-IB HP Interface Bus

GPIB General-Purpose Interface Bus.

3.2 STRUCTURE OF THE IEEE-488/IEC-625 BUS

IEC-bus compatible instruments:

To satisfy the necessary requirements, interface functions are built into the instrument as active circuits. These circuits are dependent on the role of the instreument in the system and are additional to the normal device functions for which the instrument is primarily designed.

Basically, in any communication link there are:

Listeners:

Devices addressed to receive data. More than one listener device can be active on the bus interface at a given time.

Talkers:

Devices addressed to send data. Only one talker device can be active on the interface at a given time.

Controllers:

Instruments for addressing devices as talkers and listeners, also for sending special commands and control signals. In addition to its control function, a controller must have a talk function and, generally, a listen function.

The data byte transfer control, or "handshake" functions ensure that valid data is offered by the talker, that all listeners are ready to accept it, and also verifies that they have accepted it. The IEC interface includes other functions in addition to the above mentioned listen, talk and control functions.

NOTE: The terms, listener, talker and controller refer to selectable functions and any one device may be capable of being programmed to perform one or more of these functions. For example, a device with a listener functions is able to listen and is termed "listener" when addressed as a listener.

3.3 BUS-LINE FUNCTIONS

The 16 bus-lines are divided into three functional groups:

- 8 data lines from the data-bus for multline messages.
- 3 lines are used for data-byte transfer control (handshake lines)
- 5 lines are used for interface management.

The data bus:

The 8 bus-lines allocated for input/output data (DIO 1...8) are used for:

- Measuring data
- Programming instructions
- Status bytes
- Addresses
- Interface commands

A data byte consists of 8 parallel data bits. Where more information is needed, a complete message may comprise several data bytes in series. The maximum rate of data transfer over this two-directional asynchronous bus-system is 1 Mbyte per second.

The handshake-bus:

These 3 lines control the exchange of data bytes between devices. The lines are briefly defined as:

DAV DAta Valid
(controlled by the source)

NRFD Not Ready For Data
(controlled by the acceptor)
NDAC Not Data ACcepted
(controlled by the acceptor)

These handshake lines are activated to provide the necessary control whenever data bytes are sent over the 8 data-bus lines. They ensure that a source (generally a talker) regulates its speed to that of the slowest acceptor (generally a listener). During the transfer of a byte, the source handshake function in the interface section of a talker is active, and the acceptor handshake functions of listener are also active.

The interface management-bus:

The five lines of this group each have a specific control function between the controller and the other instruments in the system. Briefly the functions are as follows:

REN Remote ENable

A system controller facility to enable instruments to be switched between local (front panel) control and remote control.

ATN ATtentioN

A controller facility to select eihter interface messages or device-dependent messages.

IFC InterFace Clear

A system controller facility to set the interface in a predefined state or to take over control in a multiple controller system.

SRQ Service ReQuest

A device function to ask for attention from the controller.

EOI End Or Identify

Used by the talker to indicate the end of a multiple-bytes transfer

if ATN = 0, or used by a controller to obtain response for parallel polling with ATN = 1 (identify message IDY = 1).

3.4 ADDRESSING

For efficient operation, devices must be properly identified at any given time as either a data source (talker) or a data acceptor (listener). Each device is therfore given a coded address as a means of recognition by the active controller. An address consists of 7 bits.

DIO lines	8	7	6	5	4	3	2	1
Address bits	x	b7	b6	b5	b4	b3	b2	b1
Talk address	x	1	0	----------				
Listen address	x	0	1	Device address				

The address of a device is determined by the position of the five address switches that select b5 to b1 (see ISO-code table, section 3.12 of this chapter). Switches b5 to b1 may

not all be switched to 1 (address 31) because it means unlisten. It should also be noted that not all addresses are available, as some are already allocated to controllers. A controller addresses a particular instrument by placing the addrss code of that instrument on the 8 data lines and, at the same time, making ATN "true" (ATN = 1). All devices on the bus compare the address code with their own address by using the message decoding function (part of the IEC interface in the device). The device having the same address as that on the data-bus performs the function of talker or listener, as defined by the address bits b6 and b7 of the address code. Therfore, for identification purpose it is necessary for instruments to have different addresses.

3.5 ADDRESS SETTING

The correct oscilloscope address can be set by performing the steps described in the operating manual chapter 4.2.12

At delivery, or after switching-on an instrument without a memory back-up battery, the oscilloscope is set to device address 8, listen and talk (addressed).

3.6 SRQ-SERVICE REQUEST AND SERIAL POLLING

Any device in a system can use the service request line SRQ to ask for service by the controller, even when the data-bus lines are otherwise occupied. An instrument that is in an error- or alarm-condition may request service by sending the SRQ message. Furthermore, some device activities may take several seconds and it may not be economical to block the whole system until these activities are completed. By using the SRQ facility, it is possible for the controller to continue with other activities and to interupt these when the time consuming operation is finished. When the SRQ signal is activated by one of the devices, the controller can interrupt all other activities and attend to the device which is requesting for service. First of all it must detect which device gave the SRQ signal.

This is done by means of the serial poll facility. As an alternative to interruption, periodic checking of the SRQ line by the controller program is also possible. If the service request facility is not implemented, only periodic polling of all devices remains as an alternative.

Serial polling

Up to 14 devices can give service request via the SRQ line, providing they each have a service request facility in their IEC interface. Such a device must also have a talk function with a serial poll facility; its message decoding section must be capable of decoding the bus commands SPE (Serial Poll Enable and SPD (Serial Poll Disable).

If an instrument is serial polled, it puts its status byte on the data-bus. In the status byte, D107 indicates whether an instrument has asked for service (SRQ = 1). The other bits may give additional status information regarding the instrument polled; e.g. alarm, busy, ready.

Basically, the service request and serial polling is as follows:

- A device requests for service by activating the SRQ line.

- The controller receives a service request message and starts a serial polling routine by placing the device in the serial poll mode with the SPE bus command. This universal command is received by all devices.
- The controller then addresses each device in turn as talker.
- The divice addressed as (serial polled) talker responds by setting its status byte on the data-bus.
- The particular device that has requested for service responds with an RQS "true" message (DI07 low) in its status byte.
- After the controller has inspected all status bytes, it terminates the serial poll mode with the universal bus command SPD (Serial Poll Disable) and carries out the necessary action for the devices(s) that made a service request. Note that when ATN becomes false, the last polled device becomes talker. Therefore, if necessary, the device may be unaddressed as talker by addressing another device as talker or the non-standarized interface message UNT (untalk; see ISO code table) may be sent. Service can be requested for many reasons:
- The device is in an erroneous condition; a programming error or operating error has occurred.
- The device has completed its programmed task and requires further control.
- The input buffer is full or nearly full. the sending device must stop data transfer to avoid blocking the interface, or to prevent loss of data bytes. After this "full input buffer" request, service will be requested again when the contents of the buffer falls below a certain level and new data can be enterred.
- New, valid data is available but the devices interface cannot output it; e.g. because it is not addressed as a talker.

Statusword:

The statusword reflects the status of the interface functions. However, since the interface functions can also be activated by device functions, the interface status may reflect some device condition; e.g. service reques. The protocols relating to interface status data are covered by the relevant international standards.

The stautsword can be read by programming a serial poll action.

Send UNlisten
Send SPe (Serial Poll Enable)
Make the oscilloscope TALKER
Make controller LISTENER
Read STATUS-word
Send UNTalk
Send SPD (Serial Poll Diable)

It may indicate incorrect process conditions (alarm or error status).

The device dependent bits in the STATUS-word of the oscilloscope specify the reasons for service request and/or reflect the status of the device functions.

3.7 THE DEVICE STATUS-WORD

The device status-word (8 bits) from the oscilloscope is built up as follows:

BIT#	Meaning:	
7	Exception/Message	1:Data0..Data3 is exception code 0:Data0..Data3 is message code
6	My_Requ	Service Request from 8608
5	Outrdy	Data from outputbuffer available
4	Cmdrdy	One command Executed
3	Data3	Exception/message code
2	Data2	Exception/message code
1	Data1	Exception/message code
0	Data0	Exception/message code

Exception code decoding (DATA3 ... DATA0):

0: Basic I/O System
1: Calculation Module
2: File System
3: System Kernel
4: User Module

3.8 MECHANICAL SPECIFICATIONS

The mechanical characteristics of the IEC interface are briefly outlined in this section. For more detailed information, refer to IEC Publication 625-1, Section 4 Mechanical Specifations, clauses 25 or 29.

Connectors:

The IEC-bus cables are provided with 25-pole connectors, type MIL-C-24308. A plug and receptacle combined design allows cables to be stacked together in piggy-back fashion and secured by a locking mechanism.

For the oscilloscope the American variant of the IEC system (the IEEE-bus), the 24-pole micro-ribbon, connentors(Amphenol or cinch) is used. The receptacle type connector is located on the instruments rear side.

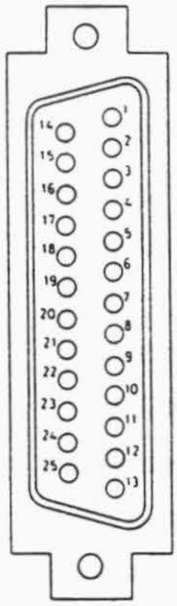
Cables:

To ensure correct system operation, the cables for the interconnection of the instrument must comply with the specification laid down in IEC-625.

The cable should contain an overall shield and at least 24 conductors, of which 16 shall be used for signal lines and the balance used for logic ground returns.

Each of the signals:DAV,NRFD,NDAC,EOI,IFC,ATN,REN and SRQ, shall be a twisted pair together with one of the logic ground wires. The maximum capacitanca existing between any signal line and all other lines connected to ground shall be 150pF per meter.

PIN CONNECTIONS:



IEC-BUS	SIGNAL LINES	IEEE-BUS
1	DIO 1	1
2	DIO 2	2
3	DIO 3	3
4	DIO 4	4
14	DIO 5	13
15	DIO 6	14
16	DIO 7	15
17	DIO 8	16
5	REN	17
6	EOI	5
7	DAV	6
8	NRFD	7
9	NDAC	8
10	IFC	9
11	SRQ	10
12	ATN	11
13	Shield	12
18	Gnd (REN)	-
19	Gnd (EOI)	-
20	Gnd (DAV)	18
21	Gnd (NRFD)	19
22	Gnd (NDAC)	20
23	Gnd (IFC)	21
24	Gnd (SRQ)	22
25	Gnd (ATN)	23
-	Gnd Logic	24

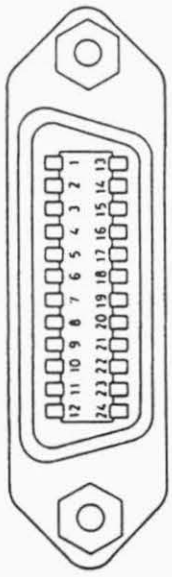


Figure 3.1 IEC-625 bus connector

Figure 3.2 IEEE-488 bus connector

3.9 SPECIFICATION IEEE-488/IEC-625 INTERFACE

Type of Interface ANSI/IEEE
Std.488-1978

-Connector	RFI/EMI shielded	Amphenol type 57LE-20240 7700D35G or similiar intermating with Amphenol 57FE series receptacles
-Bus drivers	E2	Three-state (true = 0...0,8V; false = 2...5V)
-Interfacing function repertoire		
Source handshake	SH1	Complete capability
Acceptor handshake	AH1	complete capability
Talker	T5	Basic talker:yes Serial poll:yes Talk only:yes Unadress if MLA:yes
Listener	L3	Basiclistener:yes Listen only:yes Unaddress if MTA:yes
Service request	SR1	Complete capability
Remote local	RL1	Complete capability
Parallel poll	PP0	No capability
Device clear	DC1	Complete capability
Device trigger	DT1	Complete capability
Controller	C0	No capability
Address = 0...30		Software settable
Indicator		Display in text field
Default address	8	At delivery or after switching on without backup battery

-Timing

Source: Tsc1 > 2us Tsc > 500ns Acceptor: Tac < 650ns Trd > 20ns

3.10 ISO-CODE TABLE

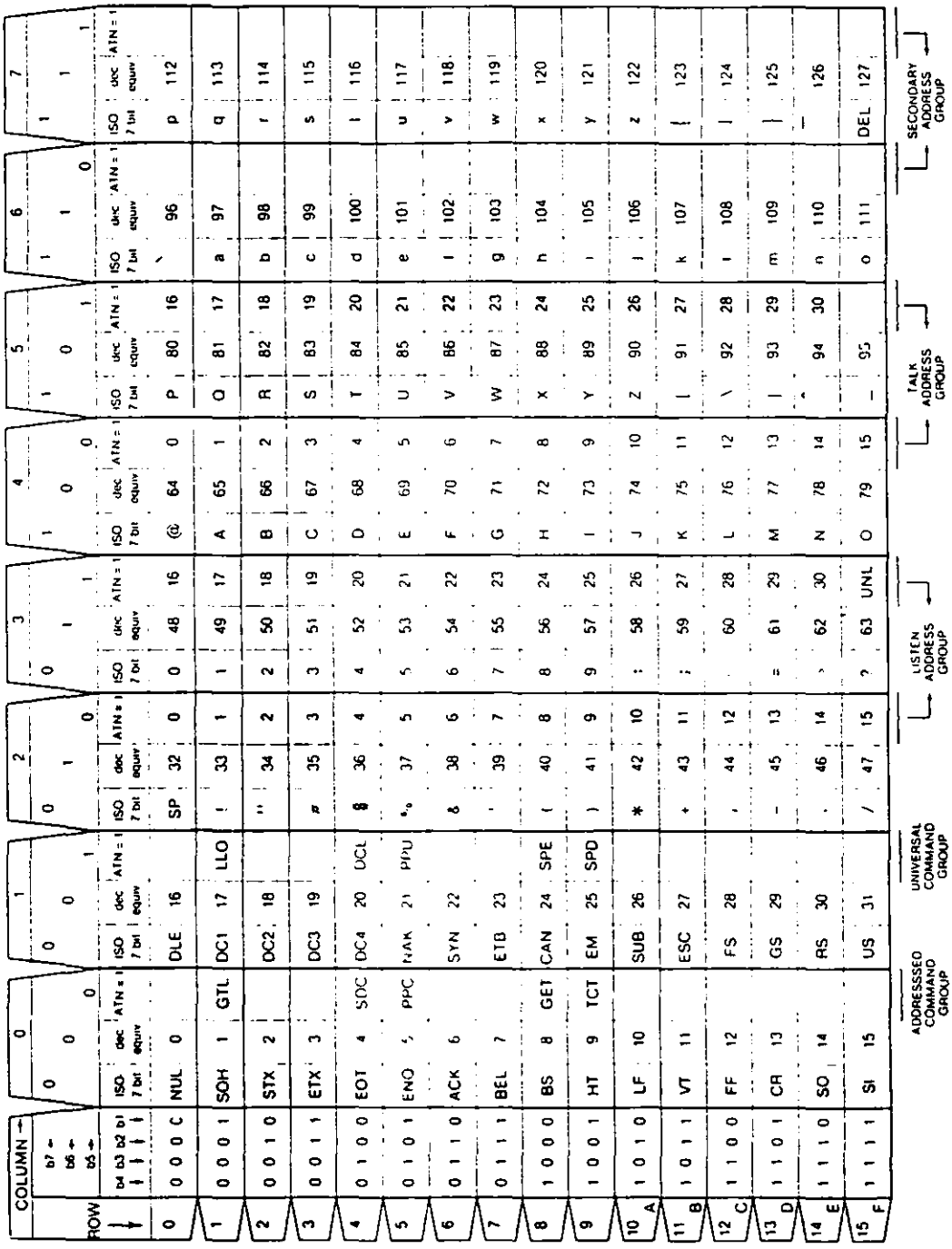


Figure 3.3 ISO-code table

3.11 MULTI LINE MESSAGES

3.11.1 GO TO LOCAL Command

There are three ways to get the oscilloscope out of the remote situation and these are:(power off,power on) and via an IEEE bus command GTL(Go To Local) or by activating the REN line.

To get the oscilloscope out of its current remote situation send:

X0000001(with ATN true)

After executing this command the oscilloscope is listening again to the frontpanel.

3.11.2 GROUP EXECUTE TRIGGER Command

It is possible for the controller to tell the oscilloscope when to start a measuremant by using the IEEE bus GET(Group Executive Trigger) command.

This is done by sending:

X0001000 (with ATN true)

Since there are two modes of measurement(recurrent,single) there are two different interpretations possible for the oscilloscope:

- Recurrent:no interpretation, measuremants are continous
- Single:re-arm the single shot mode

For these modes there will be a service request after the measurement is ready.This service request will stay active until the status of the oscilloscope is loaded into the controller.

After this, the service request is resetted by the oscilloscope and can be enabled by a new GET command

3.11.3 SELECTIVE DEVICE CLEAR COMMAND

The oscilloscope is capable of receiving and executing an SDC (Selective Device Clear) command from the IEEE bus.

The controller sets the selected oscilloscope in the selected device clear state by sending:

X0000100 (with ATN true)

On receiving this command the oscilloscope will execute the following:

- A soft restart like after power on.

3.11.4 DEVICE CLEAR Command

The oscilloscope is capable of receiving and executing a DCL (Device CLear) command from the IEEE bus.

The controller sets all the devices in the system in the device clear state by sending:

X0010100 (with ATN true)

On receiving this command the oscilloscope will execute the following.

- A soft restart like after power on.

4 RS232-C/V24 INTERFACE

4.1 INTRODUCTION

The serial interface described in this section is primarily intended to interconnect measuring instruments with other instruments to form a measuring system. A number of disciplines are already laid down for serial interfaces, but hese disciplines mainly relate to interchanges between data terminal equipment (DTE) and data communication equipment (DCE). For measuring systems, the serial interfaces referred to must be regarded as connecting data terminal equipment to another DTE.

4.2 DEFINITION OF THE RS232-C INTERFACE.

The V24 interface is based on the CCITT-Standard V24,28 giving specified signal characteristics for connecting data terminal equipment (DTE) and data communication equipment (DCE). V24 gives the functional specification of the circuits, whereas V28 specifies the electrical compatibility. The standard ISO 2110 assigns connector pin numbers to the circuits. All these documents are covered by the American Standard EIA-RS232-C exept for the interchange circuit identification. The mode of data transfer is digital, using the bit serial, byte serial method, over one signal line with common return.

4.3 DATA TRANSMISSION.

For efficient data transfer, the following characteristics must be considered.

4.3.1 Synchronization

To enable correct detection of characteristics recieved, some synchronization between transmitter and receiver is necessary. This is achieved by adding framing information to the data (ISO 1177). For the oscilloscope asynchronous data transfer is used.

Asychronous formatting is mostly used for measuring systems; common for low speed applications. It adds framing information bits to each data character.

Framing information is one start bit preceeding a data character and one or two stop bits following it.

Interfaces operating at speed up to 1200 baud normally use two stop bits; those operating above this speed normally use one stop bit. In the "standby" state, when no characters are ready to be transmitted the transmit line is held in the logical "1" state.

4.3.2 Character length

Different character lengths are possible. In measuring systems, 7 or 8 bit lengths are typical and are often switch-selectable. The character length excludes parity, start and stop bits. Normally, the ISO 7 bits code (ASCII equivalent) is used, the 8th bit being used for the parity. The least significant bit (LSB) is sent first. The total frame length does not exceed 10 bits.

4.3.3 Baudrate

The speed of the data transmission is specified in bits/sec by the baud rate, which must be selected to apply both to the transmitter and the receiver. Baudrates in use are:

50, 75, 110 , 150, 300, 600, 1200, 2400, 4800, and 9600 baud(bits/sec).

4.3.4 Interface transmission modes

A serial interface is said to operate in a simplex mode or a duplex mode depending on whether it can handle data transfer in one or both directions.

This interface handles data transfer in both directions; i.e. it can transmit and receive (Full Duplex mode).

Full duplex means that:

- The interface can handle data transfer in both directions simultaneously.
- Transmitted data is assumed to be returned via the receive line (echoing).

4.4 INITIAL SETTING

The correct data transmission parameters can be set by performing the steps described in the operating manual, chapter 4.2.12.2.

4.5 SPECIAL INTERFACE FUNCTIONS

This section deals with a number of protocols that are not standarized in V24,28 documents, but are applied to instruments equipped with a serial interface.

4.5.1 Service request and serial polling

The service request and serial poll protocol were originally developped and intended for IEC interfaces, but its application has been extended to serial interfaces. This section deals with the implementation of the protocol for serial interfaces in TRACE test and measuring instruments.

Serial interfaces do not provide a dedicated service request interrupt line. This means that the facilities for service request in devices equipped with these interfaces are somewhat restricted. If a not masked reason for service request exists, the RQS-bit in the status byte indicates that service is required; if the reason to request for service is masked,

the RQS-bit is not set. In any case, the reason may be specified with special commands (e.g.ESQ%).

A controller operating with devices via serial interfaces can periodically poll the devices to check whether service is reqiried or not. The controller executes a serial poll by sending the interface message ESC7(1B,37) to a device. Upon receipt of such a poll command, the device will respond by transmitting the ASCII equivalent of the status byte.

4.5.2 Remote local protocol

This protocol was also originally intended for application with IEC interfaces; but extended to serial interfaces. This section deals with the implementation for serial interfaces for TRACE test and measuring instruments.

The following possibilities exist for devices equipped with a serial interface and provided with a remote local interface function.

Local to Remote:

A transfer from the local to the remote state can only be performed by a controller by sending the ifnterface message ESC2 (1B,32).

Upon receipt of this "Go to remote" message, an instrument will unconditionally go into the remote state.

Remote to Local:

A transfer from the remote to the local state can be performed either by the controller or by the device in the following ways:

The controller sends:
ESC1 (1B,31) go to local or ESC3 (1B, 33) go to local.
After power on, the remote local function is always in the local state.

4.5.3 Device clear

Instruments equipped with a serial interface and provided with a device clear function, execute this function on receipt of the interface message ESC4(1B,34) i.e. device clear. The device clear will cause a soft restart as after power on.

4.5.4 Device trigger

This function is executed on receipt of the interface message ESC8 (1B, 38) i.e. device trigger.

The group execute trigger command is equivalent to the command WRT$ = "WRITE"

4.6 PIN AND CIRCUIT ALLOCATION

Pin No.	CCITT EQU.	INTERCH. CIRCUIT	CCITT-V24 description for DTE-DCE circuit	Cable connection DTE	DTE
1	101	AA	Protective ground		
2	103	BA	Transmit data(TXD)		
3	104	BB	Receive data(RXD)		
4	105	CA	Request to send(RTS)		
5	106	CB	Clear to send(CTS)		
6	107	CC	Data set ready(DSR)		
7	102	AB	Signal ground		
8	109	CF	Rec.Line Signal Det.(DCD)		
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20	108.2	CD	Data terminal rdy(DTR)		
21					
22					
23					
24					
25					

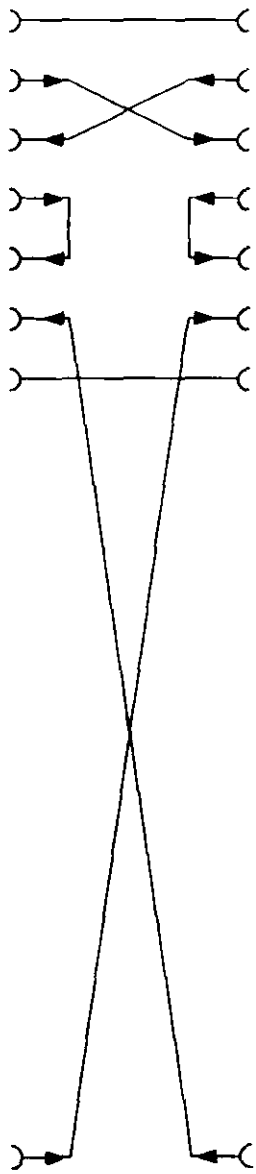


Figure 4.1 Pin and circuit allocation

4.7 ELECTRICAL SPECIFICATIONS

Bus driver requirements:

DATD (TXD,RXD)

Spacing "0" > = 3V

Marking "1" < = -3V

CONTROL (RTS; CTS; DSR; DTR)

ON > = 3V

OFF < = -3V

Current output < = 10mA

Output (driver):

Line voltage (Vo) of output -10V < = Vo < = 10V

Impedance: 300 Ohm

Input (Terminator):

Line voltage (Vi) of input -25V < = Vi < = 25V

Impedance 3000 < = Ri < = 7000Ohm

4.8 MECHANICAL SPECIFICATIONS

For RS232-C interfaces a 25-pole connector is used

- Connector requirements
- The instrument is provided with a plug type connector (female)
- The cable is provided with receptable connectors (male)
- The number of contact pins of the connector is 25
- Locking screws are provided to enable cable mounting
- The connector meeets the military specification MILC-24308 or equivalent

Cable requirements:

- The cable shall be as short as possible
- The cable length may not exceed 15 meters. However, where a longer cable is required, the total capacitance may not exceed 2500pF.
- The cable will be a "null modem" cable. A "null modem" cable implies that the wire links between the pins needed to connect two DTEs are provided within teh cable (e.g. pin 2 connected to pin 3 of the other terminal)

4.9 RS232-C CONNECTOR

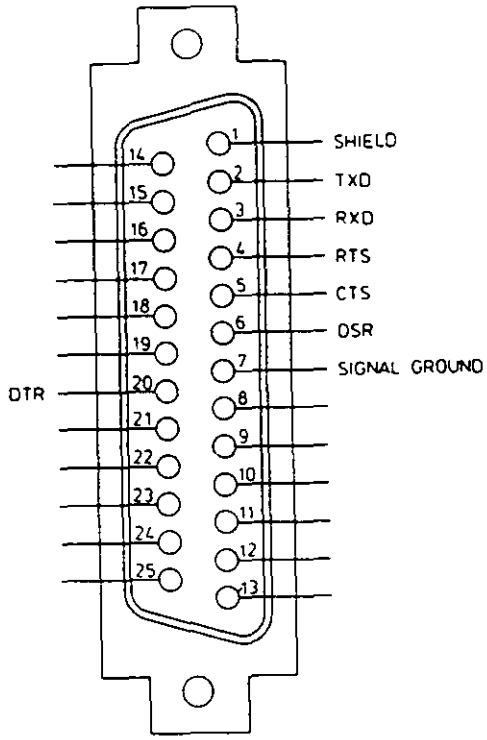


Figure 4.6 RS232-C connector

5 PROGRAMMING THE OSCILLOSCOPE

To achieve scope like operation, an interactive user may read and set parameters in volts per division or seconds per division.

As in digital storage scopes with raster scan display this parameters are not only dependend from channel adjustement but also from a zoom factor, volt- and seconds per division are in fact a mixture out of two parameters.

In interactive mode you pay with some restrictions in use of these two parameters.

In remote mode, where a user wants to generate synthetic traces this restrictions are unacceptable.

That's why programming of the oscilloscope is in some actions closer to the hardware than interactive operation.

The attenuator is set in full scale values, the time base is specified in samples per second, triggerlevel is defined as number of LSBs and delay has to be set in number of samples.

5.1 MESSAGE PROTOCOL FOR OSCILLOSCOPES

5.1.1 Separators

The programming language is line oriented. One line consists of a number of commands and is terminated with a LINE TERMINATOR string. The default LINE TERMINATOR is a CR (Carriage Return, Hex 0D, Dec 13). Commands are separated by a COMMAND SEPARATOR, which is always the colon (:).

5.1.2 Commands

Commands consist of a codeword and a body.

5.1.3 Lines

A line may have none, one or more commands up to a maximum of 256 bytes including the line terminator string.

5.1.4 Numeric Representations

Numerical data can be represented in real or integer notation.

6 PROGRAMMING CODES

A command name may have up to 5 characters, a variable name may have a maximum of 4 characters and one type identifier character.

Example:XYZt 't' is one of [% | ! | $]

- type identifier for 16 bit integer variables:%
- type identifier for 32 bit integer variables:%
- type identifier for 32 bit real variables:!
- type identifier for string variables:$

If a variable is specified without a type identifier, real is assumed. However it is recommended to use always the type identifier, as the interpreter is faster when trying to look up the variable string.

In general, all variables with a discrete set of values are string variables, even if the strings are numeric. This strategy avoids rounding and calculation precision problems. The disadvantage is, you may not directly use this variable in numeric expressions.

Variables with a continuous set of values, even if the range is restricted, are integer or real typed. Refer to TRL% as example.

An exception from this rules are ATT! and SAM! variables that may represent a discrete set of real values only.

6.1 SYSTEM CODES

6.1.1 Command Separator

Command separator is always the colon ':'.It is not programmable.

6.1.2 Line Separator

The line separator may be read and set. After restart it is set to <CR> by default. You may program for RS232 and IEEE interface one value each for input and one for output. Since line terminators are usually nonprintable characters, this variables are of type integer.

LSI% .. Line Separator for RS232 input may be set and read

Example programs for input:
LSI% = 13

LSO% .. Line Separator for RS232 output may be set and read

Example programs for output:
LSO% = 10

LII% .. Line Separator for IEEE input may be set and read

Example programs for input:
LSI% = 13

LIO% .. Line Separator for IEEE output may be set and read

Example programs for output:
LSO% = 10

6.1.3 Call for Identity

For unit identification, user may issue several identity commands.

TYP$ .. Type of unit may be read.

Example reads type code:
? TYP$
Answer is:
8608A

VER$ .. Version number of software may be read.

Example reads version number:
? VER$
Answer may be if Version 1.12:
V 1.12

SER$ .. Serial number of unit may be read.

Example reads serial number:
? SER$
Answer may be if serial number 600:
600

6.1.4 Service Request Codes

Command service request

CSQ% .. To synchronize on command execution, command executed service request of the unit may be enabled. A service request is issued whenever the "cmdrdy" bit in the status byte is set:

Possible values are: 0 or 1
Example enables command executed service request:
CSQ% = 1

Exception service request

ESQ% .. To enable exception handling, exception service request of the unit may be enabled. A service request is issued whenever the "exc/mess" bit in the status byte is set:

Possible values are: 0 or 1
Example enables exception service request:
ESQ% = 1

Line data service request

LSQ% .. To synchronize on available data lines, line data ready service request of the unit may be enabled. A service request is issued whenever the "outrdy" bit in the status byte is set:

Possible values are: 0 or 1
Example enables line data ready service request:
LSQ% = 1

Trace data service request

TSQ% .. To synchronize on recording and calculation cycles, trace data ready service request of the unit may be enabled. A service request is issued whenever on calculation cycle is finished. (The same condition is used for TDR%):

Possible values are: 0 or 1
Example enables trace data ready service request:
TSQ% = 1

6.1.5 Exception Read Out

IEX% .. For detailed information on exception reasons, actual integer exception code may be read:

Example reads last exception code:
? IEX%
Answer may be if no exception occured:
0

IEX$ .. For detailed information on exception reasons, actual exception string may be read:

Example reads last exception code:
? IEX$
Answer may be if no exception occured:
OK

6.2 SUPER FUNCTION CODES

6.2.1 Node Selection

Channels, calculated traces and calculated scalars are looked upon as nodes in a synchronized calculation and measurement system. Node specific parameters may be accessed in two ways:

- One can specify the desired node together with the node specific command.
- You can select one node for parameter access and specify no node together with the node specific command.

As channel-, calculation and scalar nodes have typical capabilities, not all node affecting commands work on all types of nodes. Commands like CPL$ or PRO% are only applicable on channel nodes. Operations on channel nodes affect the channel adustement.

TRS$ .. Selected default node for parameter access may be read and set.

Valid node names are:
CHA | CHB | TR1 | TR2 | TR3 | TR4 | FU1 | FU2 | FU3 | FU4 | SYS
Example selects channel A:
TRS$ = "CHA"
Example reads selected node's name:
? TRS$
Answer may be if selected node is CHA:
CHA
If TR1 is defined as channel node for channel A, TR1 and CHA may be used alternatively with identical results. SYS selects system parameters instead of node parameters and is used for time read and write. (RTS$, HOU%, MIN%, ..).

6.3 SUPER FUNCTION DEPENDEND CODES

Super function dependent codes access parameters that exist in each node. You may specify the desired node together with the command or, if a node name is omitted, the selected node is accessed by default.

6.3.1 Vertical Functions

ATT% .. The attenuation range code may be read or set for easy remote programming of increment/ decrement operation. This variable is available for channel nodes only.

Possible values are: 0 .. 9
Example sets attenuator of selected node to level 5:
ATT% = 5
Example reads attenuator of selected node:

? ATT%
Answer may be, if level is 5:
5
Example sets attenuator of channel A to level 5:
ATT%("CHA") = 5
Example reads attenuator of channel A:
? ATT%("CHA")
Answer may be, if level is 5:
5

ATT! .. The full range value may be read or set to the nearest possible value. LSB of node = ATT!/0FFFFH. This variable is available for channel, trace and function nodes.

Possible values for channels with 1:1 probe are:
32E-03, 64E-03, 128E-03
Possible values for channels with 1:1 or 1:10 probe are:
320E-03, 640E-03, 1.28E + 00, 3.2E + 00, 6.4E + 00, 12.8E + 00, 32E + 00
Possible values for channels with 1:10 probe are:
64E + 00, 128E + 00, 320E + 00
Example sets attenuator of selected node to 32V:
ATT! = 32
Example reads attenuator of selected node:
? ATT!
Answer may be, if full range value is 32V:
32
Example sets attenuator of channel A to 32V:
ATT!("CHA") = 32
Example reads attenuator of channel A:
? ATT!("CHA")
Answer may be, if full range value is 32V:
32

CPL$ .. The channel node's coupling value may be read or set. This variable is available for channel nodes only.

Possible values are: AC | DC | GND
Example sets coupling of selected node to AC:
CPL$ = "AC"
Example reads coupling of selected node:
? CPL$
Answer may be, if coupling is AC:
AC
Example sets coupling of channel A to AC:
CPL$("CHA") = "AC"
Example reads coupling of channel A:
? CPL$("CHA")
Answer may be, if full range value is AC:
AC

OFF! .. The real offset value may be read or set to the nearest possible value according to available OFF% values.

OFF! = (ATT! * OFF% * 5120)/0FFFFH.
This variable is available for channel, trace and function nodes.
Example sets offset of selected node to 32V:
OFF! = 32
Example reads offset of selected node:
? OFF!
Answer may be, if offset value is 32V:
32
Example sets offset of channel A to 32V:
OFF!("CHA") = 32
Example reads offset of channel A:
? OFF!("CHA")
Answer may be, if offset value is 32V:
32

OFF% .. The offset value may be read or set to the nearest possible value. Offset step of 1 is a step of 5120. This variable is available for channel nodes only.

Possible values are:
-6,-5, ..0, ..5, 6
Example sets offset of selected node to approx.30% of full range:
OFF% = 2
Example reads offset of selected node:
? OFF%
Answer may be, if offset is approx. 30% of full range:
2
Example sets offset of channel A to approx. -90% of full range:
OFF%("CHA") = -6
Example reads offset of channel A:
? OFF%("CHA")
Answer may be, if offset is approx. 90% of full range:
-6

PRO% .. Probe factor of channel node may be read. This variable is available for channel nodes only:

Example reads selected channel's probe factor:
? PRO%
Answer may be if probe is 10:1
10
Example reads selected channel's probe factor:
? PRO%("CHA")
Answer may be if probe is 100:1
100

6.3.2 Horizontal Functions

DLY! .. The delay value may be read or set to the nearest possible value according to:
DLY! = DLY% * SAM!
This variable is available for channel and trace nodes.
For negative delay values the values are continuous but for positive delay values only multiples of 40 * sample rate may be adjusted.
Example sets delay of selected node to -320us. This results in a prerecord time of 320us. The trigger line is 320us to the right from the beginning of the trace:
DLY! = -320e-3
Example sets delay of channel A to + 120us. This results in a trigger 120us before the first point is stored. The trigger line is virtually 120us to the left of the trace.:
DLY!("CHA") = 120e-6
Example reads delay of channel A:
? DLY!("CHA")
Answer may be, if delay is 120us:
1.2E-04
Example reads delay of selected node:
? DLY!
Answer may be, if delay is -320us clocks:
-3.2e-04

DLY% .. The delay value may be read or set to the nearest possible value. In remote delay is the number of sample clocks between trigger and first recorded point. As with attenuator and sample rate, not all values are adjustable. For negative delay values the values are continuous but for positive delay values only multiples of 40 may be adjusted. This variable is available for channel nodes only.

Possible values are:
- 39960 (= 999 divisions) .. actual recording length
Example sets delay of selected node to -100 clocks. This results in a prerecord time of 100 * sample rate. The trigger line is 100 points to the right from the beginning of the trace:
DLY% = -100
Example sets delay of channel A to + 120 clocks. This results in a trigger 120 clocks before the first point is stored. The trigger line is virtually 120 points to the left of the trace.:
DLY%("CHA") = 120
Example reads delay of channel A:
? DLY%("CHA")
Answer may be, if delay is 120 clocks:
120
Example reads delay of selected node:
? DLY%
Answer may be, if delay is -100 clocks:
-100

SAM% .. The sample rate level may be read or set. This command may be used when programming increment decrement sequences. This variable is available for channel nodes only.

Possible values are:
0 .. 5 for sampling operation
6 .. 22 for normal sampled operation
23 .. 35 for roll operation
Example sets sample rate of selected node to level 12:
SAM% = 12
Example reads sample rate of selected node:
? SAM%
Answer may be, if level is 12:
12
Example sets sample rate of channel A to level 12:
SAM%("CHA") = 12
Example reads sample rate of channel A:
? SAM%("CHA")
Answer may be, if level is 12:
12

SAM! .. The sample rate of the selected node may be read or set to the nearest possible value. This variable is available for channel and trace nodes.

Possible recurrent equivalent values for channels are:
.25E-09, .5E-09, 1.25E-09, 2.5E-09, 5E-09, 12.5E-09
Possible recurrent sample values for channels are:
25E-09, 50E-09, 125E-09, 250E-09, 500E-09, 1.25E-06, 2.5E-06, 5E-06, 12.5E-06, 25E-06, 50E-06, 125E-06, 250E-06, 500E-06, 1.25E-03, 2.5E-03, 5.0E-03
Possible roll mode values for channels are:
12.5E-03, 25E-03, 50E-03, 125E-03, 250E-03, 500E-03, 750E-03, 1.5E+00, 3E+00, 9E+00, 22.5E+00, 45E+00, 90E+00
Possible values for calculated nodes are valid in a range of:
0 .. 8.43E-37 .. 3.38E38
Example sets sample rate of selected node to 125ns:
SAM! = 125E-09
Example reads sample rate of selected node:
? SAM!
Answer may be, if full sample interval is 125ns:
125E-09
Example sets sample rate of channel A to 125ns:
SAM!("CHA") = 125E-09
Example reads sample rate of channel A:
? SAM!("CHA")
Answer may be, if full sample interval is 125ns:
125E-09

PPT% .. The points per trace may be read or set to the nearest possible value. This command is available for trace nodes only and intended for use with memory trace nodes. Channel node's buffer length is controlled by ALY% and MXM%, length of calculated traces depends upon their operands and length of function nodes is always 1.

Possible values are:
In Hexadecimal representation:
0FFFFH, 0FFFEH, .. 1H
In decimal representation:
65535D, 65534D, .. 1D
Example sets length of selected node to 16384 samples:
PPT% = 16384
Example reads length of selected node:
? PPT%
Answer may be, if length is 5 samples:
5
Example sets length of trace 4 to 16384 samples:
PPT%("TR4") = 16384
Example reads length of trace 4:
? PPT%("TR4")
Answer may be, if length is 5 samples:
5

6.3.3 Definition Functions

ORI$ .. The origin definition of a node may be read or set. This variable is available for trace and funtion nodes.

Possible strings are:
ORI$("target node") = trace function("source node")
ORI$("target node") = trace function("source node",ii)
ORI$("target node") = trace function("source node",rr)
ORI$("target node") = trace function("source node")
ORI$("target node") = trace function("source node","source node")

Possible trace target nodes are: TR1 | TR2 | TR3 | TR4
Possible scalar function target nodes are: FU1 | FU2 | FU3 | FU4
Possible source nodes are: CHA | CHB | TR1 | TR2 | TR3 | TR4 | Mxx
Where xx is the number of a traces stored on E-Disc. The valid combinations of target nodes and source nodes are defined in 8608A operating manual.

Possible strings to define a **ONE OPERAND TRACE FUNCTION** are:

for deletion of a trace or a scalar:
OFF
for normal display of one channel or trace:
EQU ("source node")
for differentiation of one trace:

sf = 2 | 5 | 20 | 100 | 500 | 1000
DIF ("source node","sf")
for integration of one trace:
sf = 2 | 5 | 20 | 100 | 500 | 1000
INT ("source node","sf")
for negation of one trace:
NEG ("source node","source node")
for smoothing of one trace:
sf = 2 | 4 | 8 | 16 | 32 | 64
SMO ("source node","ii")
Where sf is a scaling factor string and ii is an integer scaling factor.
Possible strings to define a **TWO OPERAND TRACE FUNCTION** are:

for addition of two traces:
ADD ("source node","source node")
for division of two traces:
DIV ("source node","source node")
for multiplication of two traces:
MUL ("source node","source node")
for subtraction of two traces:
SUB ("source node","source node")

Possible strings to define a **FUNCTION NODE** are:

Simple amplitude measurements:

Cursor to Reference amplitude of trace:
CRA("source node")
Cursor to zero amplitude of trace:
CZA("source node")
50% rise amplitude of trace:
FIA("source node")
Maximum to zero amplitude of trace:
MAA("source node")
Minimum to zero amplitude of trace:
MIA("source node")
Peak to peak amplitude of trace:
PPA("source node")
10% to 90% rise amplitude of trace:
RIA("source node")
Reference to zero amplitude of trace:
RZA("source node")

Integral amplitude measurements:

DC to zero amplitude of trace:
DCA(" source node")
Mean to zero amplitude of trace:
MEA("source node")
Root mean square amplitude of trace:
RMS("source node")

Simple amplitude ratio measurements:

Cursor amplitude of left trace to reference amplitude of right trace linear:
CRI("source node","source node")
Cursor amplitude of left trace to reference amplitude of right trace logar.:
CRO("source node","source node")

Integral amplitude ratio measurements:

Root mean square amplitude of left trace to rout
mean square amplitude of right trace linear:
RMI("source node","source node")
Root mean square amplitude of left trace to rout
mean square amplitude of right trace logar:
RMO("source node","source node")

Simple time measurements:

Cursor to reference time of trace:
CRT("source node")
Cursor to trigger time of trace:
CTT("source node")
Reference to trigger time of trace:
RTT("source node")
10% to 90% time of trace:
RIT("source node")

Integral time measurements:

Frequency of trace:
FRQ("source node")
Period time of trace:
PER("source node")

Simple time ratio measurements:

Phase difference time of left trace to right trace:
PHT("source node","source node")

Integral time ratio measurements:

Phase difference degrees of left trace to right trace:
PHD("source node","source node")
Example sets origin of selected node to channel a:
ORI$ = EQU("CHA")
Example reads origin of selected node:
? ORI$
Answer may be, if origin is CHA + CHB:
ADD("CHA","CHB")
Example sets origin of node TR1:
ORI$("TR1") = SUB("CHA","M01")
Example reads origin of node FU2:
? ORI$("FU2")

Answer may be, if origin is cursor - reference of TR3:
CRT("TR3")

6.3.4 Data Functions

TRA! .. The real value of a node may be read or set. This variable is available for trace and function nodes. For function nodes only read access is possible. Write access is in fact useful for memory trace nodes only.

Example sets n-th element of selected node to 65.536V.
n may be in the range 0 .. PPT% (node length):
TRA!(,n) = 65.536
Example reads n-th element of selected node:
? TRA!(,n)
Answer may be, if real value is 65.536V:
65.536
Example sets n-th element of channel A to 65.536V:
TRA!("CHA",n) = 65.536
Example reads n-th element of channel A:
? TRA!("CHA",n)
Answer may be, if real value is 65.536V:
65.536

TRA% .. The integer value of a node may be read or set. This variable is available for trace and function nodes. For function nodes only read access is possible. Write access is in fact useful for memory trace nodes only.

Example sets n-th element of selected node to 32767.
n may be in the range 0 .. PPT% (node length):
TRA%(,n) = 32767
Example reads n-th element of selected node:
? TRA%(,n)
Answer may be, if integer value is 32767:
32767
Example sets n-th element of channel A to 32767:
TRA%("CHA",n) = 32767
Example reads n-th element of channel A:
? TRA%("CHA",n)
Answer may be, if integer value is 32767:
32767

6.3.5 Time Functions

RTS$ .. Actual real time and date status may be read or set. This variable is available for channel, trace and function nodes and for the system.

Posible values are:
ENTER or STOP
Example sets real time status:
RTS$("SYS") = "ENTER"

Example reads real time status:
? RTS$("SYS")
Answer may be:
ENTER
The RTS$ = \STOP\ command saves the actual system or trace node time and date in an internal buffer. You may then read it by means of the commands ? HOU%, ? MIN%, .. or set parts or all of it by means of the commands HOU% = 12, MIN% = 40, ... The modified time and date may then be rewritten to the real time clock or a trace node using RTS$ = "ENTER". If you modify only parts of time and date without preceding RTS$ = "STOP" command, the remaining items are undefined. The recording time of channel or calculated nodes is updated after each calculation and recording cycle.
The following sequence sets time and date of real time clock to 26-Nov-1987, 12:45:00 :
CEN% = 19:YEA% = 87:MON% = 10:DAY% = 26:HOU% = 12:
MIN% = 45:SEC% = 0
RTS$("SYS") = \ENTER\
The next sequence copies time and date from real time clock to all traces:
RTS$("SYS") = "STOP":RTS$("TR1") = "ENTER":RTS$("TR2") = "ENTER"
RTS$("TR3") = "ENTER":RTS$("TR4") = "ENTER"
The last sequence reads recording time of trace 1:
RTS$("TR1") = "STOP"
? CEN%,YEA%,MON%,DAY%,HOU%,MIN%,SEC%

6.4 Super Function independend Codes

6.4.1 Channel Functions

ALY% .. The channel A only status may be read or set.

Possible values are:
0 | 1
Example sets channel A only status to on:
ALY% = 1
Example reads channel A only status:
? ALY%
Answer may be, if channel A only status is on:
1

AVC$ .. The average count may be read or set.

Possible values are:
4 | 16 | 64 | 256 | 1024 | 4096 | 8192
Example sets average count to 16:
AVC$ = "16"
Example reads average count:
? AVC$
Answer may be, if average count is 16:
16

AVS% .. The average status may be read or set

Possible values are:
0 or 1
Example sets average status to on:
AVS% = 1
Example reads average status:
? AVS%
Answer may be, if average status is on:
1

AVM$ .. The average mode may be read or set

Possible values are:
CONT or BLOCK
Example sets average mode to continous:
AVM$ = "CONT"
Example reads average mode:
? AVM$
Answer may be, if average mode is continous:
CONT

BWL% .. The band with limit status may be read or set.

Possible values are:
0 or 1

Example sets band with limit status to on:
BWL% = 1
Example reads band with limit status:
? BWL%
Answer may be, if band with limit status is on:
1

GLD% .. The MIN/MAX dedection status may be read or set.

Possible values are:
0 or 1
Example sets MIN/MAX dedection status to on:
GLD% = 1
Example reads MIN/MAX dedection status:
? GLD%
Answer may be, if MIN/MAX dedection status is on:
1

MXM% .. The maximum memory status may be read or set.

Possible values are:
0 or 1
Example sets maximum memory status to on:
MXM% = 1
Example reads maximum memory status:
? MXM%
Answer may be, if maximum memory status is on:
1

6.4.2 Sequence Functions

CAL% .. The autocal function may be started.
Possible values are:
0 or 1
Example sets autocal function to on:
CAL% = 1

MOD$ .. The recording mode may be read or set.

Possible values are:
RECURRENT | SINGLE | ROLL
Example sets recording mode to single shot:
MOD$ = "SINGLE"
Example reads recording mode:
? MOD$
Answer may be, if recording mode is single shot:
SINGLE

REL% .. The recording release status may be read or set.

Possible values are:

0 or 1
Example sets recording release status to on:
REL% = 1

SET% .. The autoset function may be set.

Possible values are:
0 or 1
Example starts autoset function:
SET% = 1

TDR% .. To synchronize on recording and calculation cycles, tracedata ready status may be read or set. The same condition is used for TSQ%:

Possible values are:
0 or 1
Example sets all node's data status to old:
TDR% = 0
Example reads all node's data status:
? TDR%
Answer may be, if node's data status is new:
1

WRT$ .. The recording start/stop status may be read or set.

Possible values are:
WRITE | LOCK
Example sets recording start/stop status to on:
WRT$ = "WRITE"
Example reads recording start/stop status:
? WRT$
Answer may be, if recording start/stop status is on:
WRITE

6.4.3 Trigger Functions

ATR% .. The autotrigger function may be read or set. In recurrent and single mode, autotrigger function on releases a trigger signal after a timeout. In roll mode, autotrigger funtion off results in an endless recording

Possible values are:
0 or 1
Example sets autotrigger function to on:
ATR% = 1
Example reads autotrigger function:
? ATR%
Answer may be, if autotrigger function is on:
1

ECP$ .. The extern trigger coupling may be read or set.

Possible values are:

AC | DC | LF | HF | AC_HF | TVF
Example sets extern trigger coupling to tv frame:
ECP$ = "TVF"
Example reads extern trigger coupling:
? ECP$
Answer may be, if extern trigger coupling is tv frame:
TVF

TRL% .. The trigger level value may be read or set to the nearest possible value

Possible values are:
In Hexadecimal representation:
07F00H, 07E00H, .. 0100H, 0, 0FF00H, .. 08100H, 08000H
In decimal representation:
32512D, 32256D, .. 256D, 0, -256H, ..-32512D,-32768D
Example sets trigger level of selected node to + 1/4 full range:
TRL% = 16384
Example reads trigger level of selected node:
? TRL%
Answer may be, if trigger level is + 1/4 full range:
16384
Example sets trigger level of channel A to -1/2 full range:
TRL%("CHA") = -32768
Example reads trigger level of channel A:
? TRL%("CHA")
Answer may be, if trigger level is -1/2 full range:
-32768

TGH$ .. The trigger hysteresis may be read or set.

Possible values are:
1% | 2% | 5% | 10% | 20% | 50% |
Example sets trigger hysteresis to 10 percent of full range
TGH$ = "10%"
Example reads actual trigger hysteresis:
? TGH$
Answer may be, if trigger hysteresis is 10 percent of full range
10%

TGS$ .. The trigger source may be read or set.

Possible values are:
CHA | CHB | EXT | EXT_10| LINE
Example sets trigger source to ext:
TGS$ = "EXT"
Example reads trigger source:
? TGS$
Answer may be, if trigger source is ext:
EXT

TSL% .. The trigger slope may be read or set.

Possible values are:
0 | 1
Example sets trigger slope to rising slope:
TSL% = 1
Example reads trigger slope:
? TSL%
Answer may be, if trigger slope is rising:
1

6.4.4 Display Functions

D12$ .. The display representation of TR1 and TR2 may be read or set

Possible values are:
YT or XY
Example sets display representation of TR1 and TR2 to xy:
D12$ = "XY"
Example reads display representation of TR1 and TR2:
? D12$
Answer may be, if display representation of TR1 and TR2is xy:
XY

D34$ .. The display representation of TR3 and TR4 may be read or set

Possible values are:
YT or XY
Example sets display representation of TR3 and TR4 to xy:
D34$ = "XY"
Example reads display representation of TR3 and TR4:
? D34$
Answer may be, if display representation of TR3 and TR4is xy:
XY

DOT$ .. The interpolation function may be read or set.

Possible values are:
LIN | PLS | SIN | OFF
Example sets interpolation function to linear:
DOT$ = "LIN"
Example reads interpolation function:
? DOT$
Answer may be, if interpolation function is linear:
LIN

EXP$ .. The x-expand function may be read or set.

Possible values are:
0.05 | 0.1 | 1 | 10
Example sets x-expand function to *10:
EXP$ = "10"

Example reads x-expand function:
? EXP$
Answer may be, if x-expand function is *10:
10

ILL% .. The grid illumination may be read or set.

Possible values are:
0 or 1
Example sets grid illumination to on:
ILL% = 1
Example reads grid illumination:
? ILL%
Answer may be, if grid illumination is on:
1

SEP% .. The display traces separated function may be read or set.

Possible values are:
0 or 1
Example sets display traces separated function to on:
SEP% = 1
Example reads display traces separated function:
? SEP%
Answer may be, if display traces separated function is on:
1

XPO% .. The trace screen position may be read or set

It depends on zoom and trace length
Example set trace screen position to 1:
XPO% = 1
Example reads trace screen position:
? XPO%
Answer may be, if trace screen position is 1:
1

6.4.5 Cursor Functions

CUR% .. The cursor screen position may be read or set.

It depends on zoom and trace length
Example set cursor screen position to 1:
CUR% = 1
Example reads cursor screen position:
? CUR%
Answer may be, if cursor screen position is 1:
1

REF% .. The reference screen position may be read or set.

It depends on zoom and trace length

Example set reference screen position to 1:
REF% = 1
Example reads reference screen position:
? REF%
Answer may be, if reference screen position is 1:
1

TRK% .. The cursor and reference may be shifted by a given value which depends on zoom and trace length

Example moves both cursor and reference ten data point to the left:
TRK% = -10

6.4.6 File Functions

Filenames:

"ALL" fixed name identifying the current system state (setup + all existing traces)

"SET" fixed name identifying the current system setup

"CHx" the string "CH" followed by either an 'A' or 'B' identifying one channel

"TRn" the string "TR" followed by an digit in the range '1'..'4' identifying one trace

"IEEE" fixed name identifying the IEEE interface channel

"RS232" fixed name identifying the RS232 interface channel

"Ann" the character "A" followed by two digits in the range of "00" .. "99" identifying one 'ALL'- file on E-disk

"Mnn" the character "M" followed by two digits in the range of "00" .. "99" identifying one 'TRACE'- file on E-disk

"Snn" the character "S" followed by two digits in the range of "00" .. "99" identifying one 'SETUP'- file on E-disk

Note that not all filenames are valid in all combinations.

Functions (FUn) are NOT accessible as files, but via their TRA%, TRA! etc. values.

Valid Filenames for KILL:

"Ann" "Mnn" "Snn" TRn"

Valid Filenames for COPY: The following table shows the valid combinations of filenames for use with the copy command. + indicates a valid combination, - indicates an invalid combination.

from:	ALL	SET	CHx	TRn	IEEE	RS232	Axx	Mxx	Sxx
to:									
ALL	-	-	-	-	+	+	+	-	-
SET	-	-	-	-	+	+	-	-	+
CHx	-	-	-	-	-	-	-	-	-
TRn	-	-	-	-	-	-	-	-	-
IEEE	+	+	+	+	-	-	+	+	+
RS232	+	+	+	+	-	-	+	+	+
Axx	+	-	-	-	+	+	+	-	-
Mxx	-	-	+	+	+	+	-	+	-
Sxx	-	+	-	-	+	+	-	-	+

Note that any COPY checks the data for correct syntax and framing, so for example it is NOT possible to copy random data to any E-disk file; copying to a Mxx file requires the incoming data to have the correct syntax of a single trace file.

COPY .. Copy source data file or interface to destination data file or interface.

Example copies node TR1 to E-disc trace file M01
COPY "TR1" TO "M01"
COPY "TR1" "M01"

KILL .. The data file is deleted.

Example copies node TR1 to E-disc trace file M01
KILL "M01"

CPF$ .. The file copy format may be read or set.

Possible values are:
ASCII_HEX | BINARY
The file format on E-disc is the normal binary byte value as it is read from memory. An attempt to copy such files via serial interface with activated xon/xoff protocoll leads into troubles as all bytes with values 17 and 19 decimal are interpreted as handshake control characters. To overcome this problem, serial interfaces have to use ascii coded characters and digits. The disadvantage of this strategy is the multiple number of bytes to transfer. To enable the user to select his proper format, CPF$ variable is available. In "BINARY" mode, all copies via interface RS232 and IEEE are performed with minimal byte number and maximum speed. This mode is suitable for IEEE transfers and serial transfers using modem control only.

The transfer is terminated by EOI on IEEE and on block- and byte- count on RS232.
For serial transfers using xon/xoff the "ASCII_HEX" mode is available, where each transferred byte is an ascii character in the range of '0'..'9' and 'A'..'F', interpreted as the nibble of a byte, low order nibble first.
The transfer is terminated by ^Z (90 dezimal).
Example sets file copy format to ASCII-hex:
CPF$ = "ASCII_HEX"
Example reads file copy format:
? CPF$
Answer may be, if file copy format is binary:
BINARY

6.4.7 IEEE Interface Functions

ADD% .. The IEEE address value may be read or set.

Possible values are:
0 .. 30
Example sets IEEE address value to 8:
ADD% = 8
Example reads IEEE address value:
? ADD%
Answer may be, if IEEE address value is 8:
8

TLM$ .. The IEEE talk/listen mode may be read or set.

Possible values are:
TO | TL | LO
Example sets talk/listen mode to talk only:
TLM$ = "TO"
Example sets talk/listen mode to listen only:
TLM$ = "LO"
Example reads talk/listen mode:
? TLM$
Answer may be, if talk/listen mode is talk/listen:
TL

6.4.8 Serial Interface Functions

BAU$ .. Actual serial baudrate may be read or set.

Posible values are:
50 | 75 | 110 | 150 | 300 | 600 | 1200 | 2400 |
4800 | 9600
Example sets serial baudrate:
BAU$ = "9600"
Example reads serial baudrate:
? BAU$

Answer may be:
9600

DAT$ .. The serial data bits may be read or set.

Possible values are:
7 or 8
Example sets serial data bits to 7:
DAT$ = "7"
Example reads serial data bits:
? DAT$
Answer may be:
7

MCT% .. The serial modem control may be read or set.

Possible values are:
0 or 1
Example sets serial modem control to on:
MCT% = 1
Example reads serial modem control:
? MCT%
Answer may be:
1

PAR$ .. The serial parity may be read or set.

Possible values are:
ODD | EVEN | NO
Example sets serial parity to odd:
PAR$ = "ODD"
Example reads serial parity:
? PAR$
Answer may be:
ODD

STO$ .. The serial stop bits may be read or set

Possible values are:
1 or 2
Example sets serial stop bits to 1:
STO$ = "1"
Example reads serial stop bits:
? STO$
Answer may be:
1

XON% .. The serial xon/xoff protocol may be read or set.

Possible values are:
0 or 1
Example sets serial xon/xoff protocol to on:

XON% = 1
Example reads serial xon/xoff protocol:
? XON%
Answer may be:
1

6.4.9 Plotter Interface Functions.

PIN$ .. Actual plot interface may be read or set.

Posible values are:
IEEE or RS232
Example sets plot interface:
PIN$ = "IEEE"
Example reads plot interface:
? PIN$
Answer may be:
IEEE

PLA$ .. Actual plot language may be read or set.

Posible values are:
HPGL or STANDARD
Example sets plot language:
PLA$ = "HPGL"
Example reads plot language:
? PLA$
Answer may be:
HPGL

PLT% .. The plotter screen dump status may be read or set.

Possible values are:
0 or 1
Example sets plotter screen dump status to on:
PLT% = 1
Example reads plotter screen dump status:
? PLT%
Answer may be, if plotter screen dump status is on:
1

PSI$ .. Actual plot size may be read or set.

Posible values are:
A3 or A4
Example sets plot size:
PSI$ = "A3"
Example reads plot size:
? PSI$
Answer may be:
A3

6.4.10 Time Functions

The RTS$ = \STOP\ command saves the actual system or trace node time and date in an internal buffer. You may then read it by means of the commands ? HOU%, ? MIN%, .. or set parts or all of it by means of the commands HOU% = 12, MIN% = 40, ... The modified time and date may then be rewritten to the real time clock or a trace node using RTS$ = "ENTER". If you modify only parts of time and date without preceding RTS$ = "STOP" command, the remaining items are undefined.

The recording time of channel or calculated nodes is updated after each calculation and recording cycle.

The following sequence sets time and date of real time clock to 26-Nov-1987, 12:45:00

CEN% = 19:YEA% = 87:MON% = 10:DAY% = 26:HOU% = 12:MIN% = 45:SEC% = 0

RTS$("SYS") = \ENTER\

The next sequence copies time and date from real time clock to all traces:

RTS$("SYS") = "STOP":RTS$("TR1") = "ENTER":RTS$("TR2") = "ENTER"

RTS$("TR3") = "ENTER":RTS$("TR4") = "ENTER"

The last sequence reads recording time of trace 1:

RTS$("TR1") = "STOP"

? CEN%,YEA%,MON%,DAY%,HOU%,MIN%,SEC%

CEN% .. The century may be read or set.

Possible values are:
0 .. 99
Example sets century for next RTS$ = \ENTER\ command:
CEN% = 1
Example reads century from last RTS$ = \STOP\ command:
? CEN%
Answer may be:
1

DAY% .. The day of the month may be read or set.

Possible values are:
1 .. 31
Example sets day of the month for next RTS$ = \ENTER\ command:
DAY% = 1
Example reads day of the month from last RTS$ = \STOP\ command:
? DAY%
Answer may be:
1

HOU% ..The hour of the day may be read or set.

Possible values are:
0 .. 23
Example sets hour of the day for next RTS$ = \ENTER\ command:
HOU% = 1
Example reads hour of the day from last RTS$ = \STOP\ command:
? HOU%
Answer may be:
1

MIN% .. The minute of the hour may be read or set.

Possible values are: 0 .. 59
Example sets minute of the hour for next RTS$ = \ENTER\ command:
MIN% = 1
Example reads minute of the hour from last RTS$ = \STOP\ command:
? MIN%
Answer may be:
1

MON% .. The month of the year may be read or set.

Possible values are:
1 .. 12
Example sets month of the year for next RTS$ = \ENTER\ command:
MON% = 1
Example reads month of the year from last RTS$ = \STOP\ command:
? MON%
Answer may be:
1

SEC% .. The second of the minute may be read or set.

Possible values are:
0 .. 59
Example sets second of the minute for next RTS$ = \ENTER\ command:
SEC% = 1
Example reads second of the minute from last RTS$ = \STOP\ command:
? SEC%
Answer may be:
1

YEA% .. The year may be read or set.

Possible values are:
0 .. 99
Example sets year for next RTS$ = \ENTER\ command:
YEA% = 1
Example reads year from last RTS$ = \STOP\ command:
? YEA%
Answer may be:

1

6.4.11 User Communication

MES$ .. The message line may be set.

Possible values are:
Any ASCII string of up to 44 printable characters.
The first index is the line selector. Two lines, 1 and 2 are available. A value of 0 means: delete all lines. The second index specifies line attributes:
0 = normal representation,
1 = double height upper half,
2 = double height lower half,
3 = double height upper half inverse video,
4 = double height lower half inverse video,
5 = underline this character,
7 = normal representation inverse video,
Example sets message line number 1 with normal video in normal representation.
MES$(1,0) = "Press softkey 1 if ready"
Example sets message line number 1 and 2 to a double height line:
MES$(1,1) = "Press softkey 1 if ready"
MES$(2,2) = "Press softkey 1 if ready"
Example sets message line number 1 with inverse video in normal representation.
MES$(1,7) = "Press softkey 1 if ready"

MEV% .. The message visibility status may be read or set.

Possible values are:
0 or 1
Example sets message visibility status to on:
MEV% = 1
Example read message visibility status:
? MEV%
Answer may be, if message visibility status is on:
1

RCS$ .. The rotary control selection may be read or set.

Possible values are:
TRIG | DELAY | CURSOR | REFER | TRACK
Example sets rotary control selection to cursor:
RCS$ = "CURSOR"
Example reads rotary control selection:
? RCS$
Answer may be, if rotary control selection is cursor:
CURSOR

SKV% .. The softkey visibility status may be read or set.

Possible values are:
0 or 1
Example sets softkey visibility status to on:
SKV% = 1
Example reads softkey visibility status:
? SKV%
Answer may be, if softkey visibility status is on:
1

SKY$ .. The softkey labelling may be set.

Possible values are:
Any ASCII string of up to 21 printable characters.
The first index is the key selector. Seven keys are available. A value of 0 means delete all keys. The second index specifies softkey attributes:
0 = double height,
1 = double height inverse video,
Example sets softkey number 1 labelling with normal video:
SKY$(1,0) = "I am ready"

TIK% .. To synchronize on user actions, internal keyboard status may be read

Possible values are found in 7) Key Codes.
Example reads last key depressed:
? TIK%
Answer may be, if no key was depressed:
0

6.4.12 Test Functions

TST% .. The selftest status may be executed and read.

Possible parameters are:
ACQUCTRL | ACQUMEM | DISPMEM | INTEPROC | RAM | ROM | RS232 | RTC
Possible answers are:
0 or 1
Example executes and reads system RAM selftest status:
? TST%("RAM")
Answer may be, if system RAM selftest status is passed:
1

NUL! .. A real variable for general purposes may be read and set.

Example sets general purpose real variable:
NUL! = 12345.678
Example reads general purpose real variable:
? NUL!
Answer may be:
1.2346e+04

NUL$ .. A string variable for general purposes may be read and set.

Example sets general purpose string variable:
NUL$ = "null string"
Example reads general purpose string variable:
? NUL$
Answer may be:
null string

NUL% .. A long integer variable for general purposes may be read and set.

Example sets general purpose long integer variable:
NUL% = 987654321
Example reads general purpose long integer variable:
? NUL%
Answer may be:
987654321

6.5 Print Command

The print command sends a list of elements to the remote interface. It may be placed anywhere in a multi statement line.The print command is called:

? | PRINT

Examples for valid print statements are:
INPUT LINE:
?MOD$
OUTPUT LINE:
SINGLE < EOS > < EOI >

Where < EOS > is the terminator of the remote interface. And is the IEEE end or identify message.

You may use the comma to supress the end character:

INPUT LINE:
?MOD$;

OUTPUT LINE:
SINGLE < EOI >

It is possible to send more items by means of a single print command. The printing elements are separated by a tabulator character when a comma separates the items:

INPUT LINE:

? "m" + "o" + "d" + "e",, MOD$ + " mode = " + MOD$, "XYZZY";

OUTPUT LINE:

mode SINGLE mode = SINGLE XYZZY < EOI >

If more than one print command is sent without addressing the unit to talk after each command, the return strings are queued up.

After setting the unit to talk, you get only the string for the first print statement terminated with < EOI >.

To receive the remaining strings, you have to deaddress the unit and then address it one more time as talker. You will then receive one single block terminated with < EOI > . The line terminator characters are not affected.

INPUT LINES:

? SER$
? TYP$
? VER$;

OUTPUT LINES:

601 < EOS > < EOI >
8608A < EOS > 10 < EOI >

You have to keep attention not to mix up item list with expressions that print the result of a calculation with the items:

item list INPUT LINE:

```
? CUR%, SAM!, TRL%;
```

OUTPUT LINE:

```
5100      2.2E+01      1024
```

expression INPUT LINE:

```
? CUR% + SAM! + ADD%;
```

OUTPUT LINE:

```
6.146E+03
```

7 KEY CODES

If the integer value passed by TIK% is interpreted as hex word, a class code is passed in the high order byte and a function or a value code in the low order byte.

Key Name	TIK% / 100H	TIK% AND 0FFH
DISPLAY #1	010H	001H
DISPLAY #2	010H	011H
DISPLAY #3	010H	002H
DISPLAY #4	010H	012H
DISPLAY 1vs2	010H	021H
DISPLAY 2vs4	010H	022H
DISPLAY INTERP	010H	031H
DISPLAY GRAT	010H	041H
DISPLAY X_ZOOM	010H	032H
DISPLAY SEPAR	010H	042H
INPUT A_ONLY	010H	003H
INPUT BWL	010H	013H
INPUT MIN/MAX	010H	023H
INPUT AUTOSET	010H	033H
INPUT AVG	010H	043H
VERTICAL CHA COUPL	010H	004H
VERTICAL CHB COUPL	010H	014H
VERTICAL CHA OFFSET	009H	signed value
VERTICAL CHB OFFSET	029H	signed value
VERTICAL CHA ATTEN	00AH	signed value
VERTICAL CHB ATTEN	049H	signed value
TRIGGER HYSTERESIS	010H	054H
TRIGGER SOURCE	010H	064H
TRIGGER SLOPE	010H	053H
TRIGGER EXT COUPL	010H	063H
TIME MAX MEM	010H	024H
TIME WRITE/LOCK	010H	044H
TIME MODE	010H	034H
TIME RELEASE	010H	074H
TIME TIMEBASE	069H	signed value
MEASURE SELECT	010H	051H
MEMORY SELECT	010H	071H
SETTING SELECT	010H	025H
SETTING PLOT	010H	035H
CONTROLS TRIG	010H	073H
CONTROLS DELAY	010H	056H
CONTROLS CURSOR	010H	017H
CONTROLS REFER	010H	027H
CONTROLS TRACK	010H	007H
CONTROLS ENCODER	091H	signed value
SOFT F1	010H	075H
SOFT F2	010H	045H
SOFT F3	010H	065H
SOFT F4	010H	066H
SOFT F5	010H	005H
SOFT F6	010H	046H
RETURN	010H	006H

8 FILE FORMAT

8.1 TRACE FILE FORMAT

< file header > < trace control block >

[< parameter block > < data block > ...]

Variable	typ	bytes	value
File Header:			
length of block	unsigned word	2	
type of block = File ID	unsigned word	2	5a81h
number of blocks in file	unsigned word	2	
Software Version	unsigned word	2	(0105h)
end-of-block	unsigned word	2	a55ah
Node Control Block:			
length of block	unsigned word	2	24d
type of block	unsigned word	2	5a01h
samples per file	unsigned word	2	-
versions of file	unsigned word	2	1
data first index	unsigned word	2	0
data actual index	unsigned word	2	0
data wrap flag	unsigned byte	1	-
(spare)		1	0
record century	BCD coded byte	1	-
record year	BCD coded byte	1	-
record month	BCD coded byte	1	-
record day	BCD coded byte	1	-
record hour	BCD coded byte	1	-
record minute	BCD coded byte	1	-
record second	BCD coded byte		-
(spare)		1	0
end-of-block	unsigned word	2	a55ah
Parameter Block:			
length of block	unsigned word	2	112d
type of block	unsigned word	2	5a02h
(spare)		2	0
(spare)		2	0
data type	unsigned word	2	0
addr axis scale type	unsigned word	2	0
addr axis lsb value	real	4	-
addr axis offset value	real	4	-
addr axis dimension V exp.	signed word	2	0
add axis dimension m exp.	signed word	2	0
addr axis dimension s exp.	signed word	2	1
addr axis dimension A exp.	signed word	2	0
(spare)		16	
addr axis dimension code	unsigned byte	1	-
(spare)		15	
value axis scale type	unsigned word	2	0
value axis lsb value	real	4	-
value axis offset value	real	4	-
value axis dimension V exp.	signed word	2	-
value axis dimension m exp.	signed word	2	-
value axis dimension s exp.	signed word	2	-
value axis dimension A exp.	signed word	2	-
(spare)		16	
value axis dimension code	unsigned byte	1	-
(spare)		15	
value axis overflow code	unsigned byte	1	-
trigger mode code	unsigned byte	1	-

ignored during restore operation !

trigger level	unsigned word	2	-
end-of-block	unsigned word	2	a55ah

Data Block:

length of block	unsigned word	2	2 *samples/file + 6
type of block	unsigned word	2	5a03h
data	unsigned words	2	2*samples/file
end-of-block	unsigned word	2	a55ah
checksum	unsigned word	2	-

The checksum is a 16 bit sum (mod 10000h) of all bytes written to the file including all headers, type, data and end records, but excluding the checksum itself.

8.2 SETUP FILE FORMAT

< file header > < channel control block > < trace control block >
< scalar control block > < display control block >

Variable	typ	byte	value
File Header:			
length of block	unsigned word	2	
type of block = File ID	unsigned word	2	5a82h
number of blocks in file	unsigned word	2	
Software Version	unsigned word	2	(0105h)
end-of-block	unsigned word	2	a55ah
Channel Control Block:			
length of block	unsigned word	2	?
type of block	unsigned word	2	5a12h
trigger level	signed word	2	-
trigger slope	boolean byte	1	-
trigger_source	unsigned byte	1	-
trigger_mode	unsigned byte	1	-
autotrigger	boolean byte	1	-
coupling_a	unsigned byte	1	-
coupling_b	unsigned byte	1	-
coupling_ext	unsigned byte	1	-
attenu_a	unsigned byte	1	-
attenu_b	unsigned byte	1	-
offset_a	unsigned word	2	-
offset_b	unsigned word	2	-
timebase	unsigned byte	1	-
recording_mode	unsigned byte	1	-
maximum memory active	unsigned word	2	-
delay_length	unsigned word	2	-
glitch	unsigned byte	1	-
chanlink (A-only)	unsigned byte	1	-
bandwidth	boolean	1	-
average	boolean	1	-
average mode	unsigned byte	1	-
average number	unsigned byte	1	-
end-of-block	unsigned word	2	a55ah
Trace Control Block:			
length of block	unsigned word	2	?
type of block	unsigned word	2	5a14h
trace # 1 origin code (ref 9.1)	byte	1	-
trace # 1 operand 1 typ (ref 9.2)	byte	1	-
trace # 1 operand 1 index	byte	1	-
trace # 1 operand 2 typ (ref 9.2)	byte	1	-
trace # 1 operand 2 index	byte	1	-
trace # 2 origin code(ref 9.1)	byte	1	-
trace # 2 operand 1 typ (ref 9.2)	byte	1	-
trace # 2 operand 1 index	byte	1	-
trace # 2 operand 2 typ (ref 9.2)	byte	1	-

trace # 2 operand 2 index	byte	1	-
trace # 3 origin code(ref 9.1)	byte	1	-
trace # 3 operand 1 typ (ref 9.2)	byte	1	-
trace # 3 operand 1 index	byte	1	-
trace # 3 operand 2 typ (ref 9.2)	byte	1	-
trace # 3 operand 2 index	byte	1	-
trace # 4 origin code(ref 9.1)	byte	1	-
trace # 4 operand 1 typ (ref 9.2)	byte	1	-
trace # 4 operand 1 index	byte	1	-
trace # 4 operand 2 typ (ref 9.2)	byte	1	-
trace # 4 operand 2 index	byte	1	-
end-of-block	unsigned word	2	a55ah

Scalar Control Block:

length of block	unsigned word	2	?
type of block	unsigned word	2	5a10.2h
scalar # 1 origin code(ref 9.1)	byte	1	-
scalar # 1 operand 1 typ (ref 9.2)	byte	1	-
scalar # 1 operand 1 index	byte	1	-
scalar # 1 operand 2 typ (ref 9.2)	byte	1	-
scalar # 1 operand 2 index	byte	1	-
scalar # 2 origin code (ref 9.1)	byte	1	-
scalar # 2 operand 1 typ (ref 9.2)	byte	1	-
scalar # 2 operand 1 index	byte	1	-
scalar # 2 operand 2 typ (ref 9.2)	byte	1	-
scalar # 2 operand 2 index	byte	1	-
scalar # 3 origin code(ref 9.1)	byte	1	-
scalar # 3 operand 1 typ (ref 9.2)	byte	1	-
scalar # 3 operand 1 index	byte	1	-
scalar # 3 operand 2 typ (ref 9.2)	byte	1	-
scalar # 3 operand 2 index	byte	1	-
scalar # 4 origin code(ref 9.1)	byte	1	-
scalar # 4 operand 1 typ (ref 9.2)	byte	1	-
scalar # 4 operand 1 index	byte	1	-
scalar # 4 operand 2 typ (ref 9.2)	byte	1	-
scalar # 4 operand 2 index	byte	1	-
end-of-block	unsigned word	2	a55ah

Display Parameter Block:

length of block	unsigned word	2	?
type of block	unsigned word	2	5a15h
interpol	unsigned word	2	
x-zoom	unsigned word	2	
x-position	unsigned dword	4	
y-seperation	unsigned word	2	
xy12	unsigned word	2	
xy34	unsigned word	2	
cursor position	unsigned dword	4	
reference position	unsigned dword	4	
graticule	boolean word	2	
rotary select	unsigned word	2	
end-of-block	unsigned word	2	a55ah
checksum	unsigned word	2	-

The checksum is a 16 bit sum (mod 10000h) of all bytes written to the file including all headers, type, data and end records, but excluding the checksum itself.

8.3 ALL FILE FORMAT

< file header > < channel control block > < trace control block >

< scalar control block > < display control block >

[< node control block > < parameter control block >

< data control block >]...

[ ] .. for each trace defined in traces control block.

Variable	typ	byte	Value
File Header:			
length of block	unsigned word	2	
type of block = File ID	unsigned word	2	5a83h
number of blocks in file	unsigned word	2	
Software Version	unsigned word	2	(0105h)
end-of-block	unsigned word	2	a55ah

9 FILE DATA CODES

9.1 ORIGIN CODES

delete an origin:	00	OFF
assign an origin:	01	EQU ("<source node>")
for differentiation of one trace:	16	DIF ("<source node>",rr)
for integration of one trace:	15	INT ("<source node>",rr)
for negation of one trace:	17	NEG ("<source node>,<source node>")
for smoothing of one trace:	18	SMO ("<source node>",ii)
for addition of two traces:	11	ADD("<source node>,<source node>")
for division of two traces:	14	DIV ("<source node>,<source node>")
for multiplication of two traces:	13	MUL("<source node>,<source node>")
for subtraction of two traces:	12	SUB ("<source node>,<source node>")
Cursor to Reference amplitude of trace:	22	CRA ("<source node>")
Cursor to zero amplitude of trace:	20	CZA ("<source node>")
50% rise amplitude of trace:	29	FIA ("<source node>")
Maximum to zero amplitude of trace:	23	MAA("<source node>")
Minimum to zero amplitude of trace:	24	MIA ("<source node>")
Peak to peak amplitude of trace:	25	PPA ("<source node>")
10% to 90% rise amplitude of trace:	30	RIA ("<source node>")
Reference to zero amplitude of trace:	21	RZA ("<source node>")
DC to zero amplitude of trace:	26	DCA ("<source node>")
Mean to zero amplitude of trace:	28	MEA("<source node>")
Root mean square amplitude of trace:	27	RMS ("<source node>")
Cursor amplitude of left trace to reference amplitude of right trace linear:	39	CRI ("<source node>,<source node>")
Cursor amplitude if left trace to reference amplitude of right trace logar.:	40	CRO ("<source node>,<source node>")
Root mean square of left trace to root mean square of right trace linear:	37	RMI ("<source node>,<source node>")
Root mean square of left trace to root mean square of right trace logar.:	38	RMO ("<source node>,<source node>")
Cursor to reference time of trace:	31	CRT ("<source node>")
Cursor to trigger time of trace:	32	CTT ("<source node>")
Reference to trigger time of trace:	33	RTT ("<source node>")
10% to 90% time of trace:	34	RIT ("<source node>")
Frequency of trace:	36	FRQ ("<source node>")
Period time of trace:	35	PER ("<source node>")

Phase difference time of left trace to right trace:	41	PHT	("<source node>,<source node>")
Phase difference degrees of left trace to right trace:	42	PHD	("<source node>,<source node>")

9.2 OPERAND TYPE CODES

1 = Operand is a channel:	index 0 = channel A
	index 1 = channel B
2 = Operand is a trace:	index 0 = trace #1
	.....
	index 3 = trace #4
3 = Operand is a file:	index 0 = M0
	.....
	index 99 = M99
4 = Operand is a constant:	index value = constant
	(e.g. DIF(...,10)

9.3 INTERPOLATION CODES

0 = OFF
1 = LINEAR
2 = SINE
3 = PULSE

9.4 X-ZOOM CODES

0 = *0.05
1 = *0.1
2 = *1
3 = *10

9.5 ROTARY SELECT CODES

0 = TRIG
1 = DELAY
2 = CURSOR
3 = REFER
4 = TRACK