

BoxView



Debugger for ST18950

**BoxView for ST18950 User's Guide,
September, 2003**

Domain Technologies, Inc.
811 East Plano Pkwy, Suite 115
Plano, Texas 75074
Tel.: (972) 578-1121
Fax: (972) 578-1086
E-mail: support@domaintec.com
Web page: <http://www.domaintec.com>

Disclaimer of Warranty

This software package is provided on an 'AS IS' basis and without warranty. In no event shall Domain Technologies be liable for incidental or consequential damages arising from the use of this software. This disclaimer of warranty extends to LICENSEE, to LICENSEE's customers or users of products and is in lieu of all warranties whether expressed, implied, or statutory, including implied warranties of merchantability or fitness for a particular purpose. Domain Technologies does not warrant that software furnished hereunder is free of infringement of any third party patents, copyrights, or trade secrets.

TABLE OF CONTENTS

CHAPTER 1 - Introduction	7
1.1 - Quick Installation.	7
CHAPTER 2 - Tutorial	9
2.1 - Command line parameters	11
2.2 - Function keys	11
2.3 - Mouse Operation	11
2.4 - Sample session.	12
CHAPTER 3 - BoxView Reference	19
3.1 - Calls Window	19
3.2 - Code Window	19
3.3 - Command Window	21
3.4 - Memory Window	21
3.5 - Register Window	22
3.6 - Graph Window	22
3.7 - Symbols Window	23
3.8 - Trace Window	24
3.9 - Watch window	25
3.10 - Open connection dialog	26
3.11 - Startup Options	27
3.12 - Load File Dialog	28
3.13 - Hardware Breakpoints	29
3.14 - Software Breakpoints	30
3.15 - User Buttons	31
CHAPTER 4 - BoxView Commands	33
4.1 The command interpreter	33
4.2 Command line expressions	34
4.2.1 Expression terms	34
4.2.2 Unary operators	35
4.2.3 Binary operators	36
4.3 Programming the emulator with the command line interface	36
4.3.1 Defining macro commands	36
4.3.2 Programming	38
4.3.3 Memory and register access	39
4.4 File management.	40
4.5 General notes	42

CONTENTS

ADDR	43
ALIAS	43
ARCHIVE	43
ASM	44
BASE	44
BREAK	45
BYE	45
BEEP	45
CASCADE	46
CD	46
CALLS	46
CLOSE	47
CHANGE	47
CM	48
CODEMENU	48
CLS	48
COMPARE	49
COLOR	49
COPY	50
CONNECT	50
DEFINE	51
DASM	51
@CURX, @CURY	51
DISASM	53
DIR	53
DI	53
DM	54
DISPLAY	54
DISCONNECT	54
DO	55
DR	56
ENDM	57
EVAL	58
?	58
@EOF	59
EXEIO	59

FM	60
FOCUS	60
FILL	60
FORCE	61
FUNCTION	61
FOR.	61
GRAPH	62
GOCURSOR	62
GO	62
IDCODE	63
HELP	63
HALT	63
IF	64
INPUT	65
JUMP	67
LISTSYMBOL.	68
LINE	68
LOAD	69
LOCATE	69
LOG.	69
MB	70
MAP	70
MEM	71
NEXT	72
MM	72
MIX	72
OPEN.	73
OUTPUT	74
PATH	76
PAUSE	77
PCHIST.	77
PCCURSOR	77
QUIT	78
PRINT	78
RELOAD	79
RESET	79

CONTENTS

CONTENTS

RESTART	79
REFRESH	79
RUN	80
SAVE	80
RETURN	80
SB	81
SBDEL	83
SBPUT	84
SEARCH	84
SET	85
SLOAD	86
SM	86
SR	87
SRC	87
START	87
SOURCE	87
STEP	88
STEPOVER	88
STOP	88
STATUS	88
TIME	89
TIMER	89
UNALIAS	89
TILE	89
UNDEFINE	90
USE	90
VERSION	90
VIEW	91
WAIT	91
WATCH	91
WINDOW	92

CHAPTER 1 - Introduction

The development system for the ST18950 DSPs consists of the following components:

- Target hardware (DSPlug or ISA based emulator)
- BoxView — Debugger Graphical User Interface
- BoxServer (optional) server used for remote access

This manual describes the second component of the above configuration. Description of the other components can be found in the separate documents.

Software components need to be installed first. System software requires 32-bit operating system (Windows 95/98/NT). System software can not be run on earlier versions of the Windows PC operating system (Win 3.1, 3.11), even if the Win 32s option is installed.

1.1 - Quick Installation

Software is supplied on the installation disk. Execute program setup.exe, this will create all directories, copy files and install default shortcuts in the program group.

Installed software can be removed from the host computer using standard Windows' procedures (Add/Remove Programs at Control Panel).

BoxView can be invoked directly for the different target configurations without any additional changes.

1

1

CHAPTER 2 - Tutorial

Installation will create following shortcuts in the new program group:

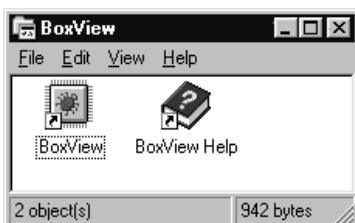


Fig. 1 - Shortcuts created by setup

- BoxView shortcut configured for the debugger execution.
- BoxView Help — this will invoke Help system

2

Multiple configuration can be created within the program folder. The debugger configuration is saved in the configuration file specified with the command line option -s within the shortcut properties. The default configuration file is called BoxView.ini. Custom configuration will have different command lines.

BoxView after invoking the program will initialize windows in the following setup:

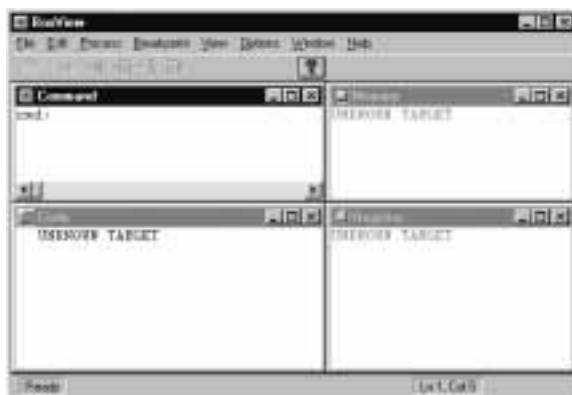


Fig.2 - Initial windows layout

Software can support various DSPs, so window contents can not be initialized before selecting the target device.

If the target device is specified correctly in the .ini file, simple CONNECT command can be used. Initialization process will take a few seconds. System is initializing connection with the target device. For customization or connection to the different target Open Connection dialogs needs to be used (available in Options pull down menu).

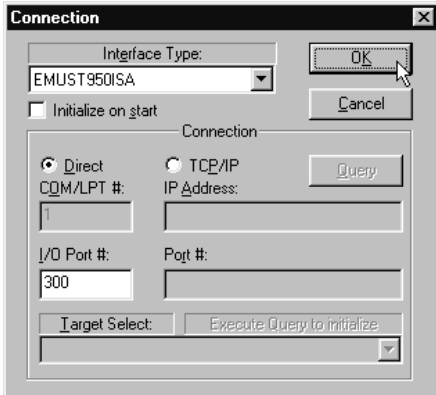


Fig.3 - Open Connection dialog

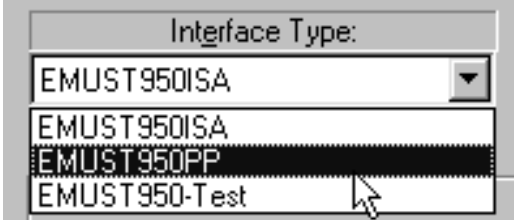


Fig.4 - Available Interface Types

For the initialization, proper Target type needs to be selected and LPT# or base I/O address specified. Above dialog also allow to select remote target DSPs. Remote target initialization procedure involves selecting BoxServer's IP address (default: boxserver.softbox.com), performing the "QUERY", to find out number of the devices, and devices types on the remote end. After selecting desired device from the list, system will connect and initialize its operation. Used devices will show IP addresses of the users connected through the BoxServer.

2

2.1 - Command line parameters

To allow target configuration at the program start-up, command line parameters can be specified. All target reference options, can be also set within the Open Connection dialog (Options pull-down menu).

For multiple configurations, it is recommended to use custom file name for the system "INI" file.

-Sname File name to save windows settings on exit

2.2 - Function keys

Defined standard function keys:

F1	Help
F5	Go
Shift-F5	Halt
F6	Activate next debugger window
F7	Go to the cursor (Code window only)
F8, F11	Step
F9	Toggle breakpoint at the cursor (Code window only)
F10	Next (jump over subroutines)

2

2.3 - Mouse Operation

The left-click sets the cursor within the window, right mouse click invokes local window menu, to set window options.

To facilitate running the target code, special mouse processing within Code window is added:

Double left-click toggles the breakpoint (same as F9 key).

2.4 - Sample session

Select target type the PC is connected to. For the DSPlug target, referenced as EMUST950PP, LPT number needs to be selected. For the ISA based emulator, I/O base address needs to be entered.

If the target is installed on the remote workstation, connection needs to be specified through the TCP/IP interface. After entering the IP address or alias name, and the port number used for the socket connection, Query button will initialize communication with the server. The Target Select box will show available targets at the remote site.



Fig.5 - Remote Connection to Server

When the OK button is pressed in the above dialog, system will start initialization to the target processor - local or remote.

If the problem is detected during interface initialization, error message will be displayed, and the Open Connection procedure needs to be repeated.

After connection to the target is initialized, the debugger will display the data from the DSP

2

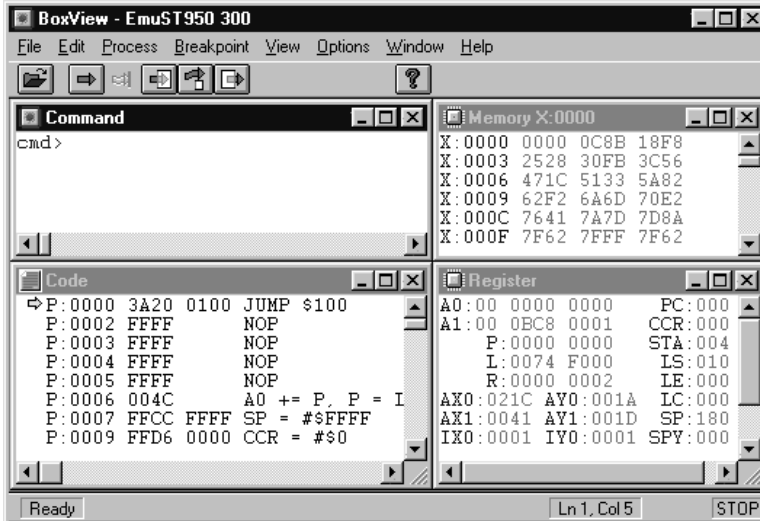


Fig.6 - After System Initialization

Next sample program can be loaded. To load the object file, “load” command can be used, but to specify additional load options Load File dialog should be used (pull-down menu LoadFile):



Fig.7 - Load File Dialog

The details of the above dialog are discussed in the following chapter. For the purpose of the tutorial, we will need only to select the file name, with the “File” button. This will open standard Windows file selection dialog:

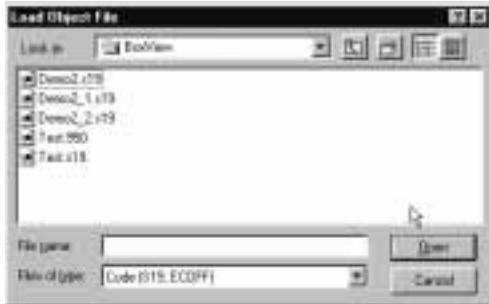


Fig.8 - File selection dialog

We will select sample ASM program — demo2.s19

2

- Set a breakpoint at “mainloop” with the “BREAK mainloop” command within the command window.
- Start code execution: GO
- Display data memory using symbolic entry: DISPLAY AM_WAVE

After those steps, debugger windows should look as follows:



Fig.9 - After Loading

Some of the above commands could be also executed using “Symbols” dialog. This dialog can stay open for the whole debugging session. To open this dialog, use View-->Symbol pull-down menu.



Fig. 10 - Selecting Symbols Window

2

Symbols dialog allows to set breakpoints, add items to the Watch window, set display address for the Memory and for Code windows, and start program execution to the temporary breakpoint.

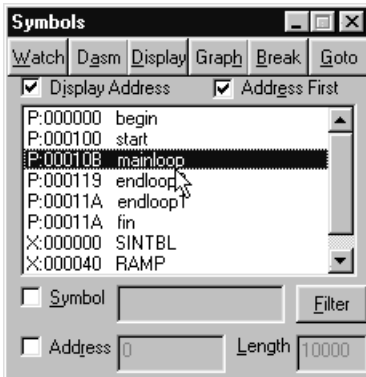


Fig. 11 - Symbols Dialog

Additional features of the above dialog are described in the following chapter.

To execute the program with the “automatic” screen refresh, set a breakpoint at address P:110 with the “show” attribute. This can be done with double-click or function F9 key, while the Shift key is depressed. Also “show” attribute can be set with the Software Breakpoint dialog. The address p:100 is inside the outside loop. Every time the execution stops at this address, sine wave was updated at Y:1000 address. To observe the sine wave in the graphical format, execute: GRAPH AM_WAVE.

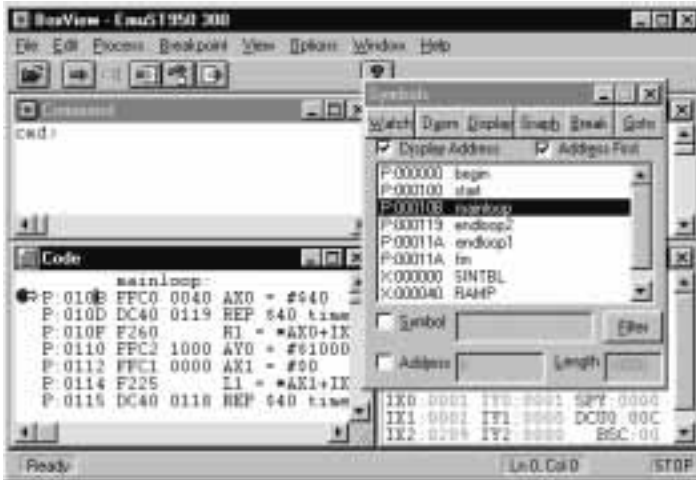


Fig.12 - Setting the Breakpoint

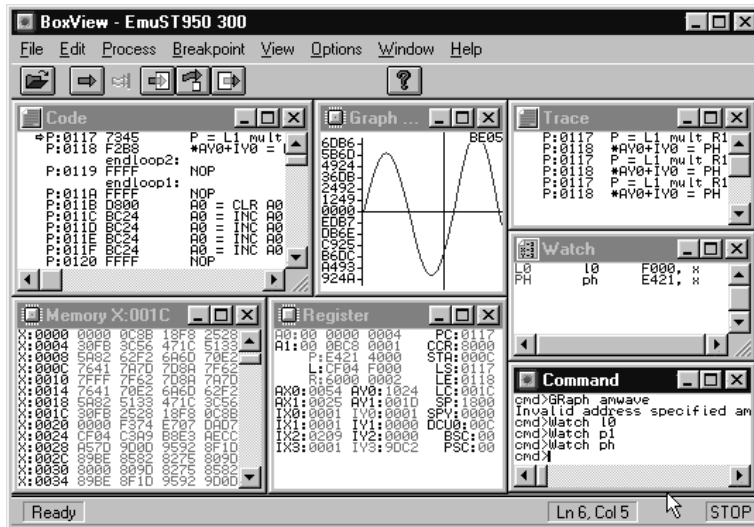
Variables can be added to the Watch window using the Watch command, or by pressing the Watch button within the Symbols window.

Once the variable is added to the Watch window, it can be expanded (arrays, data structures, pointers) by double-clicking on the line to be expanded or collapsed.

To remove the item from the Watch window, you can press Del key when cursor is on the line to be deleted.

Run code in the continuous step mode (Shift-F8 or ProcessContStep). Last selected Code window will follow program counter, opening correct source file for the display. Other Code windows will show PC (right yellow arrow) only if it is within lines displayed.

Following Figure will show few other windows opened. All windows will be described in the following chapter.



2

Fig.13 - Multiple windows opened

2

Below is the layout of the local menu for the command window:

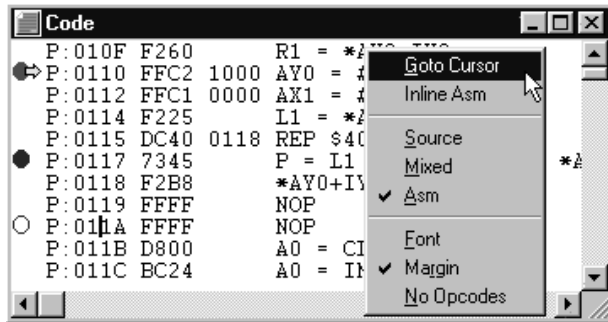


Fig.17-Code window's local menu

Local menu can be activated with the right-mouse button. Following functions can be accessed from this menu:

3

- execute code to the cursor (this is possible only for the code executed out of RAM). After reaching the desired location, temporary breakpoint is removed automatically. The same effect can be accomplished with the F7 function key.
- invoke inline assembler to modify the line of code
- set the mode of the Code window: Asm, Mix or Src. This can be also done with the commands from the command window (ASM, MIX, SRC). If the command is executed from the command window, it will affect only last selected code window (in case that multiple code windows are open).
- change font size for this window. The global font selection is available from main pull-down menu: Options-->Font.
- disable/enable display of the opcode values. If the option "No Opcodes" is checked, no opcodes will be displayed, so the code can be displayed in the smaller window.
- enable/disable selection margin. With the selection margin enabled, small icons are indicating attributes of the particular line within code window. If the margin is disabled, attributes are indicated by color of the background/foreground for the displayed code line.

Following attributes are displayed with icons and/or colors:

- current PC value: yellow arrow
- selected frame: green triangle
- SW breakpoint: dark red circle
- SW breakpoint with “show” option: light red circle
- disabled SW breakpoint: red empty circle

3.3 - Command Window



Fig.18 - Command window

Command window allows for entering text commands. It saves the history of executed commands. Old commands can be edited and/or reexecuted. Pressing Enter key when the command line is empty will repeat the last command.

3

3.4 - Memory Window

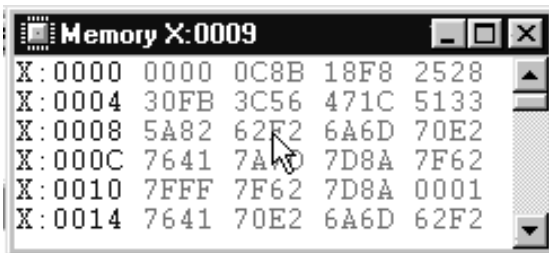


Fig.19 - Memory window

Memory window allows to display contents of the DSP memories. Starting address to display can be set from the Command window, using Display command, from the Symbol window by selecting the symbol and pressing the Display button.

The change of starting address can be modified with the command DISPLAY, or by direct editing of the address field. The memory contents can be also directly edited within the window. The highlighted portions of the memory window can be copied to the window's clipboard area, and pasted to any other application.

The local menu of the memory window (right-mouse button) allows for disabling of the address display, so more data can be displayed within the window. The current address, where the cursor is located, is displayed within the window tile. If address corresponds to valid symbol, symbol also will be displayed at the window title.

3.5 - Register Window

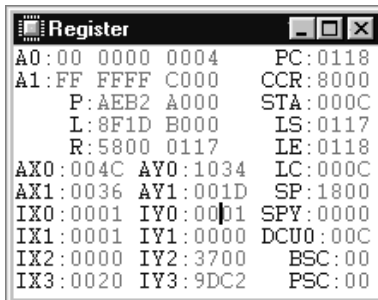


Fig.20 - Registers window

Register window displays contents of the DSPs core registers. Number and format of the registers displayed depends on the DSP type. Value of any core register can be modified with the CHANGE or SR commands, or by direct editing of the register value.

BoxView allows for multiple Register windows to be displayed.

3.6 - Graph Window

Graph window allows to display content of the memory in the graphical format. Single pixel on the horizontal axis represents one memory location. Vertical axis represents the memory value. Vertical axis can be scaled with up/down arrows. Scale can range from single increment per pixel, up to maximum memory value per graph height. There is also logarithmic scale available.

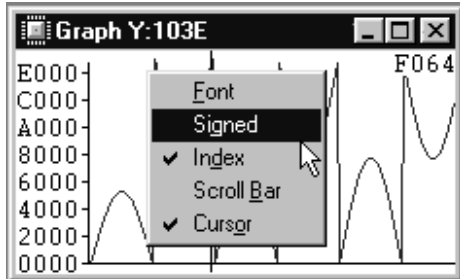


Fig.21 - Graph window

The local menu available with the right-mouse button, allows for following controls:

- Change font size - this affects index and value at the cursor display
- Sign toggle - allows for formatting the graph in signed or unsigned mode
- Index toggle - enables/disables display of the memory value scale on the left side of the window
- Scroll bar toggle - displays/hides the horizontal scroll-bar
- Cursor toggle - enables display of the cursor. Value at the cursor position is displayed in the upper right corner of the window. Position of the cursor can be set with mouse or right/left cursor keys.

3

3.7 - Symbols Window



Fig.22 - Symbols window

Symbols window displays global symbols available in the debugged application.

Format of the display can be changed with two options:

Display Address check box enables display of the numeric address value of the symbol.

Address First will control if the address of the symbol is displayed before or after the symbol name. This controls sorting of the list entries.

Here are five operations, which can be performed on the selected symbol:

- Watch** adds the symbol to the Watch window
 - Dasm** sets the starting address of the Code window
 - Display** sets the starting address of the Memory window
 - Graph** Set the starting address of the graph window
 - Break** inserts software breakpoint at the symbol address
 - Goto** inserts temporary breakpoint at symbols address and starts execution
- Displayed symbols can be also filtered with two optional filters:
- Symbol** will select all the names matching control string. * character is a wild character matching any number of any characters.
 - Address** will display symbols, which address is within the specified range. Starting address can be in two formats: with and without memory space prefix. No space prefix will allow all memory spaces to match address range.

3

3.8 - Trace Window



Fig.23 - Trace window

Trace window shows last instructions executed by the target DSP. The trace buffer holds last six addresses. The address is followed by the disassembly of the code at this address location.

3.9 - Watch window

Watch window allows to display selected symbolic items or expressions. Items can be added to the watch window with the **WATCH** command, or by pressing the **Watch** button within the **Symbols** window. To delete item from the watch window, you can press the **Del** key on the keyboard, when the cursor is over the line to be deleted, or with the **WATCH** command.

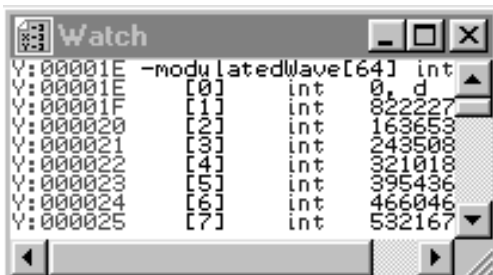


Fig.24 - Watch window with sample data

If the watch contains element which can be expanded (array, structure, union, etc.) the **+** will be displayed in front of the symbol name. Expanded symbols will have **-** character, indicating, that the item can be collapsed.

3

The watch line properties can be edited through the dialog available with the right-mouse button.

3.10 - Open connection dialog



Fig.25 - Open connection dialog

Open Connection dialog allows to select the target for the BoxView operation.. Primary item to be selected is the **Interface Type**.

Connection type can be Direct or Remote over the TCP/IP network (optional BoxServer application is required).

3

For the direct connection, depending on the interface type, proper port needs to be selected. It can be one of the COM ports, one of the printer ports, or base address for the ISA interface card.

For TCP/IP connection only required parameters are the address of the server, and port # of the IP server to be connected to. To inquiry the server about the target availability, use the Query button. After query is completed, list of the available target devices will be displayed in the Target Select list.

The **Initialize on start** check box, if checked, will cause auto target initialization, on next BoxView invocation.

3.11 - Startup Options

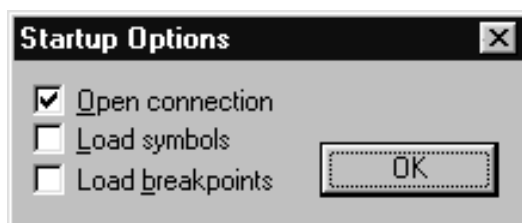


Fig. 26 - Startup Options

This dialog is available from the main pull down menu: Options --> Startup.

The options checked within this dialog, control the program action during its invocation:

- Open connection - will initialize the connection to the target, as it was last specified by the Open Connection dialog.
- Load symbols - will reload the last used symbol table. This option is disabled if the open connection is not active.
- Load breakpoints - will reload software breakpoint table. The breakpoint table is saved during program termination, just before clearing all software breakpoints on program exit.

3.12 - Load File Dialog

Load File dialog provides options necessary for uploading the code to the target DSP. Selection of the file is available with the File button. This will invoke standard Windows file browser dialog. Main check-boxes are: **Load Data** and **Load Symbols**. Disabling of data loading can be useful for debugging applications, which are loaded to the DSP through another interface.

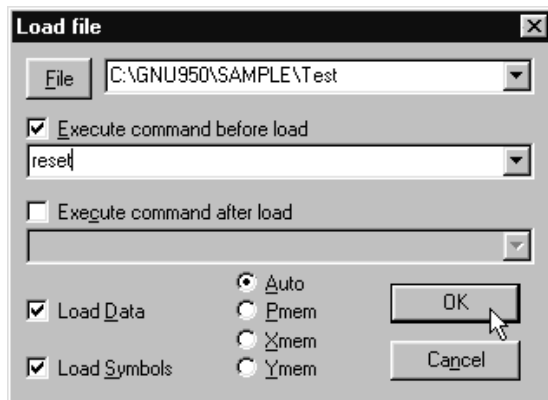


Fig.27 - Load File dialog

3

For the code loaded from the .s19 files, target memory option needs to be specified. The .s19 format does not contain the memory space information, so memory can be defined with P, X or Y radio-button selection. The Auto mode performs load with the following algorithm:

- fileName.s19 is loaded to P memory space
- fileName_1.s19 is loaded to X memory space
- fileName_2.s19 is loaded to Y memory space

Execute command before load check-box, enables editable field, where user can specify the commands required to enable access to the DSP memory, or set any initial conditions.

Execute command after load can be used to initialize certain variables, set breakpoints and start execution.

3.13 - Hardware Breakpoints

Hardware breakpoints are controlled through the dialog. Dialog is available through the main pull-down menu: Breakpoints --> Hardware break.

Dialog follows the features available on the emulation circuitry for the specific DSP processor.

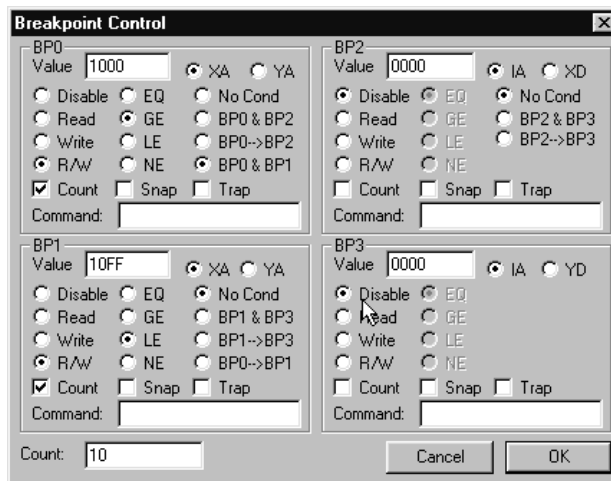


Fig.28 - Hardware breakpoint for ST18950

ST18950 provides four address/value comparators with additional control logic. Every breakpoint register BP0..BP3 allows for specification of the 16-bit value, which can be compared for: equal, greater or equal, less or equal and not equal conditions. Grouping registers BP0 and BP1 or BP2 and BP3 allows for specification of the range triggering. First pair allows for breaking on events related to the address bus for X or Y memory. Second pair controls program address execution and/or data bus contents for X and Y memory spaces.

Software breakpoints require one of the program address breakpoints (BP2 or BP3) for its operation. If any software breakpoints are active, one of the hardware breakpoints will be disabled for the user access, so it can be used for the software breakpoint control.

Any of the four breakpoints can have enabled counter option. If the counter option is enabled, the target DSP will be halted, when the counter reaches zero.

Breakpoints can have command associated with its occurrence. Every time the breakpoint condition occurs, the specified command or macro will be executed.

The trap option will cause the refresh of the debugger windows, execution of the specified command, and resume target code - run the DSP.

3.14 - Software Breakpoints

Software breakpoints can be set in the following ways:

- BREAK command
- By selecting the symbol and option Break within the symbols dialog
- By double-clicking on the code line, or with the function key F7.

The action of the mouse-click or function key can be modified by pressing the Shift or Ctrl keys:

- Ctrl key - toggles the “show” attribute. The breakpoint with the “show & go” attribute will cause the code execution to continue, after updating the DSP resource windows.
- Shift key - will toggle the disable attribute. Disabled breakpoint is removed from the memory, but it is still present in the symbol table for future use.

3

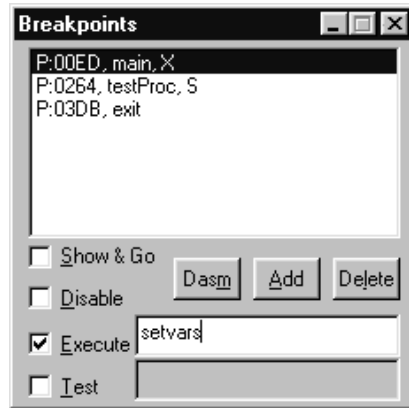


Fig. 29 - Software Breakpoints

List of the breakpoints and attributes associated with every breakpoint, can be controlled from the Breakpoint Dialog (main pull-down menu: Breakpoint-->Display).

The three buttons allow for following action:

- Dasm - will set the address of the code window, to display the code at the breakpoint address.
- Add - allows to specify new breakpoint address
- Delete - will clear the breakpoint and remove it from the list

The check boxes allow for direct control of every breakpoint attributes:

- Show & Go - causes redisplay of memory, graph and register windows, and continuation of the DSP execution
- Disable - removes breakpoint from memory
- Execute - will execute specified command or macro
- Test - will invoke expression evaluator. If the expression evaluates to non zero (TRUE), execution of the DSP code will be terminated, otherwise it will continue.

3.15 - User Buttons

BoxView allows for customization of the toolbar with user buttons. Addition of the button is provided with the dialog available at the main pull-down menu (Options->User Buttons).



Fig. 30 - User Buttons Dialog

The dialog allows for filling three fields:

Button Text: this text will be stretched to the button size, and displayed on top of it. The text can be between 1 and 4 characters long.

Quick Help: represents the text, which will be displayed, when mouse pointer is hold over the button.

Execute: represents the actual command or macro command name to be executed, when the button is pressed.



Fig. 31 - User Button with help

When the mouse pointer is hold over the button,the quick help text will be displayed. The contents of the help text is user configurable.

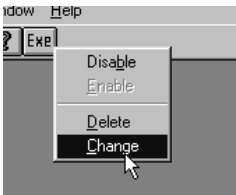


Fig. 34 - User Button local dialog

3

Every button can be changed or removed with the local menu available with the right-mouse button.

CHAPTER 4 - BoxView Commands

4.1 The command interpreter

The command interpreter is a recursive function which parses and executes commands entered at the keyboard, command files, macros and trap / breakpoint commands. It is an infinite loop that executes the following steps:

1. Reads a line from the input.
2. Breaks a read string into words using a sequence of one or more of blanks, tabs, or commas as separators. Characters enclosed between double quotes are treated as a single word (this is only used in the PRINT command).
3. Replaces each occurrence of the symbol "@n" with the actual value of that argument [DO] if the input source is a macro.
4. Scans each word and string that is enclosed in reverse quotes "" and treats it as an expression.
5. Evaluates the expression and substitutes the result (printed as one to four hexadecimal digit characters) in the expression string. Each time a new input source is set up, the command interpreter is recursively called to read the new source up to the end. If the specified command does not match any of the debugger commands the new command file is opened automatically. The debugger starts the input from a command file, the file is read until the end of the file is reached.

The DO command starts the input from a macro body, the macro is read up to the end of body.

Macros, command files, and trap commands can be nested up to 20 levels.

The IF/ELSE/ENDIF commands select a group of commands to execute without changing the input source and without affecting the level count. The command interpreter ends when the ELSE or the ENDIF command is found.

Ending the top level (standard input reading) command interpreter, terminates the emulator program (after a confirmation request).

During their execution, some commands (ASM, SM, etc...) require more input to be read. This input is taken from the same source from which the command was read.

Lines beginning with a colon character are ignored and can be used for programmer's comments.

4

The following key sequences are defined in keyboard input mode:

up and down arrows	retrieves previous commands
ESC	clears the input line
Ins	switches between insert and overwrite
Del	erases the current character

Commands and arguments are not case sensitive except in some particular cases which will be noted where applicable.

4.2 Command line expressions

Command line expressions contain one or more terms. Each term can be preceded by an unary operator, and separated by binary operators. Expressions must be single words and, therefore, cannot contain spaces. The syntax and semantics of expressions are very close to those of C language.

When an expression must be interpreted as a number, either to be used as a memory address, a register, or a value, the expression is sent to the expression evaluator. The expression evaluator returns the numeric value of the expression. Intermediate results are stored as 32 bit unsigned integers. The final result can be truncated to the lowest 8 or 16 bits as required.

4.2.1 Expression terms

In the following definitions:

- lowercase words are generic names
- uppercase words are keywords which can be in uppercase or lowercase inside the expression

Definition of terms used in expressions:

Term	Definition
number	A sequence of one or more hexadecimal, decimal or octal digits, depending on the current base selected [BASE]. It can be prefixed with '0x' to force it to be interpreted as a hexadecimal number.
'character'	A character enclosed between single quotes is evaluated as its ASCII value (so that 'A' is equivalent to the hexadecimal value 41).
\$symbol	A legal symbol name, preceded by the dollar sign '\$'. The value is the symbol value.
(expr)	An expression enclosed between parenthesis is a single term and the value is the value of the expression. It can be used to change the order of the evaluation of the expression terms.
Mterm MMterm	The value is the content of the memory (first case) at address <i>term</i> , or the content of the concatenation (second case) of the 2 memory data at <i>term</i> .
#term	The value read from channel <i>term</i> [OPEN]
R\$symbol	The value is the content of the register

4.2.2 Unary operators

The unary operators group right to left are defined as follows:

Definition of unary operators:

Operator	Description
!	! <i>term</i> is evaluated as 1 if <i>term</i> has value 0, 0 otherwise.
~	~ <i>term</i> is the bitwise negation of <i>term</i> , i.e ~0x55 is equal to FFFFFFFA.

4.2.3 Binary operators

All binary operators group left to right except the assignment operator which groups right to left. In decreasing order of precedence, the binary operators are:

Definition of binary operators

Operator	Description
* / %	Multiplication, division and modulo (remainder).
+ -	Addition and subtraction.
>> <<	Logical right and left shift, i.e. <code>0xC>>2</code> is equal to 3.
== !=	Logical equality and inequality, evaluated as 1 if true and 0 if false, i.e. <code>5==3+2</code> is equal to 1 and <code>5!=3+2</code> is equal to 0.
> >= < <=	Logical comparison for greater, greater or equal, less, less or equal, evaluated to 1 if true and 0 if false.
&	Bitwise AND.
^	Bitwise EXCLUSIVE OR.
	Bitwise OR.
=	Assignment operator. The left operand can be only an M, MM or #term. The right operand can be an M, MM, #term or the result of an expression evaluation.

4

4.3 Programming the emulator with the command line interface

4.3.1 Defining macro commands

Macro commands can be defined to carry out a sequence of individual emulator commands. The watchdata or action in breakpoint commands can be used to run a program starting from address 100, and see the value of the memory at address 50 when the program reaches the address 110. Alternatively a macro command can be written.

1. To define a macro, use the `de -m` command (ref. "DEFINE"):

```
de -m run_and_see
sr PC 100
sb 110
go
: display memory at address 50
dm 50
endm
```

The first line defines the macro name "run_and_see"; the last line indicates ends the macro command. Line 5 begins with a colon and is not read.

2. To run a previously defined macro, use the `do` command (ref. "DO"):

```
do run_and_see
```
3. To define the same macro with the possibility of defining the address at each execution, specify parameter "@ 1" in the macro definition:

```
de -m -f run_and_see
sr PC 100
sb 110
go
dm @1
endm
```
4. To execute this macro, type:

```
do run_and_see 40
```

The macro will be executed as before, replacing parameter @1 by its specified value: 40.
5. To list the macro commands, use command `li` (ref. "LISTSYMBOL"):

```
li -m run_and_see
```
6. To store your macros in a file on disk, use the command `ar` (ref. "ARCHIVE"):

```
ar -p -m file.mac
```

This command stores the macro in the file named "file.mac". This is an ASCII file which can be modified with any editor.
7. To retrieve the macro from file, type:

```
ar -g -m file.mac
```

4.3.2 Programming

Variables and programming statements can be included in a macro-commands. For full details on programming inside macros, (ref. FOR, IF, JUMP).

Up to 30 variables can be user defined from "v0" to "v1d" hexadecimal address. These variables can be set, tested, or used in programming statements.

Expressions must not contain spaces. For example, the line:

```
v0 = 0
```

is incorrect.

Using the same macro as above:

1. To run the program 10 times and see the following byte each time, use the define command (ref."DEFINE"):
de -m -f run_and_see
sr PC 100
sb 110
for v0 1 A 1
go
dm @1+v0
endfor
endm

4

When typing:

```
do run_and_see 40
```

the commands between FOR and ENDFOR will be executed 10 times (i.e. A times in hexadecimal, note that parameters are in current base). Command "dm @1+v0" will be interpreted as "dm 40+1", then "dm 40+2", and so on.

2. To execute out a test program:


```
de -m -f run_and see
reset
sb 110
v0=1
.loop
go
dm @1+v0
v0=v0+1
if (v0<=A) jump loop
endm
```
3. To define a label use the de command (ref. "DEFINE").
4. To jump to a label use the jump command (ref. "JUMP").

4.3.3 Memory and register access

To display the memory location in hexadecimal format and the memory value in decimal format, define a macro as follows:

```
de -m -f run_and_see
sr PC 100
sb 110
for v0 1 A 1
go
print -n -f "%x = " @1+v0
print -f "%d" m(@1+v0)
endfor
endm
```

In this macro, the syntax "m(@1+v0)" requests the memory value at address (@1+v0).

The PRINT command gives a formatted display.

1. To obtain the data in P memory at address 10, type or use in a macro:


```
v0=m10
```
2. To obtain the data in X memory at address 10, type or use in a macro:


```
v0=mx:10
```

3. To obtain the double data in memory at address 10 and 11, type or use in a macro:
- ```
if (m10<=m11)
```
- The test will be true if data in P memory at address 10 is lower or equal to data in memory at address 11.
- `v0=m(mm10)` v0 will contain the data in memory which address is contained in double data at address 10 and 11: this expression is an indirect access to the memory.
- `v0=mm(m10)` v0 contains the word in memory whose address is contained in data in memory at address 10.
- `v0=mmm10` will be interpreted as: `v0=mm(m10)`.

The above examples use absolute addresses, but symbolic names can be used instead. The character '\$' means a symbol name.

```
print m$
if (v0 == m$) ...
```

Symbols can be used for register addresses:

```
v0=r$a1h
v0 will contain the value of the register A1H.
```

You can write expressions such as:

```
if (r$pc==mm100) jump end
```

Memory and register access syntax is explained in full in the PRINT command description.

## 4

### 4.4 File management

Channels are used to store the results of programs. In the D950 emulator, the channel feature is similar to file management in C or other high-level language: channels can be opened, read from, written to and closed.

Channels may be opened as input or output, but not both; they can contain ASCII or binary values. All values must have the same length, for example, to read a different memory address at each step of the previous debugging program, and store the content of X-space memory in a file. Channel syntax is explained in full in the OPEN and CLOSE command description. (Ref. "OPEN" and "CLOSE").

See the following example:

1. Edit a file called "address", containing the addresses to check.  
 Since addresses are 16-bits long, use a file that contains 16-bits long values.  
 In ASCII mode, each value must be separated by spaces, tab or end of line.  
 Suppose the "address" file is:  
 0000 0100 0105 0107 0168 00a4 0078 0123 0150 0120
2. To open the file "address" for reading, giving it the channel number 1,  
 reading each time 2 bytes, type:  
 open 1 address input ascii 2  
 Output values are stored in a binary file "result", in which each value will be a  
 data.
3. To open file "result" for writing, giving it the channel number 2, writing each  
 time 2 bytes, type:  
 open 2 result output binary 2
4. To read and write from/to a channel, we use the character "#" followed by the  
 channel number. For example, "#1" means one value, read from or written to  
 the file associated ,to channel 1.  
 The macro becomes:  
 de -m test\_auto  
 open 1 address input ascii 2  
 open 2 result output binary 2  
 for v0 1 A 1  
 go  
 v1=#1  
 #2=mx:v1  
 endfor  
 close 1  
 close 2  
 endm  
 Expression "v1=#1" writes to v1 the 16-bit value read in file "address"  
 (channel #1). Expression "#2=mx:v1" writes in the "result" file (channel #2)  
 the contents of the memory at location "v1" in the memory space X.  
 By running this macro and checking the contents of the file "result" versus a  
 reference version each time the program is changed, you have a  
 non-regression procedure to test program evolution.
5. To test for the end of file, open the file as input and use the @ command.

### 4.5 General notes

The following notes describe the syntax and general concepts used in the description of commands.

1. Debugger's commands can be typed in both lower and upper case letters, arguments of commands must be separated by at least one space, tab or comma.
2. The names of commands can be written in full or in abbreviated form: in the commands descriptions, the maximum (i.e. the shortest) allowed abbreviation is shown in upper case and the remaining part of the name is shown in lower case.
3. The following syntax has been used:  
The character "|" means "or".  
An argument between square brackets is optional.  
An argument followed by dots as in "value ..." means that the argument can be specified more than once.  
Braces "{" and "}" are used to group arguments that are treated as a single construct.
4. Numeric values are interpreted and printed in the selected numeric base (DECIMAL, OCTAL, or HEXADECIMAL). The current base selected is indicated by the prompt letter at the beginning of each command line "d:" "o:" or "x:" (ref."BASE").  
Some commands require and/or print value in decimal base only; this will be explicitly indicated in the command description.
5. Iteration or long commands can be aborted by typing <ESC> key
6. The last command executed may be requested again by typing a dot character
7. All previous commands can be recovered with the up and down arrows, and modified using left and right arrows, <DEL>, <INS>, <Home>, <End>...
8. Error messages produced by the debugger are self-explanatory and indicate the cause of the error.

| <b>ADDR</b>        |                | Sets disassembly/source address                                         |
|--------------------|----------------|-------------------------------------------------------------------------|
| <b>Syntax</b>      |                | <u>A</u> DDR <i>address</i>                                             |
| <b>Menu</b>        |                | View → Symbols                                                          |
| <b>Description</b> |                | The ADDR command sets starting address for the last active CODE window. |
|                    | <i>address</i> | starting address value of the program to be displayed.                  |

| <b>ALIAS</b>       |  | Set/display command alias                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|--|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      |  | <u>A</u> LIAS [name["string"]]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Menu</b>        |  | ---                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Description</b> |  | The ALIAS command entered without any parameters will display all defined aliases. If only the name is specified, system will display current definition of the alias associated with this name.name alias name for the reference as the system commandsstring command string referenced by the name. Multiple commands in the alias definition need to be separated with the semicolon. Command parameters can be specified with the percent sign and a number (%1, %2, ...). Previously defined alias names can be used in the new command string. |

| <b>ARCHIVE</b>     |    | Archive symbols and macros in files                                                                                                                                                                                                   |
|--------------------|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      |    | <u>A</u> RCHIVE {-g -p} {-a -m} [-f] <i>file</i>                                                                                                                                                                                      |
| <b>Menu</b>        |    | ---                                                                                                                                                                                                                                   |
| <b>Description</b> |    | The ARCHIVE command loads into memory from a file or saves in a file the symbol table of the specified type:                                                                                                                          |
|                    | -a | memory address type symbols                                                                                                                                                                                                           |
|                    | -m | macros                                                                                                                                                                                                                                |
|                    | -g | (shorthand for "get"). Symbols and definitions found in <i>file</i> are loaded into the symbol table. Unless the -f option is selected, symbols already defined in the symbol table are not redefined and an error message is issued. |

4

-p (shorthand for "put"). Symbols of the given type and their definitions are stored in the specified *file*. If the file already exists and option -f is not selected, a confirmation is requested before overwriting the file.

Symbols of memory addresses are saved in files in alphabetical order, one per line with their value in hexadecimal format.

Macros are saved in files as they are defined (that is literally)

See also DEFINE, DO, DISASM, LISTSYMBOL, UNDEFINE

|                    |                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ASM</b>         | <b>Change code window display to the assembly mode, on-line assembler</b>                                                                                                                                                                                                                                                                                                      |
| <b>Syntax</b>      | <u>ASM</u> [ <i>address mnemonic</i> ]                                                                                                                                                                                                                                                                                                                                         |
| <b>Menu</b>        | Local menu of the code window                                                                                                                                                                                                                                                                                                                                                  |
| <b>Description</b> | <p>The ASM command without parameter will change display mode of the last selected Code window to assembly mode.</p> <p>The ASM command with address and mnemonic string, will invoke the inline assembler in the context of the specified address.</p> <p>This command allows to write D950 instructions in C-like syntax in the program memory at the specified address.</p> |
| <b>Example</b>     | <pre>cmd&gt;asm 100 jump begin</pre> <p>See also DISASM</p>                                                                                                                                                                                                                                                                                                                    |

4

|                    |                                                                                                                                                                                                                                                                                                               |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>BASE</b>        | <b>Base change base of numbers</b>                                                                                                                                                                                                                                                                            |
| <b>Syntax</b>      | <u>BASE</u> [x o d]                                                                                                                                                                                                                                                                                           |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                           |
| <b>Description</b> | <p>The BASE command may be used to select the base of numbers.</p> <p>"x" selected hexadecimal numbers</p> <p>"o" selected octal numbers</p> <p>"d" decimal numbers</p> <p>The base currently selected is indicated by the prompt symbol of the debugger, according to the parameter of the BASE command.</p> |
| <b>Example</b>     | <pre>Base d Base x</pre>                                                                                                                                                                                                                                                                                      |

**BEEP** Generate alert sound

**Syntax** `BEEP`  
**Menu** ---  
**Description** This command activates system's default sound (configured through "Sounds" in the Windows Panel)

**BREAK** Set/Clear/Display SW breakpoints

**Syntax** `BREAK [OFF] [address]`  
**Menu** Breakpoint → Display  
**Description** The BREAK command sets the software breakpoint at the specific address. If no parameters are specified, system will display all breakpoints set. OFF will remove breakpoint at the specific address, or all breakpoints if the address is omitted.  
  
*address* absolute address or symbol

**BYE** Exit application

**Syntax** `BYE`  
**Menu** ---  
**Description** Exit application

**CALLS** Display call stack

**Syntax** CALLS  
**Menu** ---  
**Description** The CALLS command displays current call stack in the command window. This represents the information displayed also in the Calls window, but it will allow to add this information into the log file. Call stack is build using the information created by C programs. This is not available for the assembly programs.  
 See also LOG.

**CASCADE** Cascade all windows

**Syntax** CASCADE  
**Menu** Window → Cascade  
**Description** Arrange all opened windows in the cascade format.

**CD** Change current directory

**Syntax** CD  
**Menu** ---  
**Description** CD command allows to change or display debugger's current working directory  
*pathname* directory path to be changed or displayed

| CHANGE             | Change memory/register contents                                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code>CHANGE</code> address   address range   register name value                                                                                                                                                                                 |
| <b>Menu</b>        | Direct resource editing                                                                                                                                                                                                                           |
| <b>Description</b> | The CHANGE command allows to change contents of the memory range or the register.                                                                                                                                                                 |
| address memory     | address to be changed in the format: space:offset, where space is one of the allowed memory spaces for the processor (P, X, Y), offset absolute value of the address offset. Address can be also specified in the form of the application symbol. |
| address range      | specifies memory area to be changed to the value. Range can be accepted in two formats:<br>start_address..end_address or start_address#count.                                                                                                     |
| register name      | refers to one of the valid DSP registers (A0L, SP, PC, etc.)                                                                                                                                                                                      |

| CLOSE              | Close I/O channel                                                                                                                                                                                                                                         |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code>CLOSE</code> channel                                                                                                                                                                                                                                |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                       |
| <b>Description</b> | This command allows to close the channel specified.<br><i>channel</i> must be a decimal number from 1 to 10.<br>Closing a channel that has not yet been opened does not cause an error. Once closed a channel cannot be used for reading or writing data. |
| <b>Example</b>     | <code>close 1</code><br>See also OPEN                                                                                                                                                                                                                     |

**CLS** Clear screen

**Syntax** `CLS`  
**Menu** ---  
**Description** The CLS command clears the screen, and put cursor at the first line, first column  
**Example** `CLS`  
 See also @CURX, @CURY, LOCATE

**CM** Compare memory values

**Syntax** `CM addr1 addr2 [count]`  
**Menu** ---  
**Description** The CM command allows to compare the content of *count* data of memory at address **addr1**, with the content of memory at address **addr2**.  
 If **count** is not specified, it is assumed equal to 1.  
 For each comparison resulting in a difference, the addresses and the contents of the corresponding memory cells are printed.  
**Example** `CM 10 100 20`  
`CM x:10 x:100 20`  
 See also DM, FM, SM

**4**

**CODEMENU** Add/remove entry to Code window menu

**Syntax** `CODEMENU text [command|r]`  
**Menu** ---  
**Description** CODEMENU command allows to add or remove custom commands to/from the CODE window's local menu (available with right-mouse click).

**COLOR****Change window color**

**Syntax** `COLOR window_id value`

**Menu** ---

**Description** The COLOR command allows to change default colors in the specific window. Parameter winName represents the window name, as it appears in its title. ColorId is the number for the color type to change. There are different numbers of colors for different windows. This information is displayed in the colors setup dialog (Options-->Colors). The value is a hexadecimal value representing the background (high nibble) and foreground (low nibble). The color association with color values 0..15 can be seen in the Color Setup dialog. The default for value 0xf (15) is white, default for value of 0 is black, but any color can be changed with above dialog.

*Example* `Color memory 1 f2`

**COMPARE****Compare memory values**

**Syntax** `COMPARE addr1 addr2 [count]`

**Menu** ---

**Description** The CM command allows to compare the content of count data of memory at address addr1, with the content of memory at address addr2.  
If count is not specified, it is assumed equal to 1.  
For each comparison resulting in a difference, the addresses and the contents of the corresponding memory cells are printed.

*Example:* `CM 10 100 20`  
`CM x:10 x:100 20`  
See also: DM, FM, SM

| <b>CONNECT</b>     | <b>Initialize hardware interface</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <b>S</b> |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| <b>Syntax</b>      | <code>CONNECT [interface type[,device[, [IpAddr][.port]]]]</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |          |
| <b>Menu</b>        | Options → Open Connection                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |          |
| <b>Description</b> | <p>The CONNECT command initializes connection to the target processor. Parameters specify details of the connection. interface type can be one of the following: EmuSt950ISA, EmuSt950PP or EmuSt950Test. Device indicates device # in the scan chain to connect to. This is required for the multi processor systems.</p> <p>IpAddr defines TCP/IP address for the remote connection to the server. For the local connection this parameter needs to be empty. port indicates physical address of the target to be connected to. In case of the remote connection to the IpAddr this is server's port number (by default 5417 is used).</p> <p>For the local printer port connection this is the LPT port number (1, 2, ...). For the local I/O port interface, this is the I/O port base address (240, 340, ...).</p> |          |

| <b>COPY</b>        | <b>Copy memory block</b>                                                                                                                                 |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code>COPY address_range mem_address</code>                                                                                                              |
| <b>Menu</b>        | ---                                                                                                                                                      |
| <b>Description</b> | The COPY command allows the user to execute a fast copy of the memory block specified by the address_range to another location specified by mem_address. |

**4**

| <b>@CURX, @CURY</b> | Get cursor position                                                                                                                                                 |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>       | <u>@CURX</u><br><u>@CURY</u>                                                                                                                                        |
| <b>Menu</b>         | ---                                                                                                                                                                 |
| <b>Description</b>  | The @CURX and @CURY commands are variables that return the current cursor position. CURX returns the current column value, and CURY returns the current line value. |
| <b>Example</b>      | <pre>CM 10 100 20 CM x:10 x:100 20V0=@CURX print @cury</pre> <p>See also CLS, LOCATE</p>                                                                            |

| <b>DASM</b>        | Set disassembly address or scroll Page Down Code window                                                                                                                                                                                                            |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>DASM</u> [address]                                                                                                                                                                                                                                              |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                |
| <b>Description</b> | The DASM command will set starting address of the Code window to the address value. If no parameter is specified, command will cause to scroll Code window one page down. Command affects only the last selected Code window, other Code windows are not affected. |

| <b>DEFINE</b>      | Define symbols and macros                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <pre><u>DEFINE</u> -a [-f] <i>symbol</i> [space:] <i>value</i> <u>DEFINE</u> -m [-f] <i>symbol</i> ... ENDM</pre>                                                                                                                                                                                                                                                                                |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Description</b> | In the first case, <i>symbol</i> refers to a memory address. <i>Symbol</i> and <i>value</i> must be a valid address. The second case is used to define the macro called <i>symbol</i> . A macro is a collection of commands that can be put into execution via the DO command. The lines following the DEFINE command are read up to a line containing the keyword ENDM and form the macro body. |

Notice that macro body is stored in memory literally, without any interpretation, except that leading blanks are stripped, and lines beginning with the colon character (:) are discarded.

Macro parameters can be embedded in macro body as "@n", where *n* is a single digit between 0 and 7. They will be expanded at macro execution time with the literal value of the actual parameters passed to the macro.

- *symbol* can be any word composed of letters, numbers, "\_" (underscore), "\$" characters. However, the first character can be a "." (period) but cannot be a digit.
- Symbols are case sensitive.
- Symbols can be of any length, but only its first 8 characters are considered meaningful.
- Unless the -f (shorthand for "force") option is specified, if the symbol is already defined, an error message is issued and the definition of the symbol is not changed.
- If two address symbols have the same value, the most recently defined one will be used when the symbol table will be searched for a value.
- Symbols are stored in memory for fast retrieval and alphabetical sorting. Each symbol requires memory that is allocated when the symbol is defined and released when the symbol is removed. Defining more symbols than allowed by the available memory will cause an "OUT OF MEMORY" error message.
- Address symbols may be used in any expression by writing them preceded by the dollar operator ("\$"). This operator will cause the symbol to be replaced by its numeric value.

Example

4

```
define -a lab x:10
define -m LOADANDGO
reset
print "LOADING FILE @1"
load @1
sb on @2
go
endm
```

A macro, once defined, may be invoked using the DO command. About the example illustrated above, the LOADANDGO macro could be executed like this: do LOADANDGO prog\_8 1234

Then a DSP reset will be done, prog\_8 loaded, run and finally stopped if address 1234 is reached.

See also ARCHIVE, DISASM, DO, LISTSYMBOL, UNDEFINE.

**DI** Disassemble memory

**Syntax** `DISASM addr [count] or DI addr [count]`

**Menu** ---

**Description** The DISASM command allows to display the content of memory at address `addr`, in symbolic format, i.e., in mnemonic assembler code.  
See also ASM

**DIR** Display directory contents

**Syntax** `DIR [directory name]`

**Menu** ---

**Description** The DIR command displays a directory listing in the display area of the COMMAND window. If you use the optional [directory name] parameter, the debugger displays a list of the specified directory's contents. If you do not use the parameter, the debugger lists the contents of the current directory.

**DISASM** Display memory in mnemonic format

**Syntax** `DISASM addr [count]`

**Menu** ---

**Description** The DISASM command allows to display the content of memory at address `addr`, in symbolic format, i.e., in mnemonic assembler code.  
See also ASM

**DISCONNECT** Close connection to the target processor

**Syntax** `DISCONNECT`

**Menu** Options → Close connection

**Description** Close hardware Interface.  
The DISCONNECT command will disable connection with the target processor. This allows to select another processor to be accessed (in case of the multiprocessor or remote access).

**DISPLAY** Set memory address or Scroll Page Down Memory window

**Syntax** `DISPLAY [address]`

**Menu** View → Symbols dialog

**Description** The DISPLAY command will set starting address of the Memory window to the address value. Address should have the following format: space:offset, where space is one of the allowed memory spaces for the processor (P, X, Y), offset absolute value of the address offset. Address can be also specified in the form of the application symbol. If no parameter is specified, command will cause to scroll Memory window one page down Command affects only the last selected Memory window, other Memory windows are not affected.

**DM** Display memory

**Syntax** `DM addr`  
`DM from to`

**Menu** ---

**Description** The DM command allows to display the memory content on a word basis.  
In the first case, the command displays the content of memory at address *addr*.  
In the second case, the command displays the content of the memory block enclosed between addresses *from* and *to* (included). Memory content is displayed both in numeric notation and in ASCII code.

**Example** `DM 0 50`  
`DM x:10 20`  
See also CM, FM, SM

4

| <b>DO</b>          | <b>Execute macro command</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code><u>DO</u> <i>macro_name</i> [actual_parameters_list]</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Description</b> | <p>The DO command executes the macro <i>macro_name</i> with the specified parameters. The macro body is read by the command interpreter and commands executed as if read from the keyboard.</p> <p>Interactive commands, like ASM, take their input from the lines following the command in the macro body. They also work silently, i.e., they do not display any prompt.</p> <p>Before a line is read from macro body, any instance of "@n", where <i>n</i> is a single digit between 0 and 7, is substituted with the corresponding parameter taken from the parameter list.</p> <p>"@0" is the macro name itself, "@1" is the first parameter, and so on. Up to 7 parameters can be given in the parameter list. Parameters not defined are substituted with a null string.</p> <p>Macro executions can be nested. Execution can be interrupted with &lt;Esc&gt; key.</p> |
| <b>Example</b>     | <pre>define -m LOADANDGO reset print "LOADING FILE @1" load @1 sb on @2 go endm do LOADANDGO prog_8 1234</pre> <p>See also DEFINE, LISTSYMBOL, QUIT, UNDEFINE</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

|           |                              |
|-----------|------------------------------|
| <b>DR</b> | <b>Display/set registers</b> |
|-----------|------------------------------|

**Syntax**                    DR  
                               DR *reg-name*  
                               DR *reg-1 reg-2*

**Menu**                        ---

**Description**              The DR command displays the content of registers.

- Used without argument, it displays all DSP registers
- If a single register is specified, only that register will be displayed.
- In the last form, the registers between *reg-1* and *reg-2* (included) are displayed, one per line. Register order is:
- AX0, AX1 X address registers
- AY0, AY1 Y address registers
- IX0, IX1, IX2, IX3X index registers
- IY0, IY1, IY2, IY3Y index registers
- SP, SPY Stack pointers
- LS, LC, LE Loop start/count/end registers
- L0, L1, R0, R1 DCU input registers
- PL, PH Product registers LSB/MSB
- CCR Condition code register
- DCU0CR, STA Status registers
- A0L, A0H, A0E Accumulator 0 LSB/MSB/Ext
- BSC Barrel shifter control register
- A1L, A1H, A1E Accumulator 1 LSB/MSB/Ext
- PSC Product shift control register
- PC Program counter

4

**Example**                    DR  
                                   DR A0H  
                                   DR IX0 IX3  
                                   See also SR

| <b>ENDM</b>        | <b>End of macro definition</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <pre> DEFINE -a [-f] symbol address DEFINE -m [-f] symbol ... ... ENDM </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Description</b> | <p>In the first case, defined symbol will allow references to the given address. The second case is used to define the macro called symbol. A macro is a collection of commands that can be put into execution via the DO command. The lines following the DEFINE command are read up to a line containing the keyword ENDM and form the macro body.</p> <p>Notice that macro body is stored in memory literally, without any interpretation, except that leading blanks are stripped, and lines beginning with the colon character (:) are discarded. Macro parameters can be embedded in macro body as "@&lt;n&gt;", where &lt;n&gt; is a single digit between 0 and 7. They will be expanded at macro execution time with the literal value of the actual parameters passed to the macro.</p> <p>Unless the -f (shorthand for "force") option is specified, if the symbol is already defined, an error message is issued and the definition of the symbol is not changed. If two address symbols have the same value, the most recently defined one will be used when the symbol table will be searched for a value.</p> <p>Symbols are stored in memory for fast retrieval and alphabetical sorting. Each symbol requires memory that is allocated when the symbol is defined and released when the symbol is removed. Defining more symbols than allowed by the available memory will cause an "OUT OF MEMORY" error message.</p> <p>Address symbols may be used in any expression by writing them preceded by the dollar operator ("\$"). This operator will cause the symbol to be replaced by its numeric value.</p> |
| <b>Example</b>     | <pre> define -a lab x:10 define -m LOADANDGO reset print "LOADING FILE @1"load @1 sb on @2 go endm </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

A macro, once defined, may be invoked using the DO command. About the example illustrated above, the LOADANDGO macro could be executed like this: do LOADANDGO prog\_8 1234

Then a DSP reset will be done, prog\_8 loaded, run and finally stopped if address 1234 is reached.

| EVAL | Evaluate expression |
|------|---------------------|
|------|---------------------|

|                    |                                                                                                                                                                                                                                                                             |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code><u>E</u>VAL rval=lval addr2 [count]</code>                                                                                                                                                                                                                            |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                         |
| <b>Description</b> | This command allows to evaluate any "C type" expression. The evaluation result is stored into the "lval" destination. Destination can be any register, memory location or system variable (v0..v1d). To display the result of the evaluation, "?" command needs to be used. |
| <i>Example</i>     | <code>EVAL IX0 = *ptr + 4</code>                                                                                                                                                                                                                                            |

| ? | Evaluate expression and print results |
|---|---------------------------------------|
|---|---------------------------------------|

|                    |                                                                                                                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code>? [lval= ] rval</code>                                                                                                                                                                                              |
| <b>Menu</b>        | ---                                                                                                                                                                                                                       |
| <b>Description</b> | This command displays the result from the expression evaluator. This is identical to the EVAL command, but it also displays the result in the command window. The expressions supported are any C-type expression string. |
| <i>Example</i>     | <code>? AX0+AX1</code>                                                                                                                                                                                                    |

**EXEIO**      Initializes data transfer

**Syntax**                    EXEIO

**Menu**                        ---

**Description**                Command initializes data transfer to/from the DSP. Parameters of the data transfer are passed within global variables v20..v23. Result of the data transfer is returned in the variable v24. If transfer fails, variable is set to non-zero value.

                                Global variables definition:

                                v20            transfer starting address within the DSP memory

                                v21            number of words to transfer

                                v22            direction (0x8000 - write, 0x4000 - read) and memory space (0 - PMEM, 1 - XMEM, 2 - YMEM)

                                v23            file # to use

                                v24            error flag set by transfer routine

                                See also: INPUT, OUTPUT

**@EOF**            Test end of input channel

**Syntax**                    IF @EOF (channel#) JUMP *label*

**Menu**                        ---

**Description**                The @EOF command tests if the end of a file opened when the input channel has been reached.

                                The function returns FALSE if the end of a file has not been reached, otherwise it returns TRUE. The function works like the C-language function, meaning that it returns TRUE when the program has tried to read after the end of the file; in this case, the value read is undefined.

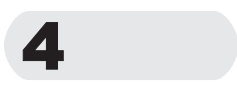
**Example**                    de -m test

```

open 1 @1 input ascii 2
.loop
v1=#1
if @EOF(1) jump endfile
print v1
jump loop
.endfile
print "end of file encountered"
close 1
endm
do test datas.2

```

                                See also OPEN, CLOSE



| FILL               | Fill memory with a pattern                                                                                                                                                                                                                                  |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>FILL</u> from to pattern                                                                                                                                                                                                                                 |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                         |
| <b>Description</b> | <p>Fill the memory block indicated by the limit address &lt;from&gt; and &lt;to&gt; with the values indicated in pattern</p> <p>Up to 8 values can be indicated. They will be repeatedly put in the memory in the same order until the block is filled.</p> |
| <b>Example</b>     | <p>FM x:100 200 1234 5678</p> <p>See also CM, DM, SM</p>                                                                                                                                                                                                    |

| FM                 | Fill memory with a pattern                                                                                                                                                                                                                                                         |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>FM</u> [space:] from to pattern                                                                                                                                                                                                                                                 |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                |
| <b>Description</b> | <p>The FM command fills the memory block indicated by the limit addresses <i>from</i> and <i>to</i> with the values indicated in <i>pattern</i></p> <p>Up to 8 values can be indicated. They will be repeatedly put in the memory in the same order until the block is filled.</p> |
| <b>Example</b>     | <p>FM x:100 200 1234 5678</p> <p>See also CM, DM, SM</p>                                                                                                                                                                                                                           |

4

| FOCUS              | Bring window to front                                                                                                                                                                                                                                                                                                                 |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>FOCUS</u> winName forward reverse                                                                                                                                                                                                                                                                                                  |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                   |
| <b>Description</b> | <p>FOCUS command allows to change active window within debugger interface. If the window name is specified, it will activate first window of the specified type. For forward/reverse parameters windows will be activated in the round robin order. Command can be assigned to the function key or button with the ALIAS command.</p> |

**FOR** Loop command execution

**Syntax**                    FOR *var start end* [step]  
 ...  
 ...  
 ENDFOR

**Menu**                        ---

**Description**                The FOR command loops from start value to end value in increments of step. The parameter *var* must be a valid variable (v0 to v1d).  
 Default value for increment is 1.

**Example :**                    Set memory between 10 and 20 to zero  
 FOR v0 10 20  
 sm v0 0  
 ENDFOR

**FORCE** Reset or halt target

**Syntax**                        FORCE Break/Reset

**Menu**                        ---

**Description**                The FORCE command allows to stop execution of the program (Break parameter), or reset the target processor (Reset parameter).

**Example :**                    Force r

**FUNCTION** Display function in Code window

**Syntax**                        FUNCTION name|address

**Menu**                        ---

**Description**                FUNCTION command displays specified function in the code window in the SOURCE mode.

**GO** Start executing target code up to address

**Syntax** GO [address]  
**Menu** Process → GO, F5  
**Description** The GO command executes code up to a specific address. If the address is not supplied, code will execute until it reaches one of the breakpoints, or it is stopped with the HALT command.

**GOCURSOR** Execute code to cursor within code window

**Syntax** GOCURSOR  
**Menu** F7  
**Description** GOCURSOR command is available only for macro command processor and custom menus of the code window. When invoked, it will set a temporary breakpoint at the current position of the cursor within the code window. If the current position of the cursor cannot be translated to the address value, command will have no effect. This command can be also assigned to the function keys with the ALIAS command.

**GRAPH** Set/PgDn graph memory window address

**Syntax** GRAPH [mem\_address]  
**Menu** ---  
**Description** The GRAPH command will set the starting address of the Graph window to the mem\_address value. If no parameter is specified, the command will cause the Graph window to scroll one page down. This command affects only the last selected Graph window; other Graph windows are not affected. If there is no Graph window opened, the command will open a new window in the graph format.

**HALT** Halt target processor

**Syntax** HALT  
**Menu** Process → HALT, Shift-F5  
**Description** The HALT command will enter the target processor into the debug mode, allowing to access target resources.

**HELP** Get help on a specific item

**Syntax** HELP [keyword]  
**Menu** Help, F1  
**Description** The HELP command will invoke help file associated with the debugger application. If keyword is one of the valid topics, it will be automatically displayed.

**IDCODE** Read JTAG IDCODE

**Syntax** IDCODE  
**Menu** ---  
**Description** The IDCODE command will read and display IDCODE of the target processor. IDCODE is a 32 bit number representing manufacturer, processor type and version.

| IF                 | Conditional operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <pre>IF_ expr command IF_ expr ... [ELSE] ... ENDIF</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Description</b> | <p>The IF command conditionally executes one or more other commands, depending on the value of <i>expr</i>. If value of <i>expr</i> is not zero, then:</p> <ul style="list-style-type: none"> <li>• in the first case the single following command is executed.</li> <li>• in the second case all following commands up to ELSE or ENDIF are executed.</li> </ul> <p>If value of expression is zero and ELSE is present, then only the commands between ELSE and ENDIF are executed.</p> <p>IF commands can be nested up to 10 levels. Nesting more than 10 IFs will result in an error message.</p> <p>If the command is read from the keyboard, the prompt displays the current nesting level by adding one colon for each level. Hitting &lt;END&gt; will result in an exit from all nesting.</p> |
| <b>Example</b>     | <pre>de -m -f run_and_see reset sb 110 v0=1 .loop go dm @1+v0 v0=v0+1 if (v0&lt;=A) jump loop endm</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

| <b>INPUT</b>       | <b>Open the file and pass data to the target</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | _INPUT [#(file_number)] OFF   TERM   filename [-dec   -hex   -bin [-cnt]]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Description</b> | <p>The INPUT command is used to open a text or binary file and use the file to pass data to the target DSP chip. The data is passed to the target DSP when the user's program reaches a software breakpoint. The file lists the data sequentially.</p> <p>The INPUT command with no parameters displays all open input files.</p> <p># file_number is the number of the file opened (multiple files can be opened simultaneously). The file number is optional, the range is 1 to 99. OFF closes an opened input file. If a file is not specified, all opened files are closed.</p> <p>TERM uses the terminal/keyboard (instead of a text file) to input data to the target DSP.</p> <p>filename is the name of the file used for data input.</p> <p>dec specifies decimal ASCII data representation</p> <p>hex specifies hexadecimal ASCII data representation. This is the default.</p> <p>bin specifies binary data representation</p> <p>cnt specifies number of bytes to read per each target word. It applies only to the binary format.</p> <p>The transfer of the data is invoked from the debugger by the EXEIO command. This command expects transfer parameters to be stored in the global variables: v20..v24.</p> <p>To automate data transfer, user needs to set a breakpoint within the DSP code, and specify "execute" and "test" options for this breakpoint. Command file, invoked with the breakpoint's "execute" option, will initialize global variables, and invoke debugger transfer routine with the EXEIO command. The breakpoint's "test" option will evaluate result of the data transfer, and if it was successful, it will restart the DSP. Global variables can be initialized using variables/registers from the user program, or setup by the user command file.</p> <p>Global variables definition:</p> <p>v20 transfer starting address within the DSP memory</p> <p>v21 number of words to transfer</p> <p>v22 direction (0x8000 - write, 0x4000 - read) and memory space (0 - PMEM, 1 - XMEM, 2 - YMEM)</p> <p>v23 file # to use</p> <p>v24 error flag set by transfer routine</p> |

Details on the INPUT data file in text formats

The text file is always ASCII, with the data listed sequentially. BoxView provides ways to simplify the file editing. For example, if one word must be send to the DSP repeatedly, you can have a repeat code instead of typing an infinitely long file.

The following are the codes that may be used in the ASCII file:

- # specifies the number of times a data word is repeated.
- () parenthesis are used to group words. If parenthesis are preceded by # the whole group is repeated. If # does not precede the parenthesis, the group of words is repeated indefinitely.

- Example 1:* 1 2 3 4  
Send data words 1, 2, 3, 4.
- Example 2:* 121#10  
Send the data word 121 10 times.
- Example 3:* (0 121)#10  
Send 0, 121, 1, 121 ....
- Example 4:* (7fff)a  
Send 7fff indefinitely
- Example 5:* 1 2 3 5#2 (10 20)#3 (10)  
send 1, 2, 3, 5, 5, 10, 20, 10, 20, 10, 20, 10, 10,  
10, 10, ...

**Examples on entering the input command**

- Example 1:* Display all currently open input files.  
INPUT
- Example 2:* Open "DATA.IN" file, label it file number 1. Data in "DATA.IO" is in decimal.  
INPUT #1 DATA.IO -dec
- Example 3:* Open DATA3.BIN file, label the file automatically. Data in DATA3.BIN is in binary format, read one byte per DSP word  
INPUT DATA3.BIN -b -1
- Example 4:* Read data from the terminal (keyboard) and send it to the DSP.  
INPUT TERM

**4**

**Example of the command file initializing global variables and breakpoint for INPUT transfer:**

```

INPUT off ;switch off old input streams
INPUT data_1 ;this will open an input file
BREAK p:110 xset_in tv24!=0 ;set a breakpoint, execute transfer,
 ;run target

eval v20=0x1040 ;start address to load
eval v21=0x10 ;read 16 words at a time
eval v22=0x8002 ;write to YMEM
eval v23=1 ;read from file #1
eval v24=0 ;error flag set if transfer fails

```

**Example of the command executed, when the breakpoint at address p:110 is reached:**

```

exeio ;execute transfer
eval v20+=v2 ;increment destination address
break disable ;disable breakpoints
step ;step over the breakpoint
break enable ;reenable the breakpoints

```

See also EXEIO, OUTPUT

**4**

|                    |                                                                                                                                                                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>JUMP</b>        | <b>Go to label</b>                                                                                                                                                                                                                                                                                                                          |
| <b>Syntax</b>      | <u>JUMP</u> [condition] label                                                                                                                                                                                                                                                                                                               |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                         |
| <b>Description</b> | The JUMP command programs execution. It continues at the specified label if the condition is true (!= 0). If no condition is specified, then it realizes an unconditional jump.<br>Labels can be defined in a command file. They must start with a point (.) and can be up to 80 characters long. A label MUST be the only entry on a line. |
| <b>Example</b>     | <pre> JUMP display_regs JUMP m(0)==2 error_label .display_regs .error_label         </pre>                                                                                                                                                                                                                                                  |

| LINE               | Display assembler source line                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code>LINE [line_number] [file]</code>                                                                                                                                                                                          |
| <b>Menu</b>        | ---                                                                                                                                                                                                                             |
| <b>Description</b> | The LINE command displays lines: line_number-4, ... to line_number+6, of the source file . By default, if line_number is not specified then it takes the current PC, and if file is is not specified, takes the current loaded. |
| <b>Example</b>     | Line 10<br>See also ADDR                                                                                                                                                                                                        |

| LISTSYMBOL         | Display symbols and macros                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code>LISTSYMBOL {-a[space]   -m}</code><br><code>LISTSYMBOL {-a[space]   -m} pattern</code>                                                                                                                                                                                                                                                                                                         |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Description</b> | The LISTSYMBOL command lists all the symbols of the given type (address or macro) in alphabetical order along with their definition (value). To list symbols in a specific memory space, specify the memory space as "x:", "y:" or "p:"<br>The expression <i>pattern</i> is built with normal symbol names and characters:<br>"*" matches any sequence of characters;<br>"?" matches any characters. |
| <b>Example</b>     | <code>li -a (displays X, Y and P space symbols)</code><br><code>li -ax: (displays X space symbols)</code><br><code>li -ax: lab (displays X space symbol "lab")</code><br><code>li -m (displays macros)</code><br><code>li -a:p _str* (displays P space symbols _str...)</code><br><code>li -a:x _std?? (displays P space symbols _stdin ...)</code><br>See also ARCHIVE, DEFINE, UNDEFINE.           |

**LOAD** Load program and symbols

**Syntax** `LOAD filename [P/X/Y]`

**Menu** File → Load

**Description** The LOAD command loads an object file to the target processor and its associated symbol table into memory. The LOAD command is equivalent to the RELOAD and SLOAD commands. If the filename has no extension, system will add automatically the .cld extension. The LOAD command clears the old symbol table.  
For .S19 type files memory space can be specified ( default P space ).

**LOCATE** Set cursor position

**Syntax** `LOCATE line column`

**Menu**

**Description** The LOCATE command positions the cursor.  
*line* must be in the range 1 to 25 and *column* must be in the range 1 to 80.

**Example** `locate 3 15`

**LOG** Open/Close session log file

**Syntax** `LOG [OFF | filename]`

**Menu** ---

**Description** Log command allows to open or close the log file. All output to the terminal window is automatically saved into the log file.

**MAP** Configure memory mapping

**Syntax** `MAP [{ON|OFF}] [{addr_block, type|-r}]`

**Menu** ---

**Description** The MAP command sets mapping for memory address ranges. Enabling mapping option, will enable READ and/or WRITE accesses to selected memory areas, while undefined memory blocks will not be accessed by the emulator.

ON enable memory mapping  
 OFF disable memory mapping  
 -r remove specified meory block or all blocks  
 addr\_block address range to apply mapping mode  
 type type of access for the specified block.

Supported types are: RAM, ROM, READ, WRITE

**MB** Modify conditional hardware breakpoints

**Syntax** `MB {num | *} [-off|-on|-z] [{-a [comp] spc:addr [acc] [and|then] [comp] addr [acc]} | {-d{x|y}comp data [acc]}] [and|then] [ ... ] [And] [ ... ] [-c count ] [-s] [-t] [-x command ]`  
 with acc = {rw|r|w}  
 comp = {=|<|=|>|=|!}  
 spc = {p|x|y}

**Menu** ---

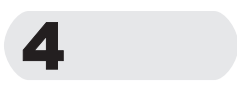
**Description** Modifies a conditional hardware breakpoint pattern previously defined (up to 4) or displays the conditional hardware breakpoint settings. If no parameters are defined, MB displays the current memory conditional hardware breakpoint settings.

**Parameters:**

- *num* Breakpoint number (1 to 4)
- \* All breakpoints.
- -off/-on Breakpoint is set but inactive/active.
- -z Breakpoint is cleared.
- -a *spc*: Breakpoint address space (p by default, x or y).
- *addr* Breakpoint address value.
- -dx -dy Breakpoint data space (x or y).
- *data* Breakpoint data value.
- = <= >= != Breakpoint value comparison (= by default).
- rw r w Breakpoint value access (rw by default).
- and then Breakpoint condition (and by default).
- -c *count* Breakpoint when count=1, otherwise count is decremented.
- -s No breakpoint, but snapshot samples recording.
- -t Breakpoint trap to stop program execution to display breakpoint message or to execute a command (option -x and except GO, NEXT, RUN), then resumes execution.
- -x *command* Emulator command executed after a breakpoint. See also SB

**MEM** Set memory address or Scroll Page Down Memory window

**Syntax** MEM [address]  
**Menu** View → Symbols Dialog  
**Description** The MEM command will set starting address of the Memory window to the address value. If no parameter is specified, command will cause to scroll Memory window one page down Command affects only the last selected Memory window, other Memory windows are not affected.



**MIX** Change Code window display to the mixed mode

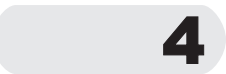
**Syntax** MIX  
**Menu** Local code window menu  
**Description** The MIX command will change display mode of the last selected Code window to mixed mode. If the source line information is not available command will have no effect.

**MM** Move memory

**Syntax** MM *from to* [count]  
**Menu** ---  
**Description** The MM command allows to copy *count* data of memory from address *from* to address *to*. *space* may be "x:", "y:" or "p:". Default value for *space* is "p:". Default value for *count* is 1.  
**Example** MM x:10 y:20 4

**NEXT** Execute next statement

**Syntax** NEXT [count]  
**Menu** Process → Next, F10  
**Description** The NEXT command will execute the single step, or jump over the subroutine calls. The NEXT command can be accessed also with the function key F10. Count specifies number of the statements to be executed before refreshing the debugger windows.



| OPEN               | Open i/o channel                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code>OPEN channel file mode format [length]</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Description</b> | <p>Opens an input or output channel with a file, for subsequent reading/writing data from/to the file.</p> <ul style="list-style-type: none"> <li>• <i>channel</i> can be a decimal number from 1 to 10.</li> <li>• <i>file</i> may be the name of any accessible file or the reserved word TTY, in which case the keyboard or the terminal, whichever is appropriate, is used.</li> </ul> <p><i>mode</i></p> <ul style="list-style-type: none"> <li>• INPUT the channel is opened for reading from the file;</li> <li>• OUTPUT the channel is opened for writing to the file;</li> <li>• APPEND the channel is opened for writing at the end of the file.</li> </ul> <p><i>format</i></p> <ul style="list-style-type: none"> <li>• ASCII data read or written should be interpreted in ASCII for-mat;</li> <li>• BINARY data should be interpreted in binary format.</li> <li>• <i>length</i> expresses the number of bytes read or written in the file. It can be 1, 2, 3 or 4; default value is 1.</li> </ul> <p>The content of a file read through an ASCII channel should be a sequence of hexadecimal numbers separated by one or more spaces, tabs or newline characters.</p> <p>When reading a TTY channel, the numbers will be read from the keyboard one for each line. Data read or written are truncated if value exceeds the <i>length</i> given with command.</p> <p>Once opened for INPUT a channel may be read by using the # operator followed by the channel number (e.g., #1) and the data read may be assigned to any variable handled by the expression evaluator.</p> |
| <b>Example</b>     | <pre>de -m test_auto open 1 address input ascii 2 open 2 result output binary for v0 1 A 1 v1=#1 #2=mx:v1 endfor close 1 close 2 endm</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

| <b>OUTPUT</b>      | <b>Open the file and save data from the target</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code>OUTPUT [#(file number)] OFF   TERM   filename [-dec   -hex   -bin] [-ovr] [-cnt]</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Description</b> | <p>The OUTPUT command is used to open a text or binary file and use the file to save data passed from the target DSP chip. The data is passed to the file when the user's DSP program reaches a software breakpoint.</p> <p>The OUTPUT command with no parameters displays all open output files.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| #(file number)     | <p>is the number of a file opened (multiple files can be opened simultaneously). The file number is optional, the range is 1 to 99. The ASCII text file lists the data sequentially.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| OFF                | <p>closes an opened output file. If a file is not specified, all opened files are closed.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| TERM               | <p>uses the terminal/monitor (instead of a text file), and display the data.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| filename           | <p>is the name of the file used for data output.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| dec                | <p>specifies decimal ASCII data representation</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| hex                | <p>specifies hexadecimal ASCII data representation. This is the default.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| bin                | <p>specifies binary data format</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| ovr                | <p>Overwrite file if the file exists.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| cnt                | <p>Column count for ASCII formats (1..8), or byte count to store into the file for every target word read in binary format (1..2)</p> <p>The transfer of the data is invoked from the debugger by the EXEIO command. This command expects transfer parameters to be stored in the global variables: v20..v24. To automate data transfer, user needs to set a breakpoint within the DSP code, and specify "execute" and "test" options for this breakpoint.</p> <p>Command file, invoked with the breakpoint's "execute" option, will initialize global variables, and invoke debugger transfer routine with the EXEIO command. The breakpoint's "test" option will evaluate result of the data transfer, and if it was successful, it will restart the DSP. Global variables can be initialized using variables/registers from the user program, or setup by the user command file.</p> |

4

**Global variables definition:**

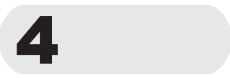
|     |                                                                                           |
|-----|-------------------------------------------------------------------------------------------|
| v20 | transfer starting address within the DSP memory                                           |
| v21 | number of words to transfer                                                               |
| v22 | direction (0x8000 - write, 0x4000 - read) and memory space (0 - PMEM, 1 - XMEM, 2 - YMEM) |
| v23 | file # to use                                                                             |
| v24 | error flag set by transfer routine                                                        |

**Examples on entering the output command:**

- Example 1:* Display all currently open output files.  
`OUTPUT`
- Example 2:* Open "DATA.OUT" file, label it file number 1, and use the file to save data. Data is saved in decimal with one word per line.  
`OUTPUT #1 DATA.OUT -d -1`
- Example 3:* Open DATA4.BIN file, overwrite old one if exists, label the file automatically. Data in DATA4.BIN is in the binary format using 2 bytes for every DSP word.  
`OUTPUT DATA4.BIN -b -2 -o`
- Example 4:* Read data from the DSP and send it to the Terminal window in hexadecimal format, 4 words per line.  
`OUTPUT TERM -h -4`

**Example of the command file initializing global variables and breakpoint for OUTPUT transfer**

```
OUTPUT off ;switch off old input streams
OUTPUT data4.txt -o ;this will create an output file (overwrite
existing file)
BREAK p:120 xset_io tv24!=0 ;set a breakpoint, execute transfer, run target
```



**Example of the command executed, when the breakpoint at address p:120 is reached**

```
eval v20=addrStart ;get the start address to read from the
 ;application variable
eval v21=xferCnt ;initialize count variable
eval v22=0x4001 ;read from XMEM
eval v23=1 ;write to file #1
eval v24=0 ;error flag set if transfer fails
exeio ;execute transfer
break disable ;disable breakpoints
step ;step over the breakpoint
break enable ;reenable the breakpoints
```

See also EXEIO, INPUT

| <b>PATH</b>        | <b>Define, display or remove search paths</b>                                                                                                                                                                                                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <code>_PATH [pathname[,OFF]]</code>                                                                                                                                                                                                                                                                                                                                    |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Description</b> | The PATH command with no parameter will display list of the defined search paths for the system. Search paths are used to locate object files, source files and alias commands. If the pathname is specified, it will be added to the list of the search paths. The OFF parameter will remove specified pathname or clear all the path list if the pathname was empty. |

**PCCURSOR** Set PC value at cursor within code window

**Syntax** `PCCURSOR`

**Menu** ---

**Description** PCCURSOR command is available only for macro command processor and custom menus of the code window. When invoked, it will change value of the target's PC to the current position of the cursor within the code window. If the current position of the cursor cannot be translated to the PC value, command will have no effect.  
This command can be also assigned to the function keys with the ALIAS command.

**PAUSE** Pause for a number of seconds

**Syntax** `PAUSE secs`

**Menu** ---

**Description** Temporization command, waits for `secs` seconds.

**Example** `Pause 10`

**PCHIST** Program counter history

**Syntax** `PCHIST`

**Menu** ---

**Description** PCHIST command displays contents of the emulator's trace buffer

**Example**

```
+ pchist
P:0122 JUMP $103
P:0103 IX0 = #$1
P:0105 IX1 = #$1
P:0107 IY0 = #$1
P:0109 STA = #$C
P:010B AX0 = #$40
```

| <b>PRINT</b>       | <b>Print strings and values</b>                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <pre><u>Print</u> [-x   -d   -o] [-n] [-r] [expr] ... <u>Print</u> [-x   -d   -o] [-n] [-r] -f "C-like-format" [expr] ...</pre>                                                                                                                                                                                                                                                                               |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Description</b> | <p>The PRINT command allows to print (display) strings and values. It may be useful when used in macros to print messages and to convert the base of numbers.</p>                                                                                                                                                                                                                                             |
| -x                 | specifies the hexadecimal output format.                                                                                                                                                                                                                                                                                                                                                                      |
| -d                 | specifies the decimal output format.                                                                                                                                                                                                                                                                                                                                                                          |
| -o                 | specifies the octal output format.                                                                                                                                                                                                                                                                                                                                                                            |
|                    | <p>Note that expressions in the list of arguments are always interpreted according the current base, independently from options -x, -d and -o.</p>                                                                                                                                                                                                                                                            |
| -n                 | omits a newline to be printed at the end of a line.                                                                                                                                                                                                                                                                                                                                                           |
| -r                 | displays in reverse video mode.                                                                                                                                                                                                                                                                                                                                                                               |
| -f                 | "C-like-format" allows to format output. The same format is applied to all <i>expr</i> parameters. Follows the ANSI C syntax.                                                                                                                                                                                                                                                                                 |
| <b>Example</b>     | <pre>print -r "TITLE IN REVERSE" print -f "value = %04x" 0x0A --&gt; value = 000A print -f "value = %04d" 0x0A --&gt; value = 0010 print -f "value = %4x" 0x0A --&gt; value = A print -f "value = %4d" 0x0A --&gt; value = 10 print "value of stack pointer register =" r\$SP print -d "LOOP COUNTER VALUE=" m\$LOOPCNT --&gt; print value (in decimal format) at the address defined by symbol LOOPCNT</pre> |

# 4

| <b>QUIT</b>        | <b>Exit application</b>                                                                                                          |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>QUIT</u>                                                                                                                      |
| <b>Menu</b>        | File → Exit                                                                                                                      |
| <b>Description</b> | <p>The QUIT command will close target connection (if opened), save current windows configuration, and close the application.</p> |

|                    |                                                                                                                                                                                                                                                                             |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>REFRESH</b>     | Refresh display window                                                                                                                                                                                                                                                      |
| <i>Syntax</i>      | <u>REFRESH</u> ]                                                                                                                                                                                                                                                            |
| <i>Menu</i>        | ---                                                                                                                                                                                                                                                                         |
| <i>Description</i> | The REFRESH command will force update of all debugger windows.                                                                                                                                                                                                              |
| <b>RELOAD</b>      | Reload program code (no symbols)                                                                                                                                                                                                                                            |
| <i>Syntax</i>      | <u>RELOAD</u> [filename]                                                                                                                                                                                                                                                    |
| <i>Menu</i>        | ---                                                                                                                                                                                                                                                                         |
| <i>Description</i> | The RELOAD command will load last loaded code to the target processor. RELOAD command leaves symbol table intact.                                                                                                                                                           |
| <b>RESET</b>       | Reset target processor                                                                                                                                                                                                                                                      |
| <i>Syntax</i>      | <u>RESET</u>                                                                                                                                                                                                                                                                |
| <i>Menu</i>        | Process → Reset                                                                                                                                                                                                                                                             |
| <i>Description</i> | The RESET command will reset the target processor to debug mode. This is the hardware reset using the physical processor reset signal. In multiprocessor systems it can affect other processors. For the software reset only, appropriate macro command should be executed. |
| <b>RESTART</b>     | Reset program to its entry point                                                                                                                                                                                                                                            |
| <i>Syntax</i>      | <u>RESTART</u>                                                                                                                                                                                                                                                              |
| <i>Menu</i>        | Process → Restart                                                                                                                                                                                                                                                           |
| <i>Description</i> | The RESTART command resets the program to its entry point. This command assumes, that program is already loaded with the LOAD command, or symbolic information is entered with the SLOAD command.                                                                           |

**RETURN** Exit from subroutine

**Syntax** RETURN  
**Menu** Process → Return  
**Description** The RETURN command will execute code in the current C function, and halts when execution reaches the caller.

**RUN** Execute multiple instructions

**Syntax** RUN [count]  
**Menu** Process → GO, F5  
**Description** The RUN command executes count of instructions using on target's "trace counter". If the count is omitted, the RUN command will behave as the GO command with no parameter.

**SAVE** Save memory block(s) to the file

**Syntax** SAVE address\_range [address\_range] filename [options]  
**Menu** ---  
**Description** The SAVE command saves memory blocks specified by one of more definitions of the address\_range to the ASCII file in the OMF format (extension .LOD). This data file can be loaded back to the processor memory with the LOAD command.

*address\_range* specifies memory area to be changed to the value. Range can be accepted in two formats: start\_address..end\_address or start\_address#count.register

**Options:**

- o overwrite an existing file
- a append to the existing file
- x save in binary format
- [1..8] number of words per line (default: 8)
- h hexadecimal format (default)
- d decimal format

**SB** Set/Display conditional hardware breakpoints

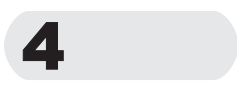
**Syntax** SB  
 SB *num* [-off|-on]  
 [{-a [comp] spc: *addr* [acc] [and|then]  
 [Comp] *addr* [acc]} | {-d{x|y}comp *data* [acc]}]  
 [and|then]  
 [ ... ]  
 [and]  
 [ ... ]  
 [-C *count*] [-s] [-t] [-x *command* ]  
 with acc = {rw|r|w}  
 comp = {=|<=|>=|!=}  
 spc = {p|x|y}

**Menu** ---

**Description** The SB command sets a new conditional hardware breakpoint (up to 4) or to display the conditional hardware breakpoint settings. These hardware breakpoints work in a real time context. If no parameters are defined, SB displays the current memory conditional hardware breakpoint settings.

**Parameters:**

- *num* Breakpoint number (1 to 4)
- -off Breakpoint is set but inactive.
- -on Breakpoint is set and active.
- -a spc: Breakpoint address space (p by default, x or y).
- *addr* Breakpoint address value.
- -dx -dy Breakpoint data space (x or y).
- *data* Breakpoint data value.
- = <= >= != Breakpoint value comparison (= by default).
- rw r w Breakpoint value access (rw by default).
- and then Breakpoint condition (and by default).
- -c *count* Breakpoint when count=1, otherwise count is decremented.
- -s No breakpoint, but snapshot samples recording.
- -t Breakpoint trap to stop program execution to display breakpoint message or to execute a command (option -x and except GO, NEXT, RUN), then resumes execution.
- -x *command* Emulator command executed after a breakpoint.



### Buses and conditions:

-a p: *addr*  
-a p: *addr* AND *addr*  
-a p: *addr* AND -a x: *addr*  
-a p: *addr* AND -a x: *addr* AND *addr*  
-a p: *addr* AND -a x: *addr* AND -a y: *addr*  
-a p: *addr* AND -a y: *addr*  
-a p: *addr* AND -a y: *addr* AND *addr*  
-a p: *addr* THEN *addr*  
-a p: *addr* THEN -a x: *addr*  
-a p: *addr* THEN -a x: *addr* AND *addr*  
-a p: *addr* THEN -a x: *addr* AND -a y: *addr*  
-a p: *addr* THEN -a y: *addr*  
-a p: *addr* THEN -a y: *addr* AND *addr*  
-a x: *addr*  
-a x: *addr* AND *addr*  
-a x: *addr* AND -a y: *addr*  
-a x: *addr* AND -dx= *data*  
-a x: *addr* THEN *addr*  
-a x: *addr* THEN -a y: *addr*  
-a x: *addr* THEN -dx= *data*  
-a y: *addr*  
-a y: *addr* AND *addr*  
-a y: *addr* AND -dy= *data*  
-a y: *addr* THEN *addr*  
-a y: *addr* THEN -a x: *addr*  
-a y: *addr* THEN -dy= *data*  
-dx= *data*  
-dx= *data* AND -dy= *data*  
-dy= *data*

## 4

There are only 4 internal DSP breakpoint registers, limiting hardware break-point resources. Only one internal DSP counter exists and it may be affected by 1 to 4 break-points (the count value written is the last supplied).

The snapshot mode uses conditional breakpoint logic to record bus samples. In this case the end of program execution occurs when the asked samples number is reached and recorded.

When a **breakpoint** is met, **1 or 2 instructions are executed**.

Remember that one program address breakpoint can be used by simple soft-ware breakpoints.

Do not use SB to modify an existing breakpoint, instead use the MB command.

*Example*

```

reset
load test_bp
sb 1 -a p:100
sb 2 -off -a 200 and -a <= x:300 w -t -x line
sb 3 -dy<= 400 r -c 10
sb
go
dr
mb 1 -z
mb 2 -on
mb
go
dr
mb * -z

```

See also MB, SBPUT

**SBDEL**

Delete simple software breakpoints

**Syntax** SBDeI <addr 1> [<addr 2> ... ]  
SBDeI \*

**Menu** ---

**Description** The SBDEL command deletes simple software breakpoints located at the specified addresses.  
\* means all breakpoints.

**Example** SBDEL \$\_swi 200  
See also SB, SBPUT

| SBPUT              | Put/Display simple software breakpoints                                                                                                                                                                                                                                                                                                            |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | SBPut <i>addr 1</i> [ <i>addr 2 ...</i> ]<br>SBPut                                                                                                                                                                                                                                                                                                 |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                |
| <b>Description</b> | <p>The SBPUT command sets simple software breakpoints located at the specified addresses.</p> <p>If no parameters are defined, the command reports all the active breakpoints.</p> <p>Note: these software breakpoints do not work in a real time context. They need one program address hardware breakpoint and three words in a valid stack.</p> |
| <b>Example</b>     | <pre>SBPUT \$_swi 200</pre> <p>See also SB, SBDEL</p>                                                                                                                                                                                                                                                                                              |

| SEARCH             | Search a pattern in memory                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>SEARCH</u> [ <i>space</i> ] <i>from to pattern</i>                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Description</b> | <p>The SEARCH command searches the value <i>pattern</i> in memory, between <i>from</i> and <i>to</i> addresses.</p> <p><i>space</i> may be "x:", "y:" or "p:". Default value for <i>space</i> is "p:", i.e., program space.</p> <p>The <i>pattern</i> can consist of 1 to 8 words</p> <p>If the <i>pattern</i> is found, the display is preceded by the address of memory where it has been found. The format is similar to that of DM command</p> |
| <b>Example</b>     | <pre>SE x:100 200 FF00</pre> <p>See also DM</p>                                                                                                                                                                                                                                                                                                                                                                                                    |

4

| SET                              | Set/reset options                                                                                                                                                                                                                                                                      |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>                    | <u>SET</u> [ option [argument]]                                                                                                                                                                                                                                                        |
| <b>Menu</b>                      | ---                                                                                                                                                                                                                                                                                    |
| <b>Description</b>               | Displays or changes the setting of the emulator.<br><b>option</b> can be: DSPVER, ECHO, GOFROM, LOG, SOURCE, VERBOSE, MECHO, TAB , WATCH , TIMER, TIME24, ERRORS, TTYLOG, TTYWRAP, SCROLL<br><b>argument</b> can be OFF, ON, or value                                                  |
| <i>DSPVER n</i>                  | decimal value representing version of the ST18950 silicon.                                                                                                                                                                                                                             |
| <i>TAB n</i>                     | decimal value representing tab character size in spaces (1..32)                                                                                                                                                                                                                        |
| <i>WATCH ms</i>                  | delay time in ms, for displaying instant watch values within code window                                                                                                                                                                                                               |
| <i>TIMER ms</i>                  | delay time in ms between checking the target status. Decreasing this time will accelerate single step operation, but it will use more PC's CPU time.                                                                                                                                   |
| <i>TIME24 ON OFF</i>             | If the TIME24 option is enabled (ON), time of file creation (DIR command) will be displayed in the 24 hour format.                                                                                                                                                                     |
| <i>ERRORS ON OFF</i>             | enabled option will stop execution of the command file, if the error is detected. If the ERRORS option is set to OFF, macro command will continue.                                                                                                                                     |
| <i>TTYLOG ON OFF</i>             | enabled option, will LOG all commands to the Terminal window. If the file logging is enabled, this option is also automatically enabled.                                                                                                                                               |
| <i>TTYWRAP ON OFF</i>            | enabled option, will wrap long lines when the right side of the terminal window is reached                                                                                                                                                                                             |
| <i>SCROLL x</i>                  | hexadecimal value limiting window's scroll range. Value can be between 0x100 and maximum address value.                                                                                                                                                                                |
| <i>GOFROM ON OFF</i>             | enabled option will interpret the address for the GO command, as a start address - it will change PC value before restarting target. Default is OFF, which treats address, as "goto" address.                                                                                          |
| <i>ECHO ON   OFF</i>             | When set to OFF, no messages are echoed in the terminal window. This command must be used in a command file. When set to ON, all messages are echoed again in the terminal window.                                                                                                     |
| <i>LOG &lt;file&gt; ON   OFF</i> | If <file> is specified this option makes a typescript of everything printed in the terminal window into <file>. When argument is OFF the current LOG file is closed                                                                                                                    |
| <i>SOURCE ON   OFF</i>           | When set to ON, it enables to display the current mnemonic instruction executed loaded from the assembler source file, if there is one. When set to OFF, it enables to display the current mnemonic instruction disassembled. The default selection is OFF when entering the emulator. |

**VERBOSE ON | OFF** When set to ON, any command, either read from the keyboard, from a macro or from a command file is echoed before being executed. The commands are displayed in the terminal window and logged to the log file.

**MECHO ON | OFF** When set to OFF, no commands are echoed to the command window during macro execution. When set to ON, during macro command execution, all commands will be echoed to the command window.

*Example*

```
set source on
set verbose on
set log trace
set
time
set echo off
go
set echo on
time
set log off
```

| SLOAD              | Load symbol table                                                                                                                                                                                                         |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>S</u> LOAD filename                                                                                                                                                                                                    |
| <b>Menu</b>        | Load → File                                                                                                                                                                                                               |
| <b>Description</b> | The SLOAD command loads the symbol table of the specified object file. SLOAD clears existing symbol table before loading the new one. If the filename has no extension, system will add automatically the .cld extension. |

| SM                 | Set memory                                                                                                                                                                                                                                          |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>S</u> M [space] address value                                                                                                                                                                                                                    |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                 |
| <b>Description</b> | The SM command changes the content of the memory at the specified address.<br>Space may be "x:", "y:" or "p:". Default value for <i>space</i> is "p:", i.e., program space.<br>The content of memory at the address specified is set to that value. |
| <b>Example</b>     | cmd> SM x:10 FFFF<br>See also CM, DM, FM                                                                                                                                                                                                            |

4

**SOURCE**

Change code window display mode to source

|                    |                                                                                                                                                                                        |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>SOURCE</u> or <u>SRC</u>                                                                                                                                                            |
| <b>Menu</b>        | Local code window menu                                                                                                                                                                 |
| <b>Description</b> | The SOURCE (or SRC) command will change display mode of the last selected Code window to the source mode. If the source line information is not available command will have no effect. |

**SR**

Set registers

|                    |                                                                                       |
|--------------------|---------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>SR</u> <i>register value</i>                                                       |
| <b>Menu</b>        | ---                                                                                   |
| <b>Description</b> | The SR command sets the register indicated by <i>register</i> with the <i>value</i> . |
| <b>Example</b>     | cmd> SR ax0 10                                                                        |
|                    | <ul style="list-style-type: none"> <li>• See also DR</li> </ul>                       |

**SRC**

Change code window display to the source mode

|                    |                                                                                                                                                                            |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Syntax</b>      | <u>SRC</u>                                                                                                                                                                 |
| <b>Menu</b>        | Local code window menu                                                                                                                                                     |
| <b>Description</b> | The SRC command will change display mode of the last selected Code window to the source mode. If the source line information is not available command will have no effect. |

4

**START**

Reset and start execution

|                    |                                                                       |
|--------------------|-----------------------------------------------------------------------|
| <b>Syntax</b>      | <u>START</u>                                                          |
| <b>Menu</b>        | ---                                                                   |
| <b>Description</b> | START command resets target and starts execution of the user program. |

**STATUS** Check target status

**Syntax** STATUS  
**Menu** ---  
**Description** The STATUS command checks the target processor status. Status is also displayed on the right side of the status bar, on the bottom of the application window.

**STEP** Single step single or multiple instructions

**Syntax** STEP [count]  
**Menu** Process → Step  
**Description** The STEP command will single step the target code. If count is not specified it will step single instruction. For the multi-instruction step, debugger windows will be refreshed after every step. To execute multiple instructions without windows refresh, RUN command should be used.

**STEPOVER** Execute next statement

**Syntax** STEPOVER [count]  
**Menu** Process --> Stepover, F10  
**Description** The STEPOVER command will execute the single step, or jump over the subroutine calls. The STEPOVER command can be accessed also with the function key F10. Count specifies number of the statements to be executed before refreshing the debugger windows.

**STOP** Stop executing code

**Syntax** STOP  
**Menu** ---  
**Description** The STOP command will enter the target processor into the debug mode, allowing to access target resources.

**TILE** Tile all windows

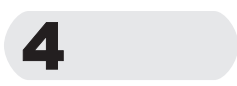
**Syntax** TILE  
**Menu** Window --> Tile  
**Description** Arrange all opened windows in the tile format.

**TIME** Get timing information

**Syntax** TIME [mode]  
**Menu** ---  
**Description** This command allows to display current system time, or time in milliseconds from last TIME command usage.  
 Parameter mode allows for different display formats:  
 0 Display full time and date (default)  
 1 Display time and delta time from last TIME (hh:mm:ss.mmm (msec))  
 2 Display delta time from the last TIME command (milliseconds)  
 3 Clear millisecond delta timer (initialize)

**TIMER** Set/get application timer value

**Syntax** TIMER [value]  
**Menu** ---  
**Description** The TIMER command allows to adjust application idle timer. This timer is used to periodically check for the target processor status.  
 Value represents the idle time in milliseconds.



**UNALIAS** Remove alias for given name

**Syntax** UNALIAS [name]  
**Menu** ---  
**Description** The UNALIAS command will remove all system aliases if parameter was empty.

|                    |                                                                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>UNDEFINE</b>    | <b>Remove symbols</b>                                                                                                                                                     |
| <b>Syntax</b>      | <u>UN</u> DEFINE {-a   -m} symbol...<br><u>UN</u> DEFINE {-a   -m} -all                                                                                                   |
| <b>Menu</b>        | ---                                                                                                                                                                       |
| <b>Description</b> | Removes the <i>symbol</i> of the given type from the symbol table, or clears the entire symbol table if the -all option is specified.<br>-a Memory address.<br>-m Macros. |
| <b>Example</b>     | un -a lab<br>un -m LOADANGO<br>See also ARCHIVE, DEFINE, LISTSYMBOL                                                                                                       |

|                    |                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>USE</b>         | <b>Define, display or remove search paths</b>                                                                                                                                                                                                                                                                                                                         |
| <b>Syntax</b>      | <u>USE</u> [pathname[,OFF]]                                                                                                                                                                                                                                                                                                                                           |
| <b>Menu</b>        | ---                                                                                                                                                                                                                                                                                                                                                                   |
| <b>Description</b> | The USE command with no parameter will display list of the defined search paths for the system. Search paths are used to locate object files, source files and alias commands. If the pathname is specified, it will be added to the list of the search paths. The OFF parameter will remove specified pathname or clear all the path list if the pathname was empty. |

4

|                    |                                                                            |
|--------------------|----------------------------------------------------------------------------|
| <b>VERSION</b>     | <b>Display program version</b>                                             |
| <b>Syntax</b>      | <u>VER</u> SION                                                            |
| <b>Menu</b>        | ---                                                                        |
| <b>Description</b> | The VERSION command will display version numbers of all active components. |

**VIEW** Open View window

**Syntax** `VIEW filename`  
**Menu** ---  
**Description** Open file in the browser window.

**WAIT** Wait for mcsecs or until device is in DEBUG mode

**Syntax** `WAIT time | DEBUG| MAIL`  
**Menu** ---  
**Description** This command allows to suspend macro command execution for specified number of milliseconds (time), or wait until the target device gets into the debug mode (DEBUG) (reaches one of the breakpoints).  
 If parameter *mail* is used it waits until receives acknowledgement for all sent commands to another debuggers. This feature is available only for multi-DSP development systems.

*Example:*

```
Load test.cld
Break main
Go
Wait debug
beep
```

**WATCH** Add watch variable

**Syntax** `WATCH address | register [,label[,mode]]`  
**Menu** View → Symbol  
**Description** The WATCH command will add or remove watch variable from the Watch window.  
*address* should have the following format: space: offset, where space is one of the allowed memory spaces for the processor (P, X, or Y), offset absolute value of the address offset. Address can be also specified in the form of the application symbol.  
*register* is a valid name of one of the target registers

*label* is a string which will be displayed instead of the watch.  
*address mode* is one of the following:

- c - ASCII character
- d - decimal
- e - exponential floating point
- o - octal
- p - address
- s - ASCII string
- u - unsigned decimal
- x - hexadecimal
- r - remove watch from the watch window.

## WINDOW

### Open a new window

**Syntax**

WINDOW type

**Menu**

View --> win\_type

**Description**

Window command allows to open new window of the specified type. Window type can be specified with the partial name only.