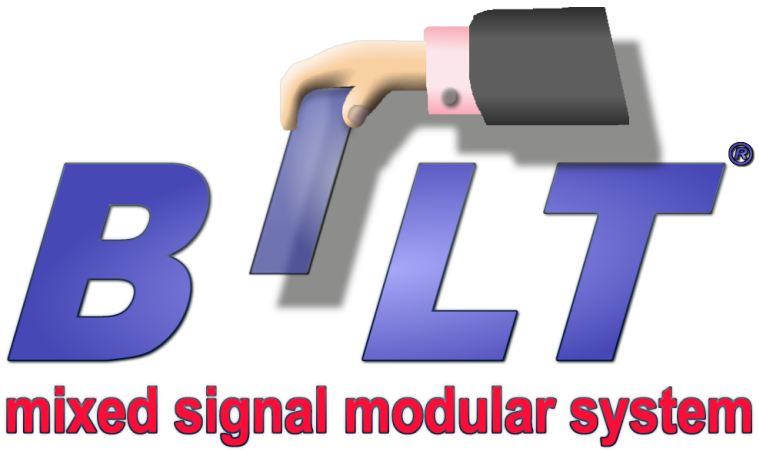


iTest - Bilt System - Programming

User manual

This document provides the fundamental programming elements of Bilt system. It does not contain the specific commands for each module, which are described on their own datasheet.

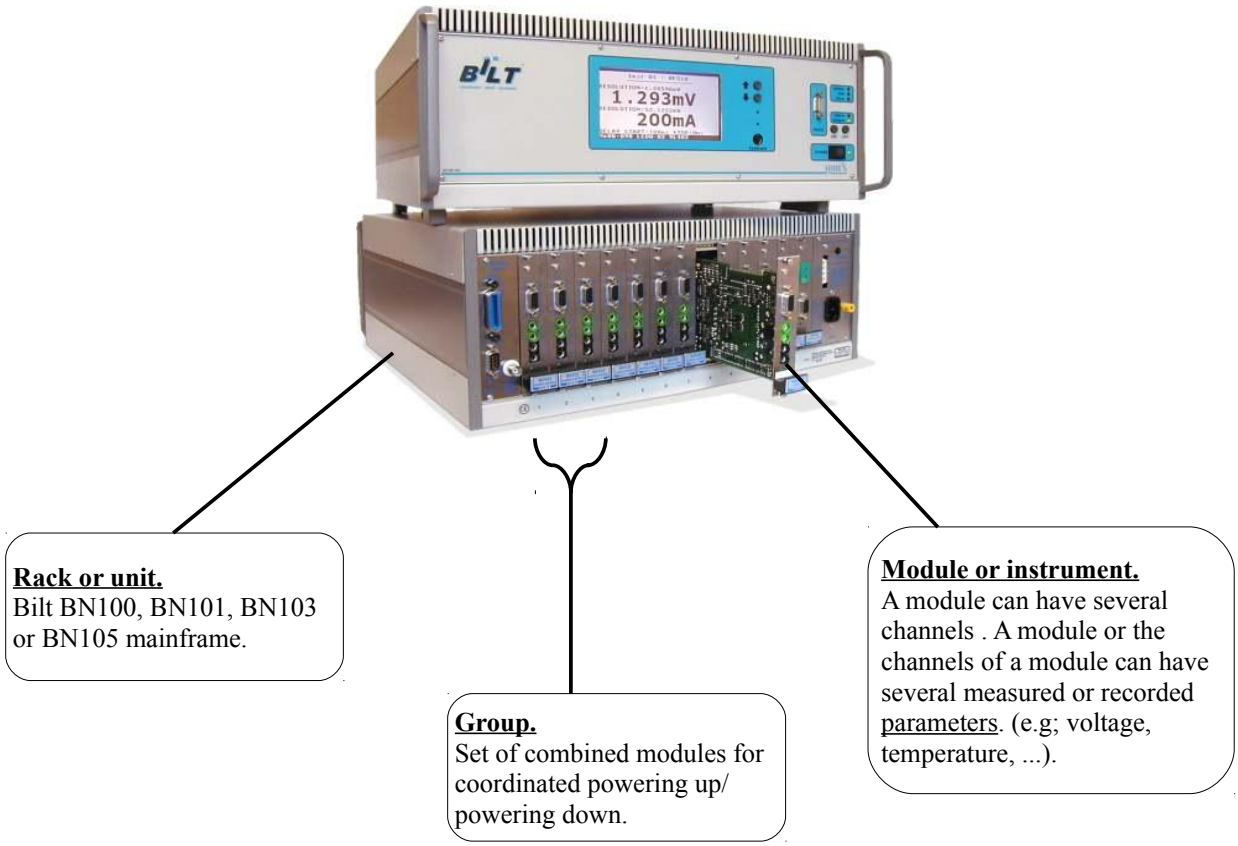
Programming Support :

Sébastien LAURENT : (+33) 05 61 54 81 37
sebastien.laurent@itest.fr

Version	*Date*	*Auteur*	*Modifications*
V2.00	01/2012	thomas	Update from v1.07

Contents

Vocabulary :



Note on multichannel modules :

There are two types of multichannel modules :

- Independent multichannel modules, called IMC modules, whose channels can each start and stop separately.
  → the main module has not an On/Off control.
  → each channel has an On/Off control.
- Non independent multichannel modules, called non-IMC modules, whose channels start and stop all together.
  → the main module has a global On/Off control.
  → the channels have not On/Off controls.

1 General presentation

1.1 Requirements

For development of programs, a terminal interface is required.

- If Bilt rack is equipped with a display (BE740 option), this display can be used as terminal by plugging a PS2 keyboard on the front panel.
- In all cases, a PC computer can be used thanks to the iTest "BiltTerminal" software provided.

Note : Under Microsoft Windows OS, "HyperTerminal" software can also be used.

1.2 SCPI commands overview

No matter what type of interface is used, the BILT system only accepts and returns ASCII character chains in an SCPI format.

SCPI stands for "***S***tandard ***C***ommands for ***P***rogrammable ***I***nstruments". The purpose of this norm is to standardize the instruments commands between programmable test and measurement devices from different manufacturers.

The syntax is based on the use of the most explicit keywords possible with a long and a short form (e.g. : the SCPI command **VOLTage** can be written **volt** or **voltage**) possibly preceded by a « root command » (e.g. **meas:volt**). If the command has an argument, it is placed after the keyword, separated by a space (**volt 12**). If the command must return data, the key word is followed by « **?** » (**meas:volt ?** ↵ → answer : 1.002).Whether on input or output, lines are ended by the LF end-of-line character (ASCII 10, represented by ↵).

The character « **;** » allows to remain at the same level of "routing" : for instance, commands **limit:upper 3.2** and **limit:lower 2.8** can be chained this way : **limit:upp 3.2; low 2.8**↵.
The characters « **;:** » allows to return to the root : **volt:range 12** and **curr:range 1e-3** can be chained on the same line : **volt:range 12 ;: curr:range 1e-3**↵

If a command needs several parameters (as input or output), they are separated with commas (',').

Bilt generally respects recommendations made by the SCPI consortium. However some syntax has been simplified or adapted to better match our product specifications.

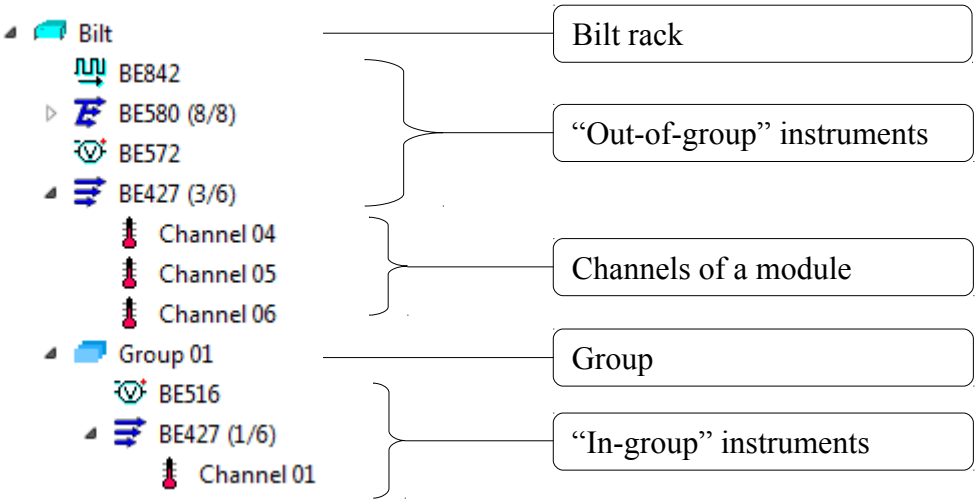
Note : SCPI commands are case insensitive.

2 BILT operating principle

Bilt is a modular system which can contain up to 13 instruments with various features. Each instrument has its own set of SCPI commands which can be retrieved in corresponding datasheet.

A widely used feature of the Bilt system is to use groups of instruments :
Several instruments can be affected to a group. It allows to create a set of instruments whose start/stops are coordinated. A start and stop sequence can be defined using delays between the main group start or stop and each module start or stop, and, moreover, it offers many functionalities like parameter memorization, cycling conditions, setting thresholds on measured parameters to automatically stop a test, triggering features, ...

As a consequence, a structure is created inside the Bilt unit, which can be represented like a tree :



Accordingly, each group, module and channel must be accessible independently :

- Modules are addressed thanks to the command "**iX**", where X stands for the position of the module inside Bilt rack :
  typing "**i3**" will cause the addressing of the following commands to the module plugged in the third position inside Bilt rack.
- Channels are addresses thanks to the command "**cX**", where X stands for the channel number X of current selected module :
  typing "**i3**" and then "**c2**" will cause the addressing of the following commands to the channel number 2 of the module plugged in the third position inside Bilt rack.
- Groups are addressed thanks to the command "**pX**", where X stands for the group number :
  typing "**p1**" will cause the addressing of the following commands to the group number 1.

Note : currently selected items can be retrieved thanks to the commands "**i?**", "**c?**" and "p?".

For example, imagine a case where a DC source module is plugged in position 4, a multichannel temperature controller is plugged in position 6 which channel number 3 is used. Both DC source module and the third channel of the temperature controller are affected in group number 1. The following command sequence could be used :

i4↵	→ selects the module in position 4 (the DC source)
VOLTage 1,5↵	→ the module in position 4 receives this command.
i6↵	→ selects the module in position 6 (the multichannel temperature controller)
c3↵	→ selects the channel 3 of currently selected module (i6 here)
MEASure:TEMPerature ?↵	→ The channel 3 of the module 6 receives this command and answers : "20,6↵"
p1↵	→ selects the group number 1
P:STATe:ON↵	→ The group number 1 receives this command and then starts. (i4 and i6;c3 starts synchronously)

Notes :

- commands could have been chained like : "i6;c3;meas:temp ?↵"
- group commands can be compacted like : "p1:stat:on↵"

3 BILT SCPI commands

3.1 General rack commands

Command	Comments	By default.
***RST**	**Completely** resets the rack: Stops all groups and modules Deletes groups and declared memories. It is best to wait 3 to 4 seconds before re-communicating with the rack.	
INST [val][?] *or* I [val][?]	Selects a module or reads the last selection	**1**
INST : LIST ? *or* I : L ?	Returns the list of instruments fitted to the rack: 1,Id ;2,Id...	
SYSTem : VERSion ?	Software version	
SYSTem : ERRor ?	Reads last error not yet read (fifo).	
SYSTem : VERBose [?][1/0/on/off]	(TCP/USB/RS232/RS422 only : in GPIB verbose is always OFF). In « verbose » mode : - Returns token type data in the form « 0, OFF » - Automatically returns error messages. - Replaces the argument separator « ; » in output by a line return. - The [LF] on output is replaced by [CR][LF].	**RS232 : on** **RS422 : off** **USB : off** **GPIB :off** **TCP : off**
SYSTem : TIMe [?][jj,mm,aa,h,mm,ss]	System date & time. Precision clock saved (operating on battery).	
SYSTem :POWer ?	Returns: Measurement +25 in Volts, Power +25 used in %, Measurement -25 in Volts, Power -25 used in %. Example : 25.3,72,-25.2,21	
SYSTem :POWer : MAX ?	Returns available power in watts on +25, on -25 (e.g. : 900,150)	
SYSTem : SERial : BAUD [val]	Programming of baud rate (bit par second).**Valid immediately !**. Possible value : 19200, 38400, 56000, 125 (=125k), 208 . The baud-rate is not altered by upgraded µC software.	**56000***(EP)*
SYSTem : SERial :ID [?][Val]	Programs or reads one 422 network address (two digits in hexa). The 00 and FF addresses are illegal.	2 last digits of serial number*(EP)*
SYSTem : GPIB :ADDRess [?][Val]	Programs or reads one GPIB address. The GPIB address is not altered by µC software.	**5***(EP)*
SYSTem:ETHernet:ADDRess [?] [a,a,a,a] **SYSTem:ETHernet:ROUTe [?][a,a,a,a]** **SYSTem:ETHernet:MASK [?][a,a,a,a]** **SYSTem:ETHernet:MAC [?]**	Programs or reads the TCP/IP address, gateway and mask. Reads the physical address (MAC). Example : SYST:ETH:ADDR 192,168,0,1;ROUT 192,168,0,255;MASK 255,255,255,0	192,168,0,1xx 255,255,255,255 255,255,255,0 *(EP)* xx=Serial No
SYSTem : NAME [?][String]	Assigns or returns the rack name (Max : 20 characters).	**"Bilt xx"***(SV)* xx=Adr 422
SYSTem : OVERPROT off	Removes over-voltage-protection for all modules which are equipped with it (Warning : An applied over-voltage on output of a module whose over-voltage protection is disabled may be destructive.)	
SYST :SET ?/SYST :SET :NEXT ?	Returns the complete set-up of the rack in the form of a series of commands which can be re-emitted as they are. In mode « syst :verbose :off », returns the first line by **SYST :SET ?** then the following lines one by one at each **SYST :SET :NEXT ?**. The last line returns «END ».	
Syst:key:def 1,[string][?]	Programs or reads the SCPI string executed by pushing the « F1 » button (new racks) or « ON » button (old racks). 80 c max.	**""(sv)**
Syst:key:def 2,[string][?]	Programs or reads the SCPI string executed by pushing the « F2 » button (new racks) or « OFF » button (old racks). 80 c max.	**""(sv)**

- (EP) = Saved in Eeprom, not affected by *RST command / (SV) = Saved in non-volatile RAM. Default value is restored by *RST.
- Unlike certain GPIB modular devices, BILT does not use the secondary GPIB address system to point a module.
- The parameters for each module are in saved RAM, and hence are not lost during powering down. They are only reinitialized when there is a physical change of the type of module or by the command « *rst ».
- Programming of GPIB 21 and 00 addresses is prohibited because they correspond to default addresses of the GPIB *HP* (21) and *National Instrument* (00) board.
- A rack which has been used with the iTest ***EasyStress***® software should be completely reinitialized by « *rst » before being used for instrumentation to avoid any surprises intrinsic to the programming of the groups of modules.
- A command allows to record a default programming for the F1 and F2 keys after the execution of a the *rst command. Please contact us.

3.2 Common commands for all the instruments

Command	Comments	By deflt.

INST [val][?] *or* I [val][?]	Selects a module, or reads the last selection.	**1**
***idn ?**	Identifies the module in the form ***num*, « *str* »** (e.g. : 510, « ITEST BE510GS… »). Apart from the type of module, contains the module's serial No; the date of the last calibration (LC) or the manufacturer's date-code (DC), as well as the software version (VL). For modules consisting of a mother-board and a daughter, the two are identified. Refer to the module technical sheet.	
DEFine [str][?] *or* NAMe [str][?]	Module name (20 characters max)	Ref module(SV)
STATe ?	Reads the internal state of the module1 :.),0,OFF ;1 ON ; 2 WARNING ; 3,ALARm On certain modules State may be the switching on/ switching off command («state on/off»).	
SET [:FIRSt] ? / SET :NEXT?	Returns the complete instrument set-up in the form of a series of commands which may be re-emitted as they are . In mode « syst :verbose :on » -conventional use as a terminal - « SET ? » returns in one go all commands separated by « ↵ ». In mode « syst :verbose :off », returns the first line by « SET ? » then the following lines one by one at each « set :next ? ». The last line returns « END ».	
STARt :TYPE [OR/AND][?]	Type of behavior for the module when it is declared in several groups. If start is of the type « OR », the module will be switched « On » as soon as one of the group in which it has been declared is switched « On » Otherwise (« AND » case) it will be switched « On » when all groups including are « On ».	**OR(sv)**
STOP :TYPE [OR/AND][?]	Type of behavior for the module when it is declared in several groups. If Stop is the type « OR », the module is left On » as long as at least one of the groups in which it is declared is « On ».	**OR(sv)**
IMC ?	Returns the type of module for multichannel module : • 1 → IMC = "Independent Module Channel". The On/Off operations are asynchronous for each channel. Each channel can be moved in separate group. • 0 → non-IMC = Mono or multichannel module which all channels are synchronous for On/Off operation (Ie: BE580).	
CMAX ? *or* VMAX ?	Returns the number of channel of the actual module. Returns 0 if the module has no channel.	
IDATa ? / IDATa :NEXT ?	Short-cut returning "Mx,State,Fail,Meas :Param 0 ,Meas :Param 1" where : • x is the instrument position • State : 0,OFF ;1,ON ;2,WARNING ;3,ALARM • Fail : token with maximum of four letters, giving the alarm status of the instrument, NO if nothing to declare .e.g. : NO, LOW, HIGH, TEMP... The multichannel boards return a line per channel and x takes the form instrument/No / channel N° (example for the module 6 channel 2 : 6/2). If the commands exceeds 100 characters, it returns NEXT after the current line. The rest may be obtained by the command IDATA :NEXT ?	
PARAM ?	Returns the list of parameters which can be memorised (see MEMory :…) and expected in the commands IDATa et MDATa. Syntax : No, « name », unit, Type of mem1, type of mém2... ; ...	

SV : Parameter in permanent memory

3.3 Groups commands

The definition of groups of instruments is used :

- To start/stop an x number of modules in a consistent way (**PROGram/STATe ON/OFF**), with pre-programmed delays between each module.(**STARt/STOP :DELay**).
- To manage a test time counter (**PROGram :TIMe**).
- To stop a consistent set of instruments when a measuring threshold is exceeded, by external triggering or by a delay command.
- To create memories in order to trace module measurements (**MEMory :…**)
- To get a test log (**HISTory :…**) : time-tagged text file tracing powering up, stops, exceeding of program thresholds, mains cut-off….
- …

3.3.1 Main group commands

Command	Comment	By default.
PROGram [*val*][?] *ou* P [*val*][?]	Selects a group The group remains selected as long as no other one has been selected. Max 12 groups.	**1**
P :DEFine / P :CLEAR :DEF	Creates or deletes the definition of the group. The definition can only be deleted if no other module is linked to the group.	
P :NAMe[str][?]	Group name (20 characters max)	**PROGRAM xx** [SV]
P :INST [:LIST] ?	Returns the list of modules attached to the current group	
P :INST:LIST VERBose ?	Returns the list of modules and channels attached to the current group under the form : Inst, Inst, Inst-channel,Inst-channel,Inst, … Example : 2,3,4-2,4-3,6 = modules 2,3,6 + channels 2 and 3 of module 4	
P :INST :ADD/DEL	Adds/Removes the current instruments of the current group.	
P :INST :LIST No[-Chan], No...	Replaces the composition of the current group with the previous list. Use '4-3' syntax to attach channel 3 of module 4.	
P :CHAN:ADD/DEL	Adds/Removes the current channel (chan x) of the current IMC instrument (I x) from the current group (p x). Example : **p2;:i4;:chan6;:p:chan:add** : Attach channel 6 of instrument 4 to group 2.	
P :CHAN :LIST No, No, No	Chooses the active channel in the current group (p x) of the current IMC instrument (I x). Example : **i4;:p2:chan:list 2,5** : Attach channel 2 and 5 of instrument 4 to the group 2. All other channels are deselected for this group.	
P :STATe [on/off][?]	Switches the common group with synchronisation of modules between them according to their STARt/STOP :DELay. It is possible to switch on the group only if its state is **Off** . To switch state form **alarm** or **time-stop** to **Off**, send **P:STATE :CLEAR.** The query form return : **0,ND ;1,OFF ;2,ON ;3,WARNING ;4,ALARM ;5,TSTOP**	**off**
P :STATe :CLEAR	Used to switch the ALARM state to OFF, or WARNING to ON. Indispensable for restarting a group after an alarm . (note : To leave the TSTOP state time stop must be increased or the memory dumped)	
P :LIMit [0/1/on/off][?]	Activates threshold monitoring for this group.	**Off** [(SV)]
P :TIMe : STATe [0/1/on/off] [?]	Activates the time counter and memorisation for the current group . The time and memorisation will be valid if the group is, moreover, On (P :STATe ON)	**off** [SV]
P :TIMe [hh,mm ,ss][?]	Reinitialises/Reads the group test time Max 9999 hours.	**0,0,0** [SV]
P :TIMe :STOP [hh,mm,ss][?]	Programs the code marking end of test . Max 9999 hours.	**1000,0,0** [SV]
P :NTESt [*val*][?]	RUN number.	**1** [SV]
P : HISTory [:NEW] ?	Returns a new log line since the last query (if any). If there are several lines then several queries should be made.	
P : HISTory : ALL[?][:NEXT ?]	Returns all log lines. If the log is longer than four lines, NEXT is passed on the last line. The following lines are available by P :HIST :ALL :NEXT ?	
P : HISTory : ADD string	Used to enter a « user » line in the log file.	
P :MEMory :CLEar	Reinitialises memories, log and time counter for this group.	
PDATa [val] ?	Short-cut returning the following data : G(group n°),(new),(groupe state),(groupe time) Example : « G1,0,2,0010h20m45s »	
MDATa [val] ?/ MDATa :NEXT?	Short-cut returning data for « pdata ? » for the group pointed at according to the data in « idata ? » for each group module.	
P:LIST?	Returns the list of defined groups (ex : 1,3,4).	

[SV] : Parameter in permanent memory

3.3.2 Memorization features

The memories described here are measurement tables. Each measure point is separated by a time interval. Memories

are defined in relation to the current group (**PROGram**), the current module (**INST**) and eventually the channel (**CHANnel**) for multichannel modules.

Usual types of memory :

- **Infinite memories :**

These memories will store measured values from the beginning of the test till the end with a constant number of points. To do this, when all points had been stored, the system will remove one measure over two and set the period twice longer. These memories are used to view parameters evolution throughout the entire test.

INFS memories : "Self Compress Memory Sample" - memorizes a point. For re-dimensioning only the second point of each pair is kept.

INFX : "Self Compress Memory Envelope" - simultaneous memorizing of two points separately (Min/Max). For each re-dimensioning only the extreme point of each pair is kept. (the smallest of the min, the largest of the Max). Thus the envelope remains consistent even after 1000h of testing.

- **Roll memories :**

These memories will store a moving, fixed time width memory. The number of points and the period are constant, so, each new measure will erase the latest one. These memories are used to view the end of the test.

ROLS memories : "Roll memory Sample" – memorizing of a point.

ROLX memories : "Rol memory Envelope" – simultaneous memorizing of Min and Max separately.

Command	Comment	By default.
MEMory [<name>][?]	Selects the <name> memory or returns the currently selected memory corresponding to the current Instrument/Program context.	
MEM :DEFine <name> , type , points , période , paramètre	Name : String 8 characters max. Type : A type supported by the module INFS/INFX/ROLS/ROLX… Points : 64,128,256,512,1024,2048 for envelope (INFX/ROLX) 128,256,512,1024,2048,4096 for sample (INFS/ROLS) Parameter : Parameter of the module to be memorised (see PARAM ?)	(SV)
MEM :DEFine ?	Returns definition parameters for the current memory.	
MEM :LIST ?	Returns the list of memories defined for the currently selected group/modules	
MEM [name] : CLEar : DEF	Frees memory linked to a name and to a currently selected pair.(I/P) (opposite command to define)	
MEM [name] : DATA [:FIRST] ? **MEM [name] : DATA : NEXT ?**	Returns the first block of memory available . Returns the following data block. Format for memories ROLS,ROLX,INFS,INFX,IOL : • Header : Period in ms (first block only) • For single point memories (ROLS,INFS,IOLS) : val ;val ;val... • For two-point memories (ROLX,INFX) : min,max ;min,max ; min... END as last argument (last block) The blocks contain at most 40 measurements (20 pairs for two-point memories).	

[SV] : Parameter in permanent memory

Advanced types of memories :

- When using group's cycling features, another type of memory is available : IOL memories. Please see ch. *#3.3.4.Cycling features* for further details.

- In conjunction with macros, yet another type of memory is available : TRIG memories. Please see ch. *#3.3.5.Macro features* for further details.

Notes :

- There is no command for reinitializing memories. This is done by the group command.« **P :MEM :CLEAR** ». (which also clears the group time counter and the test history).
- More specific memories can be available on some modules. Their explanation must be found on the module datasheet.

Specific type of memory : EVENT

The type of memory 'event' stores a set of measures each time the module asks - usually at a measurement variation. The measure at the origin of the record is marked, and the record is dated. In addition, the system creates start and stop records.

For example, BE437 module is a contact tester. It measures the contact resistance, detects, counts and measures the width of µ-cuts. This module can create a new record both on resistance variation or on µ-cuts detection. Example of BE437 event memory :

Record No	Test Time	RESistor	PULSe	WIDTh	Event Type
001	0000h00m01s	17.745mΩ	0	0s	Start
002	0000h00m08s	346.23mΩ*	0	0s	Mesure 1 event
003	0000h12m16s	348.71mΩ	1	400ns**	Mesure 2 event
004	0000h52m01s	347.12mΩ	1	400ns	stop

(*) The module has a programmable threshold and ask to memorization only if the measure variation is greater than this threshold.
(**) On this module, the width can not be the cause of event.

To create event memory :
→ ***mem event1,event,128,1,PULse***

Only one event memory can be created per module (per channel for IMC modules).
No matter the parameter used to create the memory, the memory takes all memorizable parameters.
The only possible number of records is 128.

To read event memory (this sample) :

i8	Module selection (No 8 for example)
mem event1	Memory selection
mem:data? → ***RESistor;PULSe;WIDTh***	First line : table header (only measures).
mem:data:next? → ***"R";" ";"s"***	Second line : Unit of measures.
mem:data:next? → ***512;0000h00m01s;17.745;0;0***	Data line : cause, time, measure, measure... cause : No of the measure at the origin of the record, or 512 → start / or 256 → stop.
mem:data:next? → ***1;0000h00m08s;346.23;0;0***	
mem:data:next? → ***2;00000h12m16s;348.71;1;4e-7***	
mem:data:next? → ***256;0000h52m01s;347.12;1;4e-7;END***	

in some cases, measures of last module (position 13 for std chassis) can be automatically added to module measures in memory event. Commands Mem:data?/next? still works on the same scheme.

3.3.3 Cycling features

The principle is to alternatively switch .the modules of a group On and Off according to program times (in seconds, between1 and 250s). The ON state corresponds completely to normal operation of the group. However the Off state raises some difficulties in relation to the completely powered down state :

- The output short circuit relays on DC sources are not activated.
- The zero voltage may be slightly effective (offset),specified for each module.

The program sequences for each module (start/stop delay) are expected for each switching on and off. Monitoring of threshold during the On phase after the program stabilization delay is also covered

For modules featuring memorization possibilities, a new type of memory is available : the IOL_SAMPLE memory. It operates as an infinite compression memory but the period of initial memorization is the cycle, then, after compression, every two cycles, then, every four cycles, etc. The user can adjust the moment of the cycle that he wants to memorize. This moment can be programmed in seconds starting from the beginning of the On phase. As for the other types of memory, one must use the number of graph points between 128 and 4096. The system does not offer any envelope memory possibilities for this type of memorization.

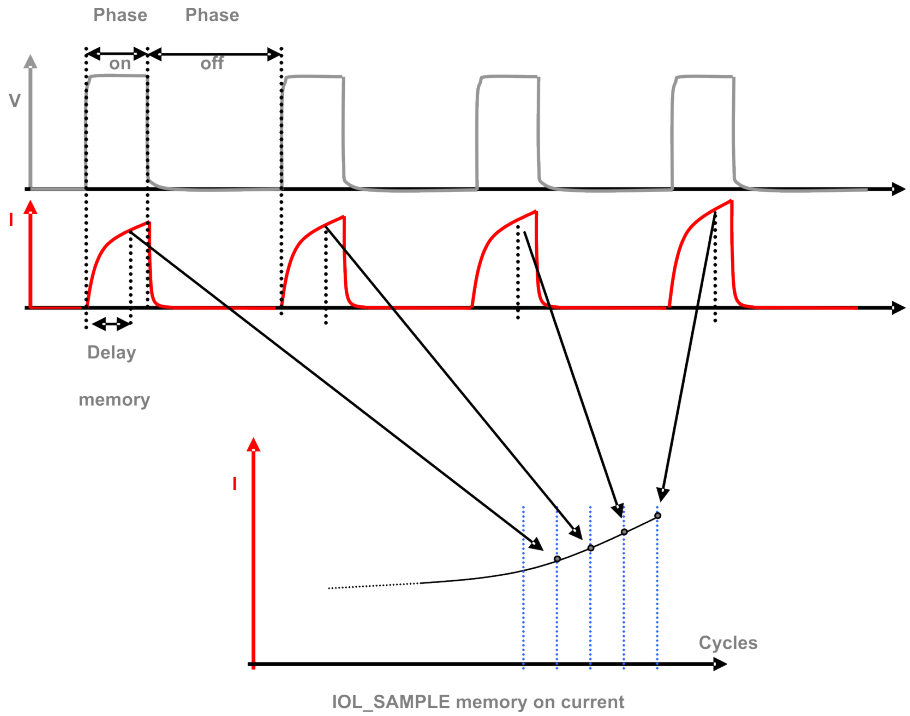
The initial state of the cycling may be programmed. The test can start by the On or the Off phase. This possibility used together with starting of all the groups of a rack (using trigger feature), is used to test two groups in phase opposition

The stopping is asynchronous for cycling in all cases (operator stop threshold or time delays).

IOL commands :

Command	Comment	Par déf.
P :IOL :STATe [0/1/on/off][?]	On/Off cycling state for selected group	**0**
P :IOL :TON / P :IOL:TOFF [val][?]	On/Off cycling phases duration in seconds (max 250)	**60** (SV)
P :IOL :FIRSt [0/1/off/on][?]	Initial state	**on** (SV)
P :IOL :SET ? **[:NEXT? (until END, verbose off only)]**	Returns the complete cycling configuration of current group.	

Example of On/Off cycling with memorizing of a cycle moment :



IOL_SAMPLE memory on current

3.3.4 Macro features

- Standard Macros :

A macro is an SCPI instructions line saved and executed by Bilt's CPU card on specified instant(s) of the test.
→ A macro belongs to a group, there's 3 macros available per group.
→ Macros are started when both “p:state on” and “p:time:state on” conditions are met.

Command	Comment	Par déf.
P :MACro **[No, Exist, DelayTime, LoopTime, "SCPI cmd"]** **[?]**	No = macro number (0, 1 or 2). Exist = 1 if active, 0 if inactive, DelayTime = delay before first execution (in hundreds of ms). LoopTime = duration between each execution (in hundreds of ms). (0 → no loop) SPCI cmd = SCPI instructions line to execute.	**–,0,0,0,""** (SV)
P:MACro:ATRestart Type	Behavior on group restart whithout time reset : Type = 0 → continue. Type = 1 → reset. Type = 2 → restart last macro executed, then, continue. (common command for each macros).	**0** (SV)
P:MACro:REINIT	Resets the time counters of each macro. (common command for each macros).	
P:MAC :SET ? **[:NEXT? (until END, verbose off only)]**	Returns complete macro configuration of current group.	

Example :

p:mac:set ?↲
p:mac0,1,10,36000,"i13;:temp 20"↲
p:mac1,1,18010,36000,"i13;:temp 125"↲
p:mac2,0,0,0,""↲
p:mac:atrestart reinit↲
END↲

→In the example above, a temperature controller (i13)
→will cycle the temperature every hours between
→20 and 125 °C.

→ On macro execution, a dated event is recorded in group history (See p:hist command). When macros are looped, a single record is produced on the first execution (after applying specified delay).
→ Macros can be programmed with a recurrence that can go down to 100ms. However, no guarantee is given on the accuracy, and, in some cases, you can saturate Bilt's CPU and make the system unstable.

- Advanced macros :

The macros also allows to modify the setup of one or several modules at run-time using computed or predefined values, stored in variables.

The system offers an area of 100 locations to store variables, using floating format, and 10 locations to store indexes, using integer format.

Variables can be accessed by their absolute address (range from 00 up to 99) or thanks to indexes. (which themselves are accessed by their number, from 0 up to 9).

Variables and indexes are in permanent memory, they are set by “*rst” command to 0.

WARNING : the data area is shared by all groups, and therefore all macros inside Bilt rack. Special attention must be given on the data accessed for each macro defined !

Command	Comments	Def.
VAR[val1],[val2]	update data located @[val1] with value[val2], [val1] address ranges from 00 up to 99 [val2] data format uses standard floating values	
VAR[val1]?	read data located @[val1] [val1] address ranges from 00 up to 99	
VAR:IND[val1],[val2]	update index N°[val1] with value[val2] [val1] index number ranges from 0 up to 9 [val2] data format uses integer values from 00 to 99	
VAR:IND[val1]?	read index N°[val1] [val1] index number ranges from 0 up to 9	
VAR:IDM[val1]	setting maximum index value [val1] data format uses integer values from 00 to 99 (useful when running increment function)	
VAR[val1]:ADD[val2]	add the data located @[val1] with [val2] [val1] address ranges from 00 up to 99 [val2] data format uses standard floating values	
VAR[val1]:MUL[val2]	multiply the data located @[val1] with [val2] [val1] address ranges from 00 up to 99 [val2] data format uses standard floating values	
VAR[val1]:DIV[val2]	divide the data located @[val1] with [val2] [val1] address ranges from 00 up to 99 [val2] data format uses standard floating values	
VAR 00:FILL[val1],[val2],...	update data table starting at address 00 with the list of values separated by a comma (data format uses standard floating values	
VAR[val1]:IFEQ[val2],... **VAR[val1]:IFLE[val2],...** **VAR[val1]:IFGE[val2],...**	Stops the SCPI parser if @val1 don't respects te condition. (IFEQ = if equal, IFLE = if lower or equal, IFGE = if greater or equal) Example : *var8:ifle 4,9;add 0,1* → if (@8 <= 4,9) then (@8 += 0,1).	
VOLT#[val1]# **or CURR#[val1]#** **or LIM:UPP#[val1]#** **etc...**	any standard module commands can be used using indirect data values instead of absolute data : at run time, #[val1]# will be replaced by the value of the data located @[val1]. [val1] address ranges from 00 up to 99	
VOLT#i[val1]# **or CURR#i[val1]#** **or LIM:UPP#i[val1]#** **etc...**	any standard module commands can be used using indirect indexed data values instead of absolute data: at run time, #[val1]# will be replaced by the value of the data located at the address indicated by the index N°[val1]. after completion, the index N°[val1] will be incremented : +1. If the maximum value specified by the VAR:IDM command is reached, then the index will be reset to 00. [val1] index number ranges from 0 up to 9. This syntax allows reading a table from 00 to MAX then cycling.	
VOLT#d[val1]#	same thing, but, after completion, the index N°[val1] will be decremented. If the minimum value 00 is reached, then the index will be set to the value specified by the VAR:IDM command This syntax allows reading a table from MAX to 00 then cycling.	
VOLT#x[val1]#	same thing, but, after completion, the index N°[val1] will be incremented until the maximum value is reached, then decremented until the minimum value is reached. This syntax allows reading a table from 00 to MAX and MAX to 00 then cycling.	
VAR:AutoReset	Sets or reads the state of index autoreset system. For auto-increment/decrement commands of indexes, if var:ar is programmed to Off, the decrement/increment will stop at the last value. Otherwise, index will be reset to the initial value and decrement/increment will continue.	

- Triggered Infinite Memories

This function allows the user to define the sample position of a memory while a macro is running.

Principle of memory "TRG" :

For each group, there are three infinite memories with external trigger (TRG1/TRG2/TRG3) which corresponds to three different instants of memorization.
As all other memories (INF, ROL ...), the TRG(x) ones are defined in reference to a module (a channel) and a parameter (eg : Module3, channel3, TEMPerature).
It's possible to define several memories of the same type (TRG**1** for example) in the same group : for instance, one for the module 1 on current and another for the module 2 on voltage.
The "triggered measurement" will be updated at the same time for both of them : at the execution of the SCPI instruction "*mem:trig 1*". This instruction can be included in a macro.

Memories TRG(x) are infinite sample memories which operate exactly as standard INFS memories. They are synchronous of the group, with an initial storage period of 28.125 seconds, multiplied by two at each compression. The difference is the memory point value : it is not the current value of measurement but the last "triggered measurement" of the module, triggered by the command line "*mem:trig (x)*".

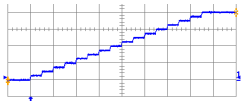
Notes :

At each start of the group, the "triggered measurement" of memory is reset to the value 0,00. Accordingly, as long as no update is made by the command mem:trig x, it is the value 0,00 which is memorized.

The SCPI command "*mem:trig (x)*" is referenced to the group.
The normal usage is "*p(GroupNo);mem:trig (TrigNo)*", however, this selection of the group is unnecessary in macros because the group is pre-selected by the system.

- Examples :

– ***10 voltage steps starting from 2V up to 20V, 2s step duration, stop when completed***



method 1 : **using constant increment value** (allows unlimited slew rate & time values)

MACRO1 / DELAY=0 / no loop : ***var 01,2***
operation : initialize data located @01 = 2

MACRO2 / DELAY=0s / LOOP=2s : ***i1;:volt #01#;:var01:add2***
operation : set module 1 voltage with the data located @01
then, add 2 to the data located @01

MACRO3 / DELAY=20s /no loop : ***p1;:p:stat off***
operation : stop group

note:
as data located @01 is used with this group, it is requested to use different addresses for other groups.
macro 3 can be replaced by the time stop function (parameter “step duration” in group properties)

method 2 : **using predefined data values** (allows arbitrary waveform)

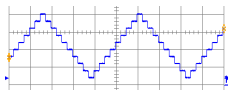
MACRO1 / DELAY=0 / no loop : ***var:ind 1,0;idm10;:var 00:fill 2,4,6,8,10,12,14,16,18,20***
operation : initialize index 1 = 0
set maximum index value = 10
initialize data area starting at location 00 with values 2,4,6,8,10,12,14,16,18,20

MACRO2 /DELAY=0s / LOOP=2s : ***i1;:volt #i1#***
operation: set module 1 voltage with the data value located at the address given by index 1
then, post increment index 1 = +1
if index 1 = maximum value = 10, then reset index 1 = 0

MACRO3 / DELAY=20s / no loop : ***p1;:p:stat off***
operation : stop group

note : as index 1 is used with this group, it is requested to use different index for other groups.
The data table always starts at address 00 and all the group use the same table.

– ***10 step voltage saw tooth, up to 20V and down to 2V then cycling, 2s step duration***



MACRO1 / DELAY=0 / no loop : ***var:ind 1,0;idm10;:var 00:fill 2,4,6,8,10,12,14,16,18,20***

operation : initialize index 1 = 0
set maximum index value = 10
initialize data area starting at location 00 with values 2,4,6,8,10,12,14,16,18,20

MACRO2 / DELAY=0s / LOOP=2s : ***i1;:volt #x1#***
operation: set module 1 voltage with the data value located at the address given by index 1
then, post increment or decrement index 1 = +/- 1
when index 1 = maximum value = 10, then change slope direction : decrement
when index 1 = minimum value = 0, then change slope direction : increment

note:
as index 1 is used with this group, it is requested to use different index for other groups.
The data table always starts at address 00 and all the group use the same table.
The macro1 is identical with the previous example.
The only difference is the command x inside macro2.
Using d command instead of x, gives a constant decreasing slope 20,18,16... 4,2,20,18,16 etc...

3.4 Triggering features

Bilt produces/listens Trigger on two medias :

- Using a discrete logic signal via a BNC jack (TTL levels, Open collector).
- Over the LAN, thanks to UDP broadcast commands.

Bilt Trigger system have two different purposes :

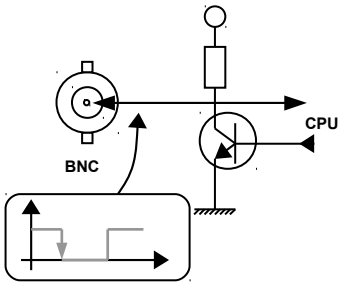
- For synchronous validation of armed commands on several modules. We call this purpose ***'Module Trigger'***. The Module Trigger accuracy is of the order of +/-1 ms.
- For link start/stop groups of modules, inside the same chassis or with others. We call this purpose ***'Group Trigger'***. The Module Trigger accuracy is of the order of +/-10 ms - configuration dependent.

It is recommended to not mix the two features.

Warning : ***'Module Trigger'*** is not available on BE718 CPU boards before the 'F' version. Trigger function - as explained in this document - are present from version 4.5.00 of BE718 firmware (can be released).

3.4.1 Trigger through BNC Jack

Open collector with 3kΩ pullup / TTL levels.
Isolated from chassis ground (common ground with RS422 interface).



'Module Trigger' usage :
You have to connect the Trigger-in signal to the backplane signal via the **syst:trig:source ext** scpi command.
Then, it is a falling-edge active signal with a minimum width of 10µs. Features depend on modules specifications/programming.

'Group Trigger' usage :
The system decodes and produces negative pulses :

- 15 ms ± 2.5 : start trigger.
- 5 ms ± 2.5 : stop trigger.

3.4.2 UDP LXI-Trigger over Ethernet

It is a broadcast UDP packet to synchronize several chassis over the LAN.
The host and remote port must be the 5044 (LXI-Trigger).
Bilt accepts full broadcast (ie:255.255.255.255) or mask limited broadcast (ie:192.168.0.255).
The packet format is very similar to the LXI format, as described in the 'LXI Triggering, revision 1.0' white paper.

3 Bytes	Packet Header, 'LXI', ASCII.
1 Byte	Domain. 0-255. Programmable by **'syst:trig:udpdom 0-255'** command. The chassis executes only commands within its domain.
16 Bytes	Event ID. String, left aligned, filled with 0. LAN0 : ***(trigger module)*** produces a pulse on the Bilt backplane trigger*. LAN1 : ***(trigger groupe)*** Starts groups with trigger-in/start setting. LAN2 : ***(trigger groupe)*** Stops groups with trigger-in/stop setting. BILT ? : Authentication request.
4 Bytes	Sequence number. Must change (incrementation) at each new packet. It permits double transmission detection (Bilt ignores commands with same seq-number that previous). Useful for command reception checking : Use **'syst:trig:udpseq ?'** Command (in TCP) to read the sequence number of the last packet received.

12 Bytes	Timestamp/Epoch. Not used. Filled with 0.
2 Bytes	Flags. Always 0x0004 (hardware trigger)
2 Bytes	Packet Terminator : 0x0000

(*) if programmed for : See **syst:trig:source ...** scpi command.

Total length : 40 Bytes.

3.4.3 UDP LXI-Trigger usage for Bilt authentication over Ethernet

The LXI-Trigger command with 'BILT?' in EventID field is for Bilt authentication over LAN.
Suitable chassis (right broadcast mask, right LXI domain) respond a mask-limited-broadcast packet like above, with 'BILT *xx-yyy*' in EventID field, where xx-yyy is serial number of their CPU board.
User can extract IP and MAC address from the UDP packets to know who is where.
It useful for the discovery of Bilt chassis on the local LAN.

3.4.4 About UDP packets reliability...

Original white paper from LXI consortium :
Network-savvy engineers will know that UDP data transmissions, which are the basis of LXI LAN trigger packet transmissions, do not guarantee packet delivery. LAN traffic problems can cause packets to be lost in transmission, and LAN configuration problems can cause packets to arrive more than once. The TCP protocol handles these problems with a very high probability of success, but the UDP protocol makes no attempt. How can this be handled in an LXI test system?
It is important to realize that modern network configurations minimize these problems. A test system that uses a small unrouteable subnet (probably the dominant configuration) with a modern LAN switch will virtually never experience packet loss or multiple deliveries. This type of test system will probably never experience problems of this type and need not go to any lengths to account for them.

Itest has developed a light software to test the UDP-LXI-Trigger reliability of the targeted LAN. Ask for it.

Note : On secure virtual LAN, switches classically transmit packets only to knowing servers. As Bilt does not support DHCP commands, at power-on, it produces broadcast ARP commands “i am xxx at yyy” as long as there is no incoming answer (or query). Then, routers and switches must maintain proper ARP table entries to be sure that UDP (or TCP) packet is well transmitted to the recipient.

3.4.5 Trigger SCPI commands

Command	Comment	RST value.
SYSTem:TRIGger:SOURce [*media*][?]	Sets the source of the backplane trigger (***module trigger***) : **- 1 / ext** : BNC-in. **- 2 / lan** : Lan UDP command. **- 3 / extlan** : Both. **- 0 / none**.	**none** (SV)
SYSTem:TRIGger:UDPDOMain [0-255][?]	LXI Trigger "Domain". Useful to split a sets of chassis on the same lan. Bilt performs only trig within its domain.	**0** (SV) (*)
SYSTem:TRIGger:UDPSEQuence [?]	Read-back of the 'sequence No' field of the last UDP trig command. Useful to check the command reception.	
SYSTem:TRIGger:UDPLAST [?]	Last state of Lxi trigger parser. Useful for program developer. **- 1** : Bad length (must be 40). **- 2** : Bad LXI header. **- 3** : Bad Domain (not necessarily an error) **- 4** : Flags is not 0x0004 **- 5** : Terminator is not 0x0000 **- 6** : Bad EventId string (must be 'LAN*x*' or 'BILT?') **- 7** : The sequence number is the same that the previous. **- 10** : Command LAN(x) wirh x > 2 **- 20** : Valid Module Trigger received. **- 21** : Valid Group-Start Trigger received. **- 22** : Valid Group-Stop Trigger received. **- 23** : Valid authentication request received.	
P[No]:TRIGger:STATe [*in-sensibility*][?]	Sets the group (No) sensibility to the trigger-in (***groupe trigger***). **- 1 / start** : Every incoming start-trigger will start the group **- 2 / stop** : Every incoming stop-trigger will stop the group. **- 3 / startstop** : Both. **- 0 / none**.	**none** (SV)
P[No]:OUTPout:TRIGger [*out-behavior*][?]	Sets the trigger-out options for the group (No) (***groupe trigger***). **- 1 / start** : Every start of the group will produce a Start-trigger. **- 2 / stop** : Every stop of the group will produce a Stop-trigger. **- 3 / startstop** : Both. **- 0 / none**.	**none** (SV)
P[No]:TRIGger:SOURce [*media*][?]	Sets the media of the ***group trigger*** for both *in* and *out* operations. **- 0 / ext** : BNC-in. **- 1 / lan** : Lan UDP command.	**ext** (SV)

(*) The LXI Trigger 'domain' is a chassis attached data, memorized in flash memory. The value remains the same even after reset (***rst**), a deep reset (**system:rst**) or a firmware release. The given default value is the factory value. As other chassis attached data, this command is not present in the **syst:set?/next?** File.

Note for group trigger : Within a given chassis, whatever the selected trigger media (***p:trig:source ext*** *or* ***lan***), starting a group with start option selected for trigger out (***p:outp:trig start*** *or* ***startstop***), starts all groups that have start option selected for trigger in (***p:trig:state start*** *or* ***startstop***). Same scheme for the stop sequence.

3.5 Managing alarms and threshold monitoring between groups and instruments

All the modules capable of measurements have programmable thresholds which can stop either the module or the group(s) in which they have been declared.

For “Out of group” modules, if the threshold monitoring is activated, they will stop on any threshold cross : module is in alarm state

“In group” modules, if the threshold monitoring is enabled, will just create a warning on threshold cross. If the group in which they are affected has the threshold monitoring disabled, the group (and therefore corresponding instruments) will continue the test. Otherwise, if the group has the threshold monitoring enabled, the group will stop.

Notes :

- the warning message of the module which generated the warning or alarm will ascend to the group in which its affected.
- If no modules inside group has any threshold activated, the group threshold monitoring command is useless.
- On DC sources the parameter should first of all be chosen (current or voltage) :The monitored parameter will be the one which is not regulated. To do this, you must program the expected mode : voltage regulation / voltage or current (FUNC CV/CC). On most sources this command has no physical effect : the modules have independent voltage and current settings (and measurements), and are regulated in one or the other of the modes according to the load.

Commands at module level :

Command	Comments	By default.
STATe ?	Returns the status: 0 or OFF → Off. Nothing to report . 1 or ON → On. Nothing to report . 2 or WARNING → Threshold exceeded but module is in a group whose threshold alarm is not validated. (**p :limit off**), and it is not stopped. 3 or ALARM → Threshold exceeded, module stopped.	
LIMit :STATe [0/1/on/off][?]	Activates threshold monitoring for this module.	**Off** (SV)
LIMit :FAIL ?	Returns the type of alarm which caused the stop. (LOW, HIGH…)	
LIMit :CLEar	Deletes and re-arms the thresholds/alarms.	
LIMIt :UPPer [val][?] **LIMit :LOWer[val][?]**	Programs the upper and lower thresholds. If you wish to monitor only one of the two, enter a value which is out of range for the other.	
LIMit :DELay[val][?]	Programs the delay after which the thresholds must be monitored (= measurement stabilization time). Generally in ms from 1 to 65000.In seconds on temperature modules.	

Group level commands :

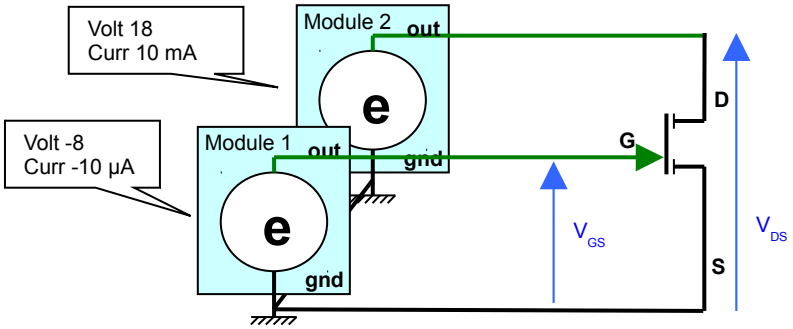
Command	Comment	By default.
P :STATe ?	Returns status : 0 or ND → Not defined. 1 or OFF → Off. Nothing to report. 2 or ON → On. Nothing to report. 3 or WARNING → one or several thresholds exceeded in a group but the threshold alarm has not been validated (**p :limit off**), and the group has not stopped. 4 or ALARM → Threshold exceeded, module stopped. 5 or TSTOP → Stop on end of test. Stop code.	
P :LIMit [0/1/on/off][?]	Activates threshold monitoring for this group.	**Off** (SV)
P :STATe :CLEar	Deletes and re-arms threshold for all modules and the group.	

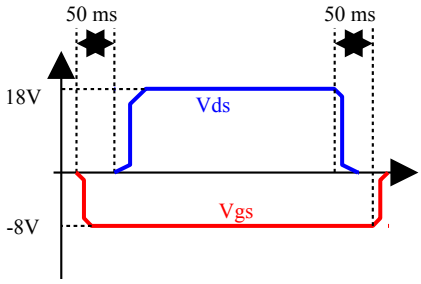
TO RESTART A GROUP STOPPED FOLLOWING AN ALARM YOU MUST FIRST ACKNOWLEDGE AND RE-ARM THE ALARMS. (P :STATE :CLEAR)

4 SCPI program samples

4.1 Creating group and programming start/stop delays between modules

The programmer has to polarize a FET transistor. The transistor must be blocked with -8 Volts. The expected leakage currents are about one µA on the gate and of one mA on the drain. Not to stress the component, the gate voltage must imperatively be present before, and disappear after, the drain voltage.

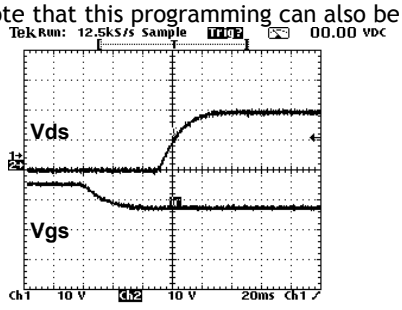




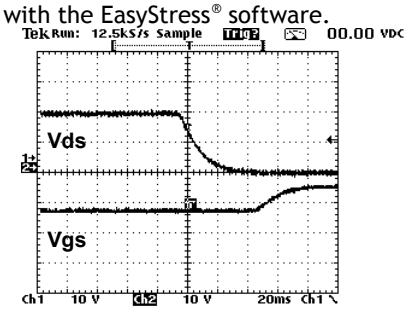
Two BE510 are used, in position 1 and 2 in the rack.

*RST	Rack reset, wait at least 4 seconds before starting again to communicate with the rack.
P1	Group 1 selection
P:DEF	Group creation
P:INST:LIST 1,2	Assignment of modules 1 and 2 at the group 1
I1	Module 1 selection
VOLT –8;CURR –10E-6	Voltage and current setting programming
START:DELAY 100; :STOP:DELAY 50	Start-delay of 100 ms (min value for BE510) and stop delay of 50 ms.
I2	Module 2 selection
VOLT 20;CURR 10E-3	Voltage and current setting programming
START:DELAY 150 ; :STOP:DELAY 0	Start-delay of 150 ms (50 more than module 1) and stop delay of 0 ms (50 less than module 1).
…	
P1:STATE ON / OFF	Power On/Off of the group of module.
I1;MEAS:CURR?;:I2;MEAS:CURR? → -2.2356E-7;8.456E-4	Current readback.

Please note that this programming can also be made very simply with the EasyStress® software.



p :state on



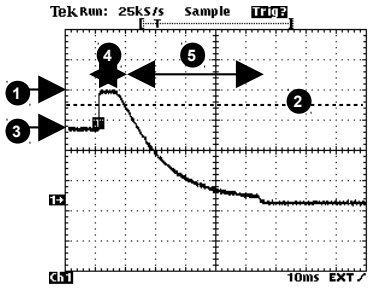
p :state off or main power defect or threshold

4.2 Activation of a safe stop on threshold breach

"Out-of-group" module :

A BE510 module, inserted into the position 10 of a Bilt unit, is used to powered a 5V system with nominal consumption of 2.5A. The module is programmed to stop if consumption exceeds 3A. Nevertheless, the system can consume up to 3.2A during the launching phase, and the threshold should be supervised only 1 second after starting.

*RST	Rack reset – wait at least 4 seconds before starting again to communicate with the rack.
I10	Module 10 selection
VOLT 5 ; CURR 3.5A	Voltage and current setting programming.
FUNC CV	FUNC CV/CC The function is used to choose which is the supervised parameter: - CC regulation awaited on current → threshold on voltage - CV regulation awaited on voltage → threshold on current
LIMIT :LOWER -5 ;UPPER 3	Threshold programming. The lower threshold is not used: an out of range value is entered. The upper threshold is programmed with 3A.
LIMIT :DELAY 1000 ;STATE ON	Threshold delay programming with 1000 ms. Threshold activation.
OUTPUT ON / OFF	Module power on/off
STATE ?	State control: 0 → Off, 1 → On, 2 → Alarm.



The oscillogram visualizes the current.: 1 → current limitation (3.5A), 2 → threshold (3A), 3 → system nominal consumption (2.5A), 4 → reaction time (hard limitation to programmed setting), 5 → hold time. After a stop on threshold, you can directly re-start with **output on** command, or clear the threshold before with **lim:clear**. If several thresholds are programmed, it is possible to know which one was crossed by using the **lim:fail?** command.

"In-group" module :

Continuing the example of group program (previous page), a monitoring of high threshold on the drain current of the transistor fixed to 2mA is added:

I2	Module 2 selection
FUNC CV;:LIM:LOW -5;UPP 2E-3	The high threshold of current is programmed with 2 mA. The low threshold is entered with a unreachable value.
LIMIT:STATE ON	Threshold activation – module level.
P:LIMIT ON	Threshold activation – group level.
To retrieve the group state:	
P:STATE ?	1 → Off, 2 → On, 3 → On and Warning (Threshold active and crossed on module level, but not active on group level), 4 → Off and Alarm (Stop on threshold)
After a stop on threshold, and before starting again:	
P:STATE:CLEAR	If not, the system refuses to start and out the error "Command incompatible with the current state"…