Sefram

PROGRAMMING MANUAL

DAS220/DAS240

Edition Octobre 2020

CONTENT

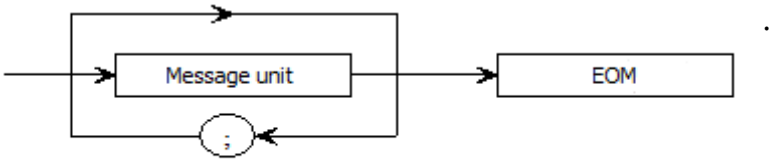
1. Programming language

1.1. Format of the reception messages

In all following examples, the blank character is displayed as a space.

Exchanges from a controller to the recorder are made of messages as successive ASCII characters (and possibly binary octets) with an EOM at the end.

Syntax of a reception message



Message unit: if the message includes several message units, they are separated by a semi-colon " ; " and with possible one or several "filling" characters before and after in ASCII code (0 to 32, decimal, except 10 and 13).

The EOM is designed for the Ethernet link:
- LF: Line Feed (10 in decimal)

The EOM may be preceded with one or several "filling" characters in ASCII code (0 to 32, decimal, except 10 and 13).
Message example made of 3 message units:
MESSAGE 1; MESSAGE 2; MESSAGE 3; EOM
CHANNEL 1; TYPE:VOLTAGE DC; CALDEC ? EOM

Syntax of a message unit

A message unit (example: REAR:SETUP 1) is made of several fields:

- Header:
For command messages (example: **REAR:SETUP** 1) or interrogation messages (example: **REAR** ?), it is made of a chain of characters (simple header) or of several chains separated with the ":" character (composite header).
A chain includes 1 to 12 alphanumerical characters or "_" (ASCII code 95 in decimal). Recommended chain length: 4 characters.
A header chain must start with an alphanumerical character. It may be preceded by 2 dots ":" (composite header) or finish with a question mark "?" (interrogative message).

An interrogative message must be followed by an EOM.

- Header separator:
One or several ASCII characters (0 to 32, decimal, except 10 and 13).

- One or several data items:
(example: SPEED **1**, **MM_S**), alphanumerical, numerical or made of any characters and binary octets.

- Data separator:

A comma "," possibly followed and/or preceded with one or several "filling" characters in ASCII code (0 to 32, decimal, except 10 and 13).

Data:
There are several types of data items:

- *Alphanumerical data:*
1 to 12-character words that can be alphabetical (upper or lower case), digital or the "-" character (95d).
A word always starts with an alphabetical character.
For example, for a non-digital parameter: S1M.
- *Decimal digital data:*
Made of a significand and, possibly, an exponent, and displayed as a chain of ASCII-coded characters starting with a digit or a sign (+ or -). It is of NR1 (integer), NR2 (decimal) or NR3 (with exponent) type or a combination of these three types.
- *Text:*
Any chain of characters under 7-bit ASCII code, between quotation marks (") or apostrophes (').
For example: "Channel 1"

1.2. Formats of the emited messages

Exchanges from the recorder to a controller are made of messages as successive ASCII characters (and possibly binary octets) with an EOM at the end.
The format of the emission messages is identical to the reception messages. However, their structure is stricter.

The syntax of an emission message is: **message unit + EOM**

Message unit:
If the message includes several message units, they will be separated with a semicolon ";".

EOM:
- LF: Line Feed (10 in decimal)

Syntax of a message unit:

A message unit (for example: TYP:THE J, COMP) is made of several fields

- Header:
(for example: **TYP:THE**) is made of one (simple header) or several (composite header) 1 to 12-character alphabetical chains (upper case only or digital or "_" (ASCII code 95 in decimal)
A header chain starts with an alphabetical character.
In a composite header, the chains of characters are separated with the ":" character (for example: TYP:THE).

- Header separator:
"Space" character (32d) only.

- One or several data items:
(for example: **J**, **COMP**) alphanumerical, numerical or made of any characters and binary octets.

- One data separator:
A comma ",".
Data:
There are several types of data items:

- Alphanumerical data:
1 to 12-character words that can be alphabetical (upper case only), digital or the "-" character (95d) (example: **J**).

- Decimal digital data:
Made of a chain of ASCII-coded characters starting with a digit or a sign (+ or -). It is of NR1 (integer), NR2 (decimal) or NR3 (with exponent) type.
For example, for a digital character: -25.02.

- Text data:
Any chain of characters under 7-bit ASCII code, between quotation marks (") or apostrophes (').
For example: "A".

- *Any chain of ASCII characters:* ends with the EOM.

2. Standard instructions

All these instructions start with an asterisk "*".

***IDN ?** IDENTIFICATION REQUEST OF AN APPLIANCE
answer by the appliance: 4 data items separated with ',':

- the trademark of the appliance
- the name of the appliance followed with _nn, where nn is the number of inputs of the recorder
- the serial number of the appliance (0 if unknown)
- the software version number as x.xx x

***OPT ?** IDENTIFICATION REQUEST OF THE OPTIONS OF AN APPLIANCE
answer by the appliance: n data items separated with ';':

- number of cards
- number of channels per card

***RST** INITIALIZATION OF AN APPLIANCE
action: initialization of the recorder in a fix configuration (inputs under voltage, caliber: 10 V, center: 0 V...)

***REM** TRANSITION TO PROGRAMMING (REMOTE)
compulsory with RS232C before sending any other program command.

***LOC** RETURN TO LOCAL MODE

***CLS** CLEARING THE STATE REGISTERS
action: the appliance reinitializes the state registers.

***ESE** VALIDATION OF THE STANDARD EVENT BITS OF AN APPLIANCE
*ESE is followed with a number between 0 and 255
action: changes the standard event validation register and updates the ESB bit in the state register of service requests (see the following paragraph).

***ESE ?** REQUEST OF THE CONTENT OF THE STANDARD EVENT VALIDATION REGISTER OF AN APPLIANCE
Answer by the appliance: NR1 number from 0 to 255 (see the following paragraph).

***ESR ?** REQUEST OF THE CONTENT OF THE STANDARD EVENT VALIDATION REGISTER OF AN APPLIANCE
Answer by the appliance: NR1 number from 0 to 255
All events are erased and the register is cleared (see the following paragraph).

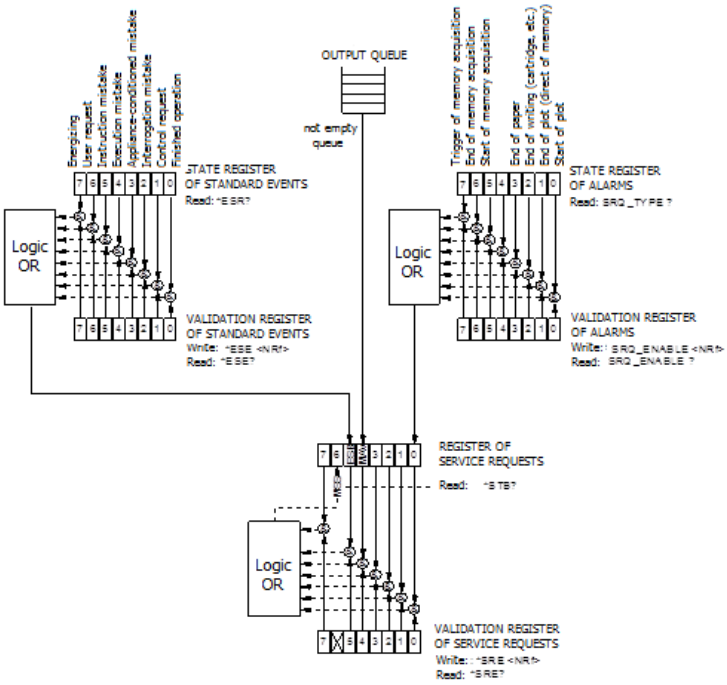
***SRE** VALIDATION OF THE SERVICE REQUESTS OF AN APPLIANCE
*SRE is followed with a number between 0 and 63 or from 128 to 191.
action : the appliance changes the validation register of service requests (see the following paragraph).

***SRE ?** INTERROGATION OF THE VALIDATION REGISTER OF THE SERVICE REQUESTS OF AN APPLIANCE
answer by the appliance: NR1 number from 0 to 63 or from 128 to 191 (see the following paragraph).

***STB ?** READING THE SERVICE REQUEST REGISTER OF AN APPLIANCE
answer by the appliance: NR1 number from 0 to 255: state word with bit 6 MSS (Master Summary Status) (see the following paragraph)

2.1. State indication of the appliance

Here is the model of structure of the state data that documents state changes in the appliance (energizing, printing launch...).



Overview of the structures of the state data of the register:

4 registers are used:
- the register of service request (STB) associated with its validation register
- the register of standard events (ESR) associated with its validation register

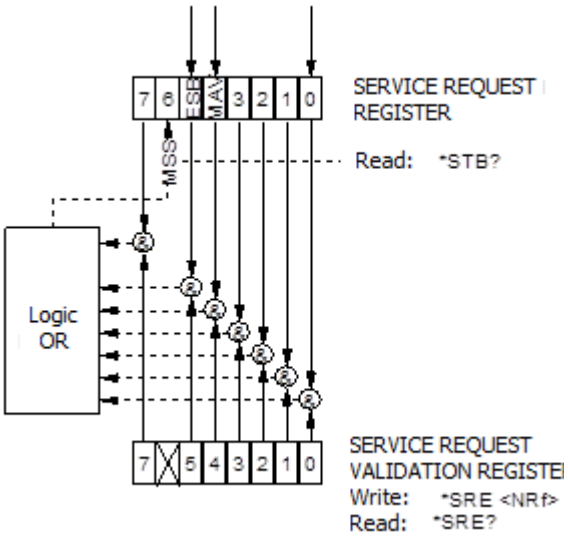
The bits #0, 1, 2 and 7 of the STB register are available as sum-up messages specific to the appliance. Each of these bits can be associated with a data structure, whose model is defined and manages the events of the appliance that may induce a service request.

The user can set up the recorder so that it triggers the bit #6 of the service request register at a few specific events.
In RS232, you have to regularly read the service request register to detect events.
Events are identified by reading the state word, then the associated event register(s).

State of these registers at power-up:

The content of the STB, ESR and alarm registers is systematically cleared at power-up (except the bit #7 of the ESR that specifies a power-up).

2.2. Service request register



State register:
It contains the state word of the appliance.
This state word can be read by request with the instruction "*STB?": In this case, the bit #6 is MSS (Master Summary Status) resulting from the logic operations as in the figure here.
In fact, MSS is 1 when at least one other bit is 1 both in the state register and in the validation register.

Composition of the STB register:
The bit #6 (value 64) contains the sum-up message "MSS" (reading with "*STB?").

The service request takes place in the following cases:

- a bit from the service request state register switches from 0 to 1 while the corresponding bit in its associated validation register is at 1, and reversely
- the bit #5 of the service request validation register is at 1 and an event happens in the following conditions:
- a bit from the service request state register switches from 0 to 1 while the corresponding bit in its associated validation register is at 1
- a bit from the service request validation register switches from 0 to 1 while the corresponding bit in its associated state register is at 1
- the bit #0 of the service request validation register is at 1 and an event happens in the following conditions:
- a bit of the alarm state register switches from 0 to 1 while the corresponding bit in its associated validation register is at 1
- a bit of the alarm state register switches from 0 to 1 while the corresponding bit in its associated state register is at 1.

The bit #5 (ESB: Event Status Bit, value 32) contains the sum-up message of the standard events state register (see the detail of these bits in the description of this register). Its state specifies whether one or several authorized events showed up in the standard events state register after its latest clearing (an event is authorized if the corresponding bit in the event validation register is 1).

The bit #4 (MAV: Message AVailable, value 16) contains the sum-up message of the output queue. Its state specifies if a message or data from the appliance are ready for emission through the interface (ex: answer to an interrogative instruction).

The bits #7 and 3, 2, 1, 0 are used to receive sum-up messages as defined by the appliance. In the case of the recorder, the bit #0 is used while the bits #1, 2, 3 and 7 are always 0.
The bit #0 contains the sum-up message of the alarm state register (see the detail of these bits in the description of this register). Its state specifies whether one or several authorized events showed up in the alarm state register after its latest cleaning.

Validation register:
A state word is associated with a validation register, which makes it possible to control the service request by authorizing only specific cases.
When a bit is 1, it allows that the state 1 of the bit of same rank in the state register (STB) leads to the activation of the bit #6 in the same state register.
Writing into the validation octet is made with the *SRE<NRF> command, where <NRF> is the sum of the binary values of the bits 0 to 5 and 7.
Reading the validation octet is made with the instruction *SRE?. The answer is given in decimal format (NR1).

2.3. Standard events register

See the overview of the structures of the state data.
The structure of the standard events register is assigned to the bit #5 of the service request register.

State register:

This register contains a few specific messages with the following meanings.
You can read its content with the *ESR? command.
Reading leads to the erasing of the register.
The bits of the events state register are assigned to specific events:

- BIT 7: POWER-UP (Value 128)
  Shows that the appliance is energized.
- BIT 6: USE REQUEST (Value 64)
  Not used, positioned at 0
- BIT 5: INSTRUCTION MISTAKE (Value 32)
  Specifies that an unknown or incorrect instruction has been sent to the recorder.
- BIT 4: EXECUTION MISTAKE (Value 16)
  Not used, positioned at 0
- BIT 3: APPLIANCE-CONDITIONED MISTAKE (Value 8)
  Not used, positioned at 0
- BIT 2: INTERROGATION MISTAKE (Value 4)
  Specifies that the output queue is full and some data is or may be lost.
- BIT 1: CONTROL REQUEST (Value 2)
  Not used, positioned at 0
- BIT 0: FINISHED OPERATION (Value 0)
  Not used, positioned at 0.

An event is authorized is the corresponding bit in the event validation register is 1.

Validation register:
It makes it possible to control the standard events state register:
When a bit in this register is 1, it makes it possible that the state 1 of the bit of same rank in the standard events state register leads to the switch to 1 of the **bit #5** of the service request state register (STB).
Writing into this register is made with the *ESE<NRF> command, where <NRF> is the sum of the binary values of the bits inside the validation register.
Reading this register is made with the "*ESE?" command.

2.4. Alarms register

See the overview of the structures of state data.
The structure of alarm registers is assigned to the bit #0 of the service request register.

State register:
This register contains a few specific messages to the recorder with the following meanings.
You can read its content with the SRQ_TYPE ? command
Reading the register leads to the erasing of its content.

The bits of the alarms state register are assigned to specific events:

- BIT 7: TRIGGER OF MEMORY ACQUISITION (Value 128)
  Specifies that the triggering condition of a data acquisition into memory has been achieved.
- BIT 6: END OF MEMORY ACQUISITION (Value 64)
  Specifies that a data acquisition into memory has ended.
- BIT 5: START OF MEMORY ACQUISITION (Value 32)
  Specifies that a data acquisition into memory has started.

An event is authorized only if the corresponding bit in the event validation register is 1.

Validation register:
It makes it possible to control the alarms state register:
When a bit in this register is 1, it makes it possible that the state 1 of the bit of same rank in the alarms state register leads to the switching to 1 of the **bit #0** of the service request state register (STB).
Writing into this register is made with the *SRQ_ENABLE <NRF> command, where <NRF> is the sum of the binary values of the bits of the validation register.
Reading this register is made with the "SRQ_ENABLE ?" command.

2.5. Using the structure of state data

Before any use, it is advisable to send the recorder the instruction *CLS that clears all state registers.
You should first determine which events you would like to detect by authorizing them in the validation registers:

- with the "SRQ_ENABLE n" command for events associated to the alarm registers
- with the "*ESE n" command for events associated with standard events registers
- with the "*SRE n" command for events associated with the service request register.

Example:
Programming a service request for a start of end of paper printing, an instruction mistake, the presence of data at the output of the recorder, is made with the commands:

SRQ_ENABLE 3	(Bits 0 and 1 switch to 1)
*ESE 32	(Bit 5 switches to 1)
*SRE 49	(Bits 0, 4 and 5 switch to 1)

In RS232 mode, the controller must regularly read the service request register with the "*STB?" command. Switching the bit #6 (MSS) to 1 shows that an authorized event happened.
When read, the word of state makes it possible to determine the type of event that happened. In the case of a standard or specific event, you must read the associated state register with the "*ESR?" or "SRQ_TYPE ?" command to precisely know the event.

A standard event happened: The user sends the "*ESR?" command:
Answer by the recorder: 160 (Bits 7 and 5 switch to 1)
Two events are displayed (energizing and instruction mistake); the instruction mistake (only event authorized in the validation register) triggered the service request.

3. Programming dictionary

In the following tables, sending the lower case characters of the headers and parameters is facultative.
As a rule, the digital parameters are integers (NR1); where it is specified "decimal" can be of NR1, NR2 or NR3 type.

3.1. Configuration

HEADER	PARAMETERS	EXAMPLES
PAGE	P1 Display selected screen P1= SETUP : Config CHAN : channel N (see command: CHAN) TRigger : trigger SCOpe : direct display REPLay : Display the last records files open or the current recording.	:CHAN A3 ;:PAGE CHAN Display of the page with channel A3
ALArm	P1 Number of alarm P1 = A, B , C or D	ALA A,:ALA:DEF TR TRIG :CHAN A1,S1,POS The trigger is defined by the TRig command: (see 15.5.8)
:ALArm:DEF	P1 Alarm behavior P1 = NO, TRigger, RECtr (trig on record start)	
ALArm ?	Display alarm behavior	
DATE	P1, P2, P3 Define the current date P1,P2,P3 = day, month , year	DATE 11,12,18 December 11th 2018
DATE ?	Display the date	
HOURS	P1, P2, P3 Define the current time P1,P2,P3 = hour, minutes, second	HOURS 10, 6, 0 10 hours 6 minutes
HOURS ?	Display the time	

3.2. File Setup

HEADER	PARAMETERS	EXAMPLES
RECALL	P1 recovering a set-up P1 = name of the set-up	RECALL “foldercnf/File1” Recover s the set-up File1.cnf in the folder foldercnf
STORE	P1 saving a set-up P1 = name of the set-up	:STORE “Conf 2” save the set-up into the file with the name « Conf 2.cnf »
READSETup	Recovering the current set-up in binary format; the appliance sends 4 octets specifying the number of bytes and 2 bytes specifying the checksum to send and then the setup file : N bytes (N=34008)	READSETup
SENDSETup	Send a set-up in binary format: 4 bytes specifying the length of the file and 2 bytes specifying the checksum of the set-up (format little endian)	SENDSETup D8 84 00 00 AB 23 xx xx xx xx xx *000084D8= 34008 bytes *23AB = checksum … add the file
KEYBLock	P1 locking the keyboard (ON or OFF) P1= ON or OFF	KEYBLock ON

3.3. Parameters of the channels

HEADER	PARAMETERS	EXAMPLES
CHANnel	P1 Define the CHANNEL input to change with P1= selection of the input A1, A2 etc. For board A to J K1 to K4 for logic channnels	CHAN B3 We selected to change the channel 3 of the card B
CHANnel ?	Displays the number of selected input and value	
VALID	P1, P2 Defines the activation status of channels P1= ALL for all channels or A1, A2 etc. for each channel LOG for logic channels P2= ON or OFF	VALID ALL,OFF ;VALID A1 ON ; VALID LOG, ON FUNCT ON, VALID FA1,ON We authorized the channel A1 the function FA1 and the logic channels only
VALID ?	Displays the validity of all channels	
NAMe	P1 Change the name of the CHANNEL input P1= name (26 characters max.) between two ' or "	:CHAN B3 ; :NAM 'OWEN N1'
NAMe ?	Display the channel name	
COLOR	P1, P2, P3 Color of each channel P1= value for red (0 to 100) P2= value for green P3= value for blue	:CHAN A2,COLOR 100,100,100 White color
FILTER	P1 Definition of the filter of the channel as defined with the CHANNEL command P1= value from 0.01 to 10 (0 is no filter)	FILTER 0.5 Filter 0.5Hz FILTER 0 No filter
FILTER ?	Display the filter of the selected input	
RANGE	P1, P2, P3 Change input range and center P1= range in ISO units (Volts or °C) in real time P2= center in ISO units in real time P3= position in percentage (-100 to 100)	RANGE 12, 3, 0 Range = 12 V center on 3 V RANGE 20 0, -100 Range=20V Min=0V Max=20V
RANGE ?	Display the range and the center of the selected input	
THREShold	P1, P2, P3 Define thresholds P1= SI or S2 P2= ON or OFF (validity of the screen) P3= value of the threshold	:THRES S1, ON, 10 Threshold S1 is 10 V
THREShold ?	Display the values of the 2 thresholds	

HEADER	PARAMETERS	EXAMPLES
TYPe:VOLtage	Change channel type to voltage	TYPe:VOLtage
TYPe:SHUNT	P1 Change channel type to shunt P1= shunt value	TYPe:SHUNT
TYPe:PT100	P1, P2 Change channel type to Pt100 P1= W2, W3 for 2 wires, 3 wires P2= Resistance value (unused if 3 wires)	TYPe:PT100 W3,0 Or TYPe:PT100 W2,1.2
TYPe:PT1000	P1, P2 Change channel type to Pt1000 P1= W2, W3 for 2 wires, 3 wires P2= Resistance value (unused if 3 wires)	TYPe:PT1000 W3,0 Or TYPe:PT1000 W2,0.3

TYPe:THErmo	P1, P2, P3 Change channel type to thermocouple P1= Thermocouple= J, K, T, S, B, E, N, C,L P2 : COMP or NOCOMP P3 : unit (CEL, FAR, KEL)	TYPe:THERMO J,COMP,CEL
TYPe:RESistance	Change channel type to resistance	TYPe:RESistance
TYPe:FREQ	Change to frequency (only channel K1 to K4)	CHAN K1;TYPe:FREQ
TYPe:PWM	Change to PWM (only channel K1 to K4)	CHAN K2;TYPe:PWM
TYPe:COUNTer	Change to counter (only channel K1 to K4)	CHAN K3; TYPe:COUNTER
TYPe ?	Displays the type of channel	
CHANNELSAMPPERIOD	Set the sampling period of a channel (in ms) P1 : 1, 2, 5, 10, 20, 50, 100 For Pt100/1000, the sampling period is twice as much.	CHAN A2; CHANNELSAMPPERIOD 2

DEFLOG	P1, P2, P3, P4, P5 Definition of the logic channels P1= number of the logic channel P2= value for red (0 to 100) P2= value for green P4= value for blue P5=name of channels	

HEADER	PARAMETERS	EXAMPLES
FUNCMAth	P1 Allows the selection of a mathematical function for the CHANNEL input P1 = Type of function : NONe, UNIT, AX, ABSX, SQRX, SQROOTX, LOGX, EXPX,AINVX *(none, change of unit, ax*+b, a|x|+b, ax^2+b, ...)	CHAN A2; FUNCTION LOGX; The channel 2 = aLog(x)+b
FUNCMAth ?	Returns the function of the CHANNEL function	
COEFf	P1,P2 Definition of the coefficients of the function P1 = A, B, C, D or X1, X2, Y1, Y2 Returns the values of the coefficients of the function of the CHANNEL input	:COEF A,2;COEF B,0 A = 2 B = 0
UNITFunction	P1 Definition of the unit of the function "P1 = name of the unit (max. 6 characters) between two "" or ' ."	UNITF 'DB'

HEADER	PARAMETERS	EXAMPLES
FUNCXY	P1,P2,P3 Additional function between channels P1 = Number of the channel 1 (from A1 to K4) P2 = Operateur PLUS,MINUS,MULT,DIV P3 = Number of the channe 1 (from A1 to K4)	CHAN FA2; FUNCXY A1,PLUS,A2; COEF A,1;COEF B,2; COEF C,3; FA2=A1+2*A2+3;
FUNCMAth ?	Returns the function of the CHANNEL function	
FUNCTion	P1 Validity of the functions in general P1 = ON or OFF	

3.4. Recovery of instant values:

HEADER	PARAMETERS	EXAMPLES
RDC ?	Sends the values of all channels and the logic channels or the parameters in network analysis	RDC ? Answer : A1:> 50.000°C,A2:=0.0123 V,
RDCBINary	Sends the values of all (256) channels (A1-J20, K1-K4, FA1-FJ4 and 12 logicals channels) as binary floating point single precision according to IEEE 754. Read 1024 bytes.	RDCBINary

3.5. Acquisition

HEADER	**PARAMETERS**	**EXAMPLES**
:START:MANual	Manual triggering (stop or start)	:START:MANUAL
:START:TRIG	Triggering with a combination of thresholds (see 7.3)	:START:TRIG;:TRIG:CHAN A1, S1, POS
:START:WAIt	P1,P2, P3 Triggering according to a delay P1= number of hours waiting (0 to 23) P2, P3= minutes, seconds (0 to 59)	:START:WAIT 0, 2, 10 Waiting for 2 min 10 s
:START:DATe	P1, P2, P3, P4, P5, P6 Triggering with a date P1= day (1 to 31) P2= month (1 to 12) P3= year (0 to 99) P4= hour (0 to 23) P5, P6= minute ? second (0 to 59)	:START:DATE 3,10,06,15,30,10 Start on 3/10/06 at 15:30:10
:START:AUTO	Automatic triggering (except in DIRECT mode)	
START ?	Displays the initial command	
:STOP:MANual	Manual stop (direct mode)	
:STOP:TRIG	Triggering with a combination of thresholds (see 7.3)	
:STOP:AUTO	Automatic stop	
STOP ?	Displays the command of end of acquisition	

HEADER	PARAMETERS	EXAMPLES
MEMSpeed	P1, P2 Definition of the sampling period P1= period (1 to 500) P2=, MIL, Sec, Min, HOur is the unit	MEMSPEED 10,MIL 10 millisec period (100 Hz)
MEMSpeed ?	Displays the acquisition speed	
:FILE:NAMe	P1 Name of the save file P1 : name of the file (20 characters max.)	:FILE :NAME " FileName"
:FILE:NAMe ?	Displays the name of the save file	
:FILE:LENGth	P1, P2 Restriction of the number of samples P1= value (P1 means illimited length) P2= KSample or MSample or GSample	:FILE:LENG 100,KS =Limited to 100ksamples :FILE:LENG 0,KS =nolimit
:FILE:LENGth ?	Displays the limitation of the length of file	
REARm	P1 Definition of manual reloading P1= SINgle, AUTo	REARm SINGLE
REARm ?	Displays the type of loading	

3.6. Launching acquisitions

HEADER	PARAMETERS	EXAMPLES
RECORD	P1 Start or stop of the record P1= ON : launching OFF : stop TRIG : forcing the triggering	RECORD ON
RECORD ?	Displays the state of the command and the percentage of memory acquisition	

3.7. Trigger types

HEADER	PARAMETERS	EXAMPLES
:TRIG:TYP	P1 defines the type of general trigger P1= EDGE or LEVEL	:TRIG :TYP EDGE
:TRIG:LOG P1	P1 Selection of trigger on the logic channels P1= defines the 16 trigger values; add a message delimiter (quotation marks)	:TRIG :LOG « XXXXXXXXXX1 » Trigger on logic channel VL1
:TRIG:CHAN P1, P2, P3	P1= number of the channel (A1, A2, etc.) P2= threshold (S1 or S2) P3= POS or NEG For rising or falling edge	:TRIG:CHAN A1,S1,POS Trigger on the rising edge of the channel A1 (threshold 1)
:TRIG:COm P1	Selection of the type of complex trigger P1= OR, AND are: • one threshold (OR) • all thresholds (AND)	:START:TRIG; :TRIG :COm AND ;:TRIG:COM:RESET; :TRIG:COm:ADD A1,S1,POS; :TRIG:COm:ADD A2,S1,POS; There are 2 thresholds (S1 on A1 and S1 on A2)
TRIG:COm:REset	ON removes all channels (complex trigger)	
TRIG:COm:ADD P1, P2,P3	Adds a threshold to the complextrigger P1= number of the channel (A1, A2, etc.) P2= threshold (S1 or S2) P3= POS or NEG For a rising or a falling edge	
TRIG ?	Displays the value of the selected trigger	

The programmed trigger depends on the latest sent command (alarm, start/stop trigger, etc.)

3.8. Direct display

HEADER	PARAMETERS	EXAMPLES
SCREEN	P1 Definition of the visualization mode P1 = FT, TEXT or XY	SCREEN FT
SCREEN:XY	P1, P1= channel X of A1, A2, etc.	SCREEN:XY A3
SCREEN:TIMEBASE	P1,P2 Definition for the time base in Scope mode P1= value (1 to 500) P2= MILlisec, Sec, Min or HOurs	SCREEN :TIMEBASE 500,MIL; :SCREEN FT;:PAGE SCOPE; :SCREEN ON

SCREEN:RUN	P1 Start or stop the Scope mode P1= ON or OFF Stop the Scope mode	We change the time base, then we display the scope f(t) screen
SCREEN:RUN ?	Displays the Scope mode	
:SCREEN:FT	P1,P2 P1= BOUNON/BOUNOFF : Display or not the scale of the graph P2=FULLON/FULLOFF: Full screen activation	:SCREEN:FT BOUNON,FULLON;" BOUNON/BOUNOFF

3.9. Mathematical functions

HEADER	PARAMETERS	EXAMPLES
MATH	P1 Number of mathematical functions P1 = (1 to 5) max 5 math. functions.	MATH 3
MATHDEF	P1, P2, P3 Definition of a function P1= number of the function P2= used channel P3= function MIN MAX PK_PK LOW HIGH AMPL P_OVERSH N_OVERSH FREQ PERIOD R_EDGE F_EDGE P_WIDHT N_WIDTH P_DUTTY_CYCLE N_DUTTY_CYCLE MEAN RMS STD_DEV	MATHDEF 1,A1,MIN
MATH ?	Reading of the mathematical functions as binary floating point single precision according to IEEE754. Read 20 bytes (5 values as 4 bytes). ON must be in visualization f(t) mode to get the values. If the result value is higher or lower to the channel range (range 10V and result is more than 5V) or calculation is wrong, the returned value will be NaN.	MATH ?

3.10. Service request

Refers to the explanations about the data state structure.

HEADER	PARAMETERS	EXAMPLES
SRQ_ENABLE	P1 Changes the alarm validation register P1= value of the register bit decimal value use 5 32 start of acquisition 6 64 end of acquisition 7 128 trigger acquisition	SRQ_ENABLE 3
SRQ_ENABLE ?	Displays the value of the alarm validation register	
SRQ_TYPE ?	Displays the value of the alarm state register Then, the register is erased The definition of each bit is the same as for SRQ_ENABLE	SRQ_TYPE ? The recorder displays :SRQ_TYPE 4 which means « one finished writing operation »

3.11. Error messages

In case of trouble with the programming through the recorder interface, a debugging window shows up on screen to help you identify your mistake:

# error	Explanation
1	Unknown header
2	Unknown parameter
3	Forbidden parameter
4	Absent parameter
5	Wrong parameter separator
6	Wrong message separator
7	Too long word
8	Wrong format for text parameter
9	Forbidden interrogation
10	Digital parameter out of range
11	Text parameter out of range
12	Compulsory interrogation
13	Emission buffer full
14	Impossible in this context
15	Checksum error

At each error matches a line specifying:

- a mistake number
- the received message

When the window is full, the mistakes are displayed from the first line.
The last error line is followed by a blank page.