AN978
APPLICATION NOTE

KEY FEATURES OF THE STVD7 ST7 VISUAL DEBUG PACKAGE

by Microcontroller Division Application Team

INTRODUCTION

The STVD7 is the brand new debugger developed by STMicroelectronics, which replaces the WGDB7 and which is still free of charge.

This is an IDE (Integrated Development Environment) which means that the same graphical Windows interface can be used for both editing and debugging.

The most recent version of the STVD7 Debugger can be downloaded from the 8-bit MCUs Internet site (http://mcu.st.com at the Free Software page) or from the MCU ON CD CD-ROM. The WGDB7 is still available at the same Free Software page.

The purpose of this application note is to explain how to get started with this ST Visual Debugger and the ST Assembly tool chain and to describe the main features of the STVD7. Similar application notes have been written for the STVD7 and ST7 C Compilers (developed by COSMIC and HIWARE).

The following operating systems are supported: Windows 95, 98, NT, 2000 (soon).

If you need more information about this debugger, please refer to the ST Visual Debug User Manual included with the software in a .pdf version. A complete on-line help service is also available through the Help menu in the STVD7 program.

1 STVD7 AND ASSEMBLY TOOL CHAIN

Once the STVD7 Debugger has been installed, 3 different icons will be displayed in the Windows Start menu: one for the simulator, one for the development kit and one for the emulator.

Click on the corresponding icon to select the required tool.

The first time the STVD7 program is launched (i.e. any of the 3 icons is selected), the tool chains must be configured:

- The ST Assembly Toolchain path is already specified,
- for Hiware the correct path is C:\Hiware\prog (demo or licensed version),
- for Cosmic: C:\Program Files\cosmic software\st7eval\cxst7 (demo version) or C:\COSMIC\ST7 (licenced version) if the default paths were selected during the install procedure.

Click on the Browse button in this window to select the correct path.

Once STVD7 has been launched, a new workspace (*.wsp) must be created:

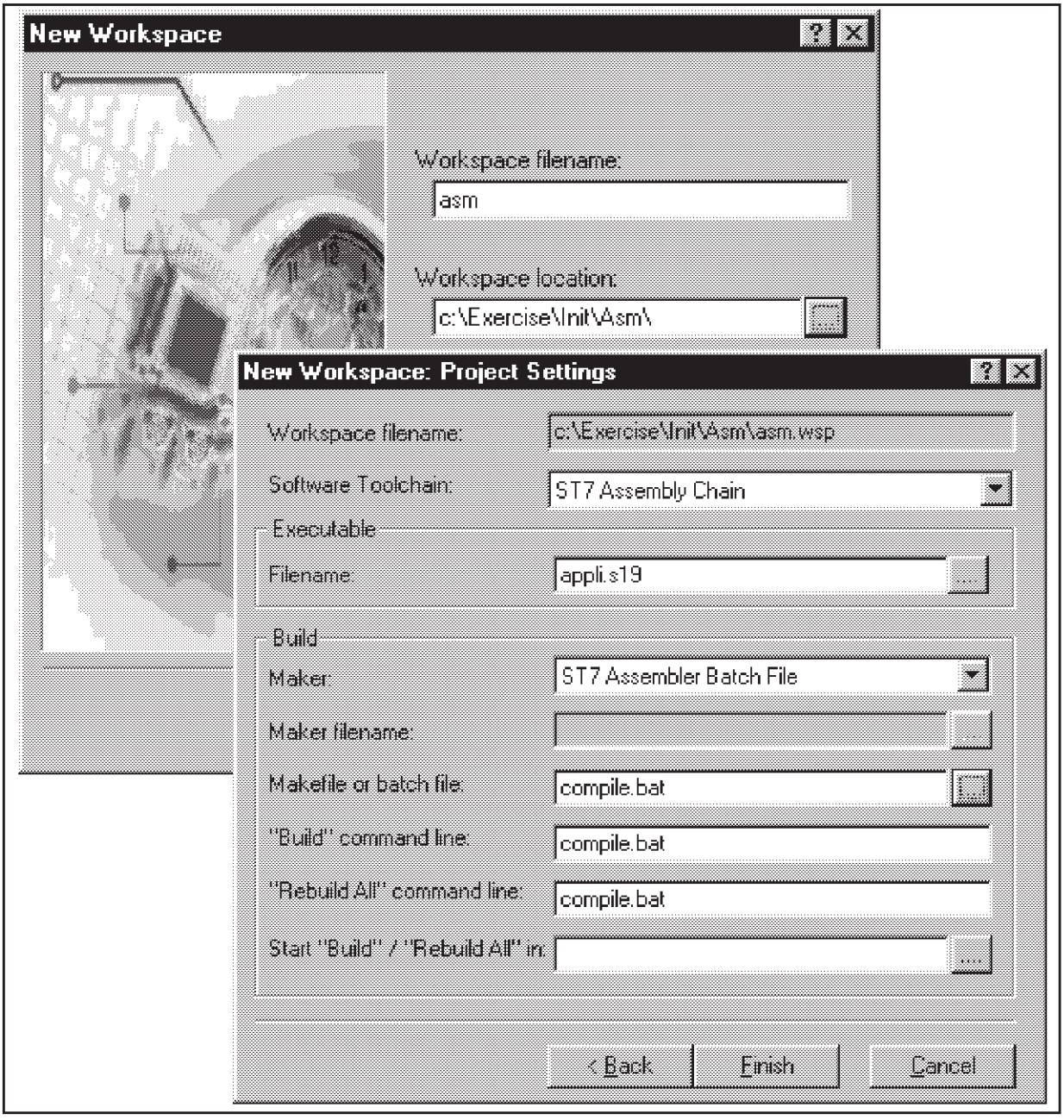
- Click on "***File, New Workspace***" and enter the requested information (see Figure 1.):
  - Workspace Filename: Assign a name to the workspace,
  - Workspace location: Use the Browse function to identify the work directory.

Then, click on ***Next*** and enter the **Project Settings**:

- Software Toolchain: Select one of the following toolchains: ST Assembly, C Hiware or C Cosmic,
- Executable Filename: Name the executable file (.s19 usually) used to launch the debug session. (If this file already exists, use the browse function to identify its location.) It is also possible to return to this window ("***Project, Project Settings***") and enter this field at a later time.
- Maker: Select the type of maker. A default maker is proposed depending on the toolchain selected.
- Make File or Batch File: Assign a name to the make file or batch file, or use the browse function to identify its location. This file will be linked to the Build and Rebuild buttons by default. If other files will be used for the Build and Rebuild commands, enter the correct name instead of the default one listed in the corresponding line.

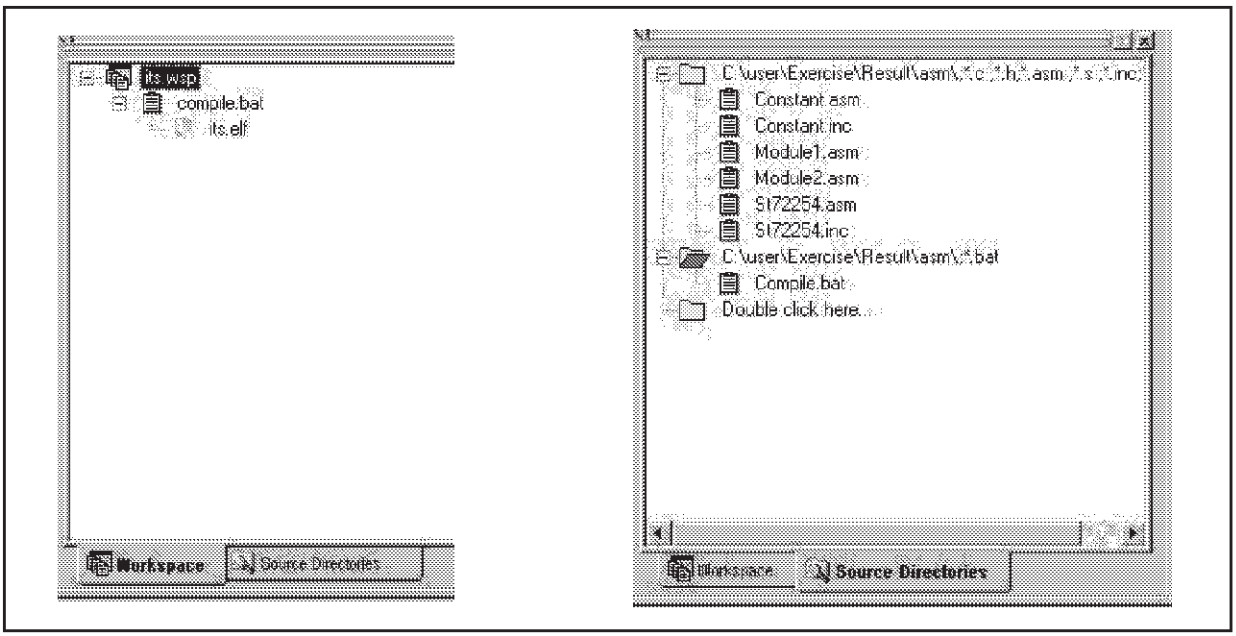
Click on "***Finish***".

Figure 1. STVD7 New Project



- Click on the "*Source Directory*" tab in the Workspace window (see Figure 2.) and double-click on the directory: the work directory is selected by default and the list of source files is displayed in the workspace window. Double-click on the file to be edited. New files can also be created in this window. This workspace window is displayed by default on the left-hand side of the STVD7 window (Edit or Debug modes).

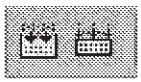
Figure 2. Workspace Window



The project has been created.

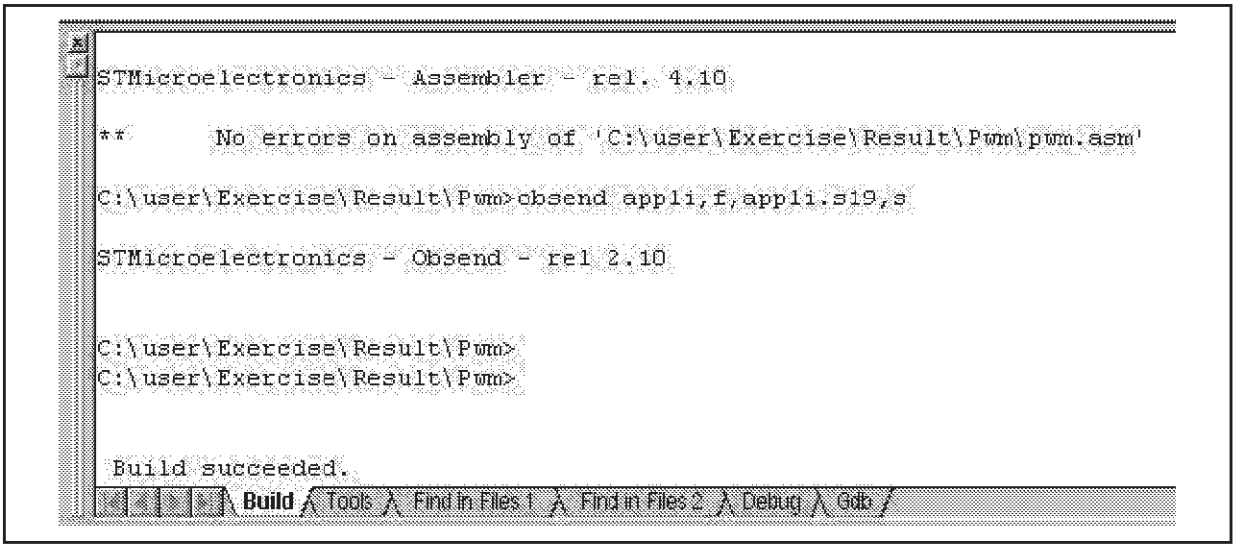
- In order to build, click on the Build or Rebuild button on the "**Project, Build or Rebuild**" menu. These 2 icons may be added to the toolbar using the "**Tools, Option**" menu and clicking on "**Project**" in the **Toolbar** tab.

Figure 3. Build and Rebuild Buttons



The results of all operations, debug messages and other information (emulator/connection information, run/stop information, warning messages) are displayed in the output window, which is located below the source window. See Figure 4.

Figure 4. Output Window

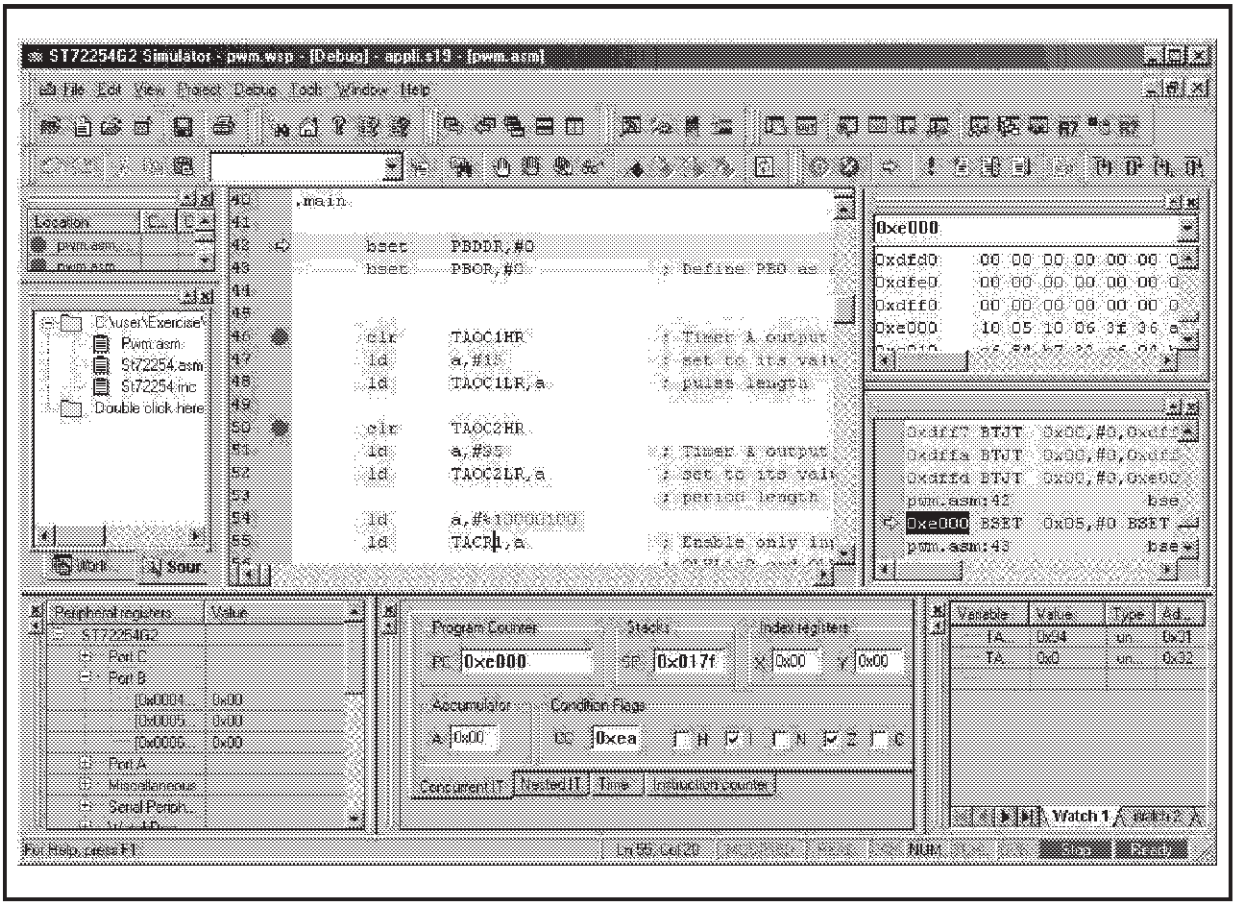


- Detected errors are displayed in the output window. Double-click on the error to be directly placed on it. Correct the error, then rebuild.
- When the build is successful, Debug mode can be entered by clicking on the blue icon shown in Figure 5.

Figure 5. Debug and Edit Mode Buttons

The Edit mode is very easily differentiated from Debug mode in two ways. First, by the pressed button displayed in the toolbar (the blue D is highlighted when in Debug mode or the red cross for Edit mode). Secondly, in debug mode, the line reached by the Program Counter is displayed in yellow (debug environment). See Figure 6.

Figure 6. STVD7

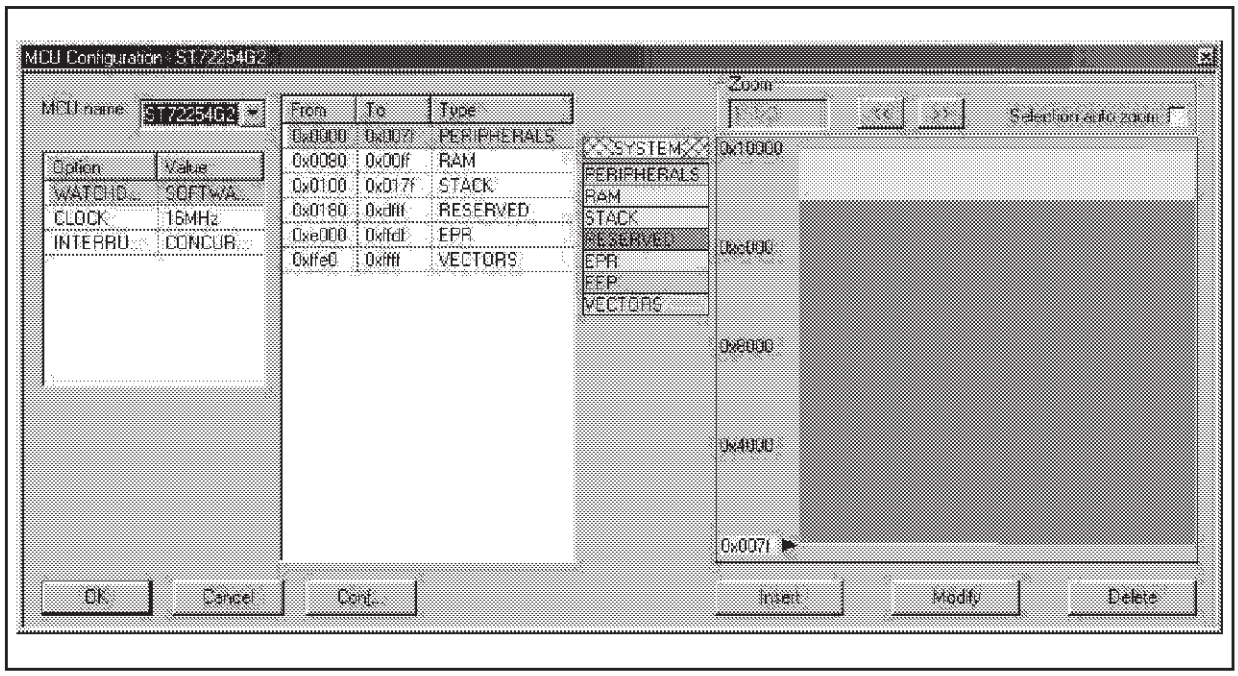


The first time Debug mode is entered, the information in the "**MCU Configuration**" window must be entered (see Figure 7.):

- MCU name
- CPU frequency
- LVD (on, off, level)
- Watchdog (hardware or software)...

All the possible MCU configurations that can be set through the option byte or certain registers when working with the MCU are listed in this window.

Figure 7. MCU Configuration Window



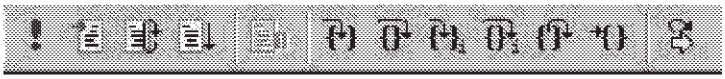
2 STVD7 MAIN FEATURES

The STVD7 has been designed with many debugging features. The entire list of features is described in detail in both the STVD7 User Manual and the Help menu. Only the key features are described in this document. Some of these features are dedicated windows that can be also seen in Figure 6.

2.1 RUN/STOP/STEPS

These buttons are displayed by default in the toolbar. Several steps are possible.

Figure 8. Run/Stop/Step Buttons



Below is the meaning of each of the icons displayed above (from left to right):

- Run
- Chip reset
- Restart application
- Continue
- Stop program
- Step into
- Step over
- Step into ASM
- Step over ASM
- Step out
- Run to cursor
- Set PC

2.2 ST7 REGISTER WINDOW

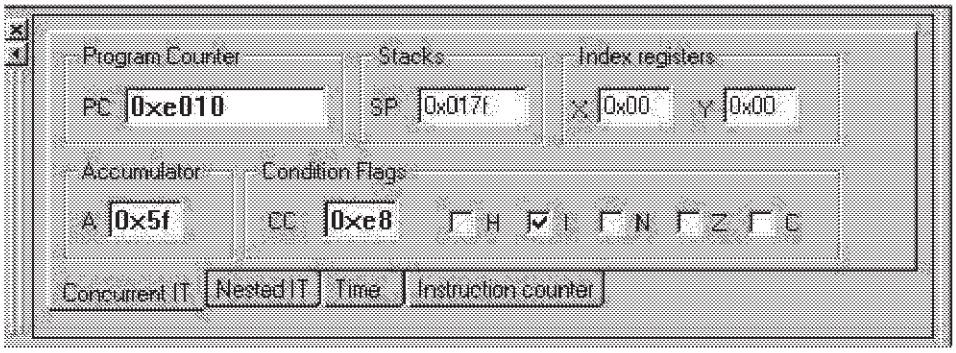
The contents of all ST7 internal registers are displayed in this window.

- Accumulator A,
- Index registers X and Y,
- Program Counter,
- Stack Pointer,
- Condition Code Register (flag bits).

For nested interrupts, a dedicated tab is displayed with the corresponding condition flags (I0 and I1), as well as the I level (between 0 and 3).

For the simulator only, two other tabs, Time and Instruction Counter, are displayed in this window. These tabs are used for measuring times in CPU cycles, which are also converted into seconds depending on the CPU frequency.

Figure 9. ST7 Register Window

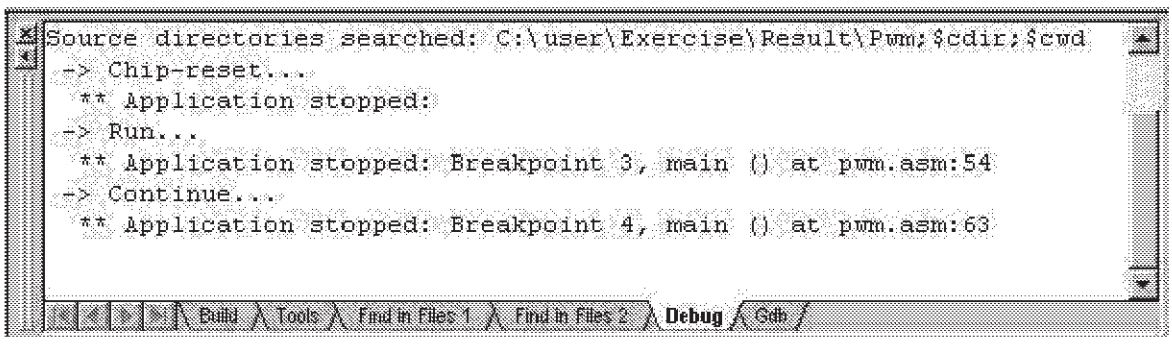


2.3 OUTPUT WINDOW

Several tabs are displayed in this output window:

- Debug: All information relative to the debug session is displayed here (emulator connection, run/stop, warning messages).
- Print: This feature is not available on the current STVD7 version.
- GDB: This is an interface used to directly access the core Gnu Debugger program. A list of commands is available in the Help menu and can be used to write script files for launching automatic debug sessions.
- Build: This feature is not available on the current STVD7 version.

Figure 10. Output Window

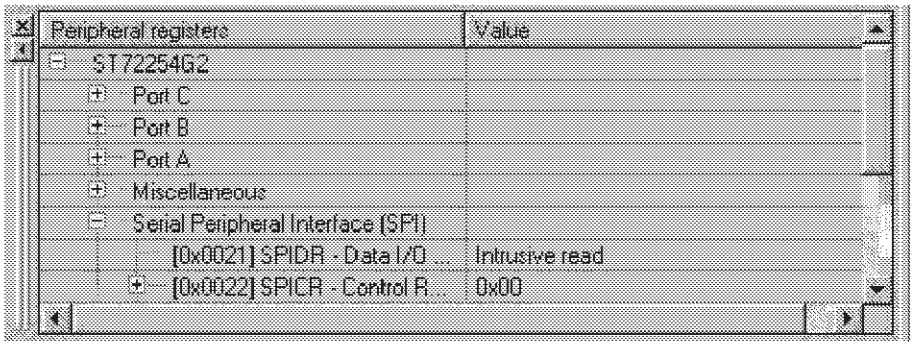


2.4 PERIPHERAL REGISTERS WINDOW

This window displays all the registers of the targeted MCU and their content. Some registers are displayed bit by bit with the corresponding bit names. When certain registers are read, a flag reset sequence may be launched, and an “Intrusive read” message will be displayed. In order to read the register value nevertheless, right-click on the mouse and select “Forced Read” to display the value.

A value may be modified by double-clicking on it, and then overtyping its new value.

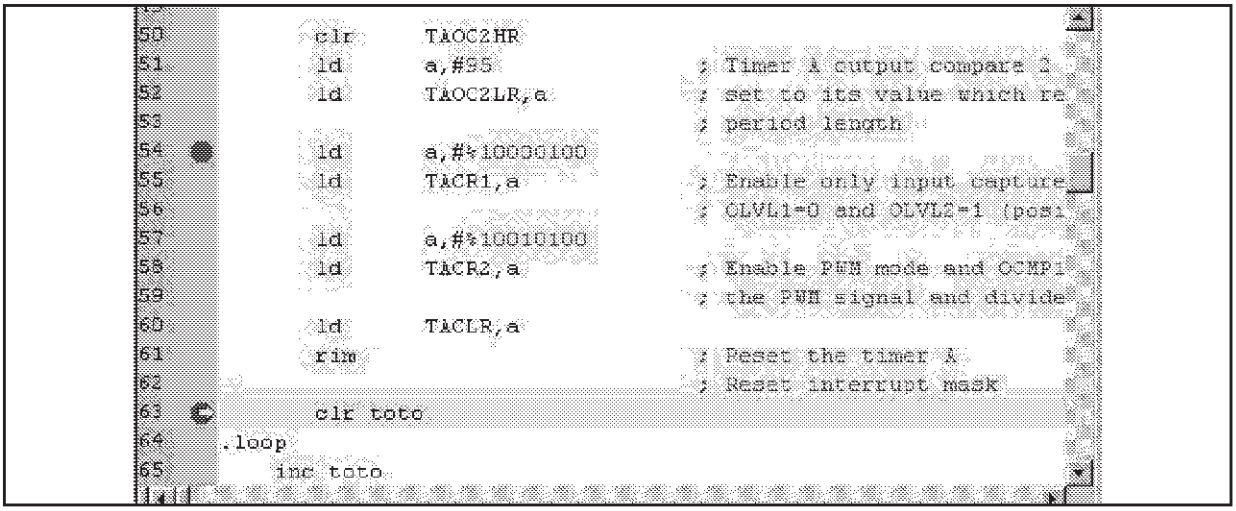
Figure 11. ST72XXX Peripheral Register Window



2.5 INSTRUCTION BREAKPOINTS

To insert a breakpoint, click on the left-hand side of the targeted line, or create the breakpoint using the Instruction Breakpoints window. A red panel is then displayed to the left of the line where the breakpoint has been inserted. The program execution will be stopped as soon as this instruction is reached, causing this instruction not to be executed. There are also 2 columns in the Instruction Breakpoints window: Counter (to stop after the PC reaches counter+1 times this instruction) and Condition (to stop counter+1 iteration after the condition is true, the condition has to be a C expression which returns a boolean value).

Figure 12. Instruction Breakpoint WIndow



The yellow line shows the instruction to be executed (position of the current Program Counter).

2.6 WATCH WINDOW

The Watch window displays the current value of selected variables/registers. The values are refreshed when the execution of the program is stopped and can be displayed in several formats (decimal, hexadecimal, binary...).

Several tabs (Watch 1, 2, 3, 4) are available in order to improve readability or to group variables in particular contexts.

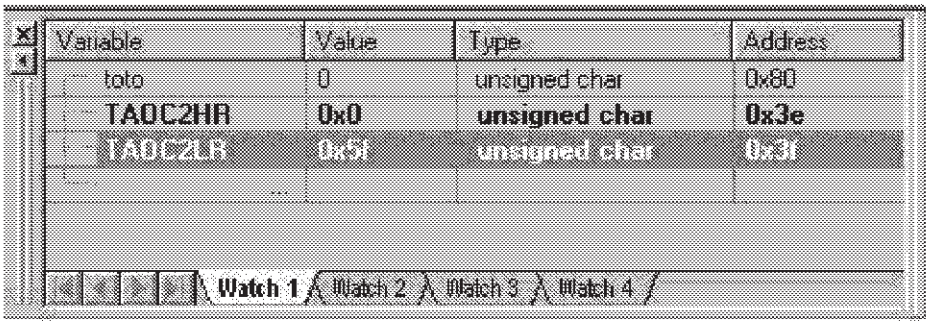
The name of the variable can be typed directly into the Variable column or the name can be dragged and dropped from the Editor window. An “Add to Watch” menu is also available by clicking on the variable using the right-hand mouse button.

The following values can be displayed in this Watch window:

- structure, bit field,
- array,
- pointer.

The displayed variables can be modified by double-clicking on their value and overtyping the new value.

Figure 13. Watch Window



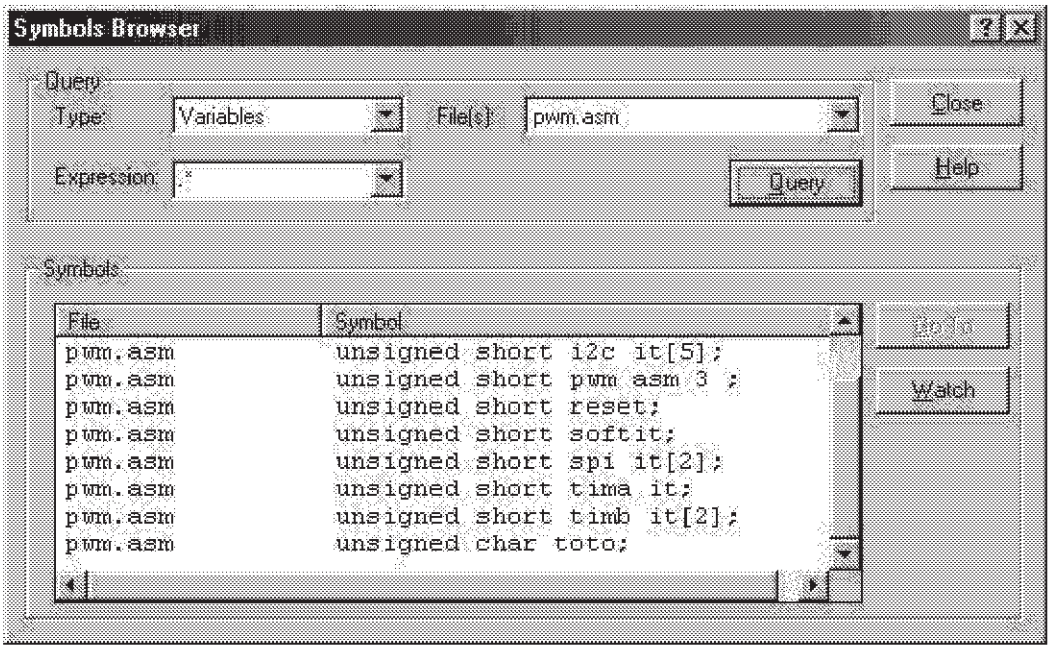
2.7 SYMBOL BROWSER WINDOW

This feature is used to find functions or data in all application files. A string (using the “Expression” box) can also be searched.

A “Go to” button is used to access the function or variable selected on the Found list in the corresponding source file.

A “Watch” button is used to copy the found symbol to the Watch window.

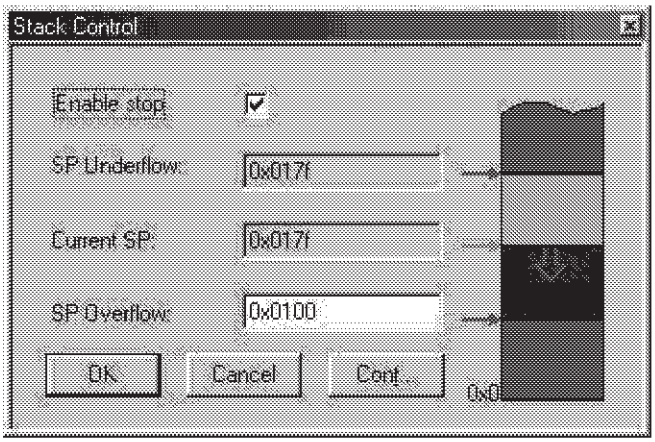
Figure 14. Symbol Browser Window



2.8 STACK CONTROL WINDOW

Stack address underflow and overflow values can be defined using this window. A command can also be generated to stop the application from running if the SP (Stack Pointer) value exceeds or falls beneath these address values.

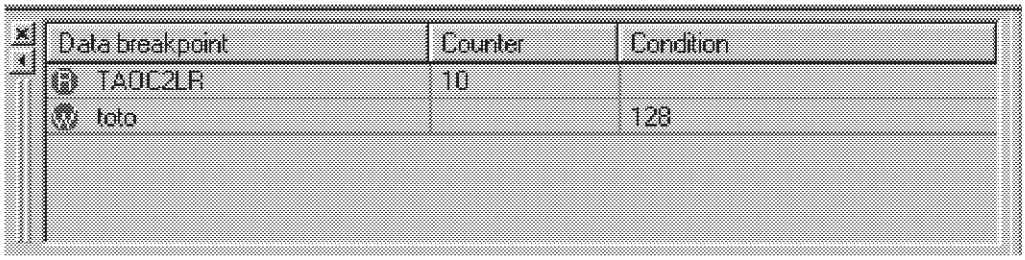
Figure 15. Stack Control Window



2.9 DATA BREAKPOINTS

Data breakpoints interrupt the program when a data variable is either read or written. To create a data breakpoint, right-click on the variable and select “Insert Read (or Write) Data Breakpoint”. Counter and Condition features are also included. The Counter value is used to stop the program after the counter has read- or write-accessed the variable. The Condition value is used to stop the program when the variable has reached the Condition value. In this case, the real time is lost.

Figure 16. Data Breakpoint Window



Example: In the above case, the program will be stopped when variable TAOC2LR is read-accessed 10 times or when variable toto reaches the value of 128.

2.10 HARDWARE EVENTS

This feature is used to filter the trace or to trigger output signals from the DVP/HDS emulators.

Hardware Events are different for the Emulator (HDS) and the Development Kits (DVP). (See Figure 17.) For the emulator (HDS), this feature is used to control a trigger output (OUT1 or OUT2). For the DVP2 (most recent Development Kits), there are 2 different types of Hardware Events: Trace Filtering mode or Trigger Output mode. See Table 1. These features are described in the following section, **Trigger/Trace Settings**.

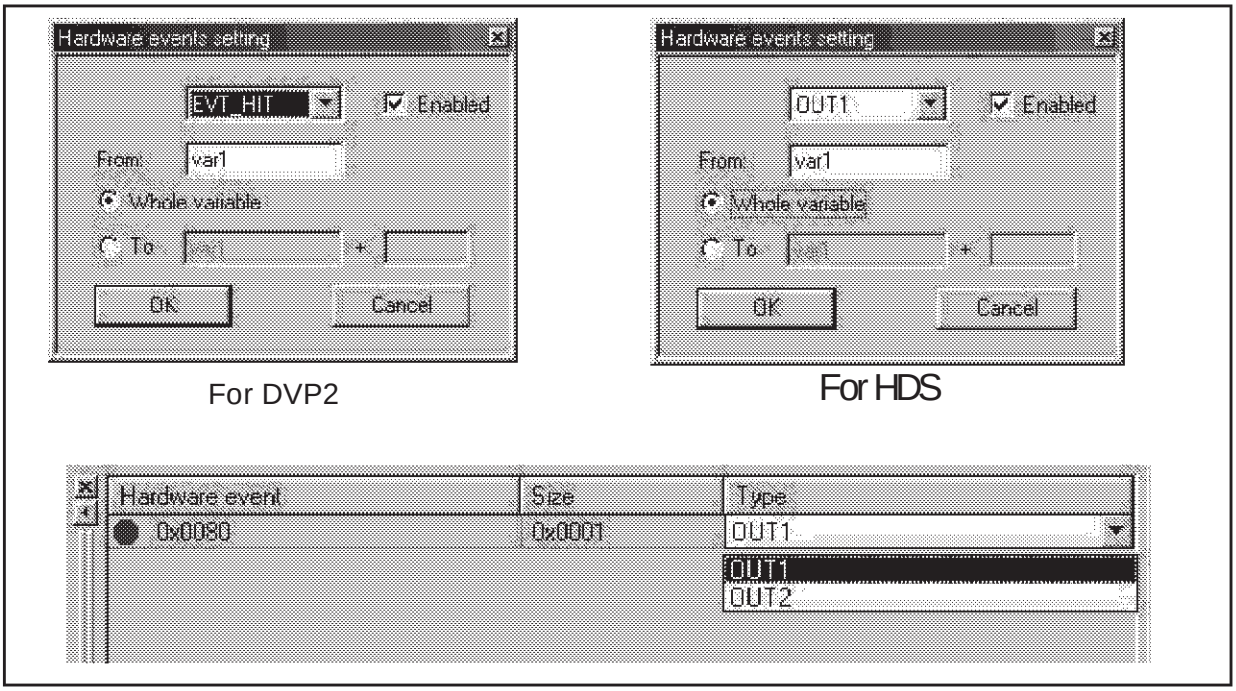
Table 1. Hardware Event Modes

	Trace Filtering mode	Trigger Output mode
EVENT ON	Address where the events begins	TRIGOUT signal rise from 0 to 1
EVENT OFF	Address where the events ends	TRIGOUT signal fall from 1 to 0
EVENT HIT	One specific address is accessed	A signal pulse (1 CPU cycle long) on TRIGOUT

For the older generation Development Kits (DVP), there are 2 different Hardware Events:

- FORCE_HIGH: Forces the output trigger signal to 1,
- FORCE_LOW: Forces the output trigger signal to 0.

Figure 17. Hardware Events Window



For the emulator (HDS), either one of the 2 available output triggers (OUT1 and OUT2) may be selected in the Hardware Events Setting window.

If “Whole variable” is selected, the trigger will be activated on the whole variable range. If “To” is selected, the end address of the memory range on which the event is set must be defined (symbolic name or address, the use of an offset is possible).

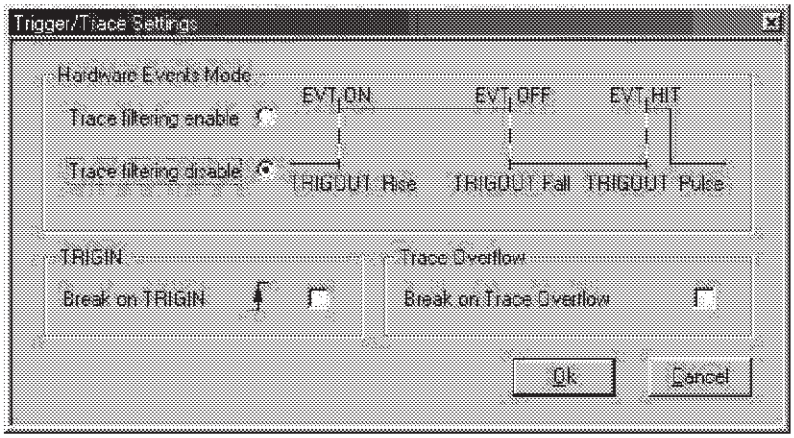
The size and type of the hardware event can be directly modified in the corresponding window (see Figure 17.), but not the address.

2.11 TRIGGER/TRACE SETTINGS (FOR DVP ONLY)

The following features are available using this window:

- Hardware Event mode (see Section 2.10).
- TRIGIN mode: causes a break in the program being executed when a rising-edge signal is received on the TRIGIN pin on the DVP board.
- Trace Overflow Break mode: stops the program as soon as the trace buffer is full (256 records).

Figure 18. Trigger/Trace Settings Window



2.12 TRACE WINDOW

The Trace window is used to view the contents of the trace buffer (physical memory module in the emulator/DVP that records hardware cycles that occur when a program is executed).

Step functions will not alter the information in the trace buffer. The trace buffer has a limited physical size (1 Kbytes on HDS2 emulators, 256 bytes on DVP2 emulators, none on DVP emulators).

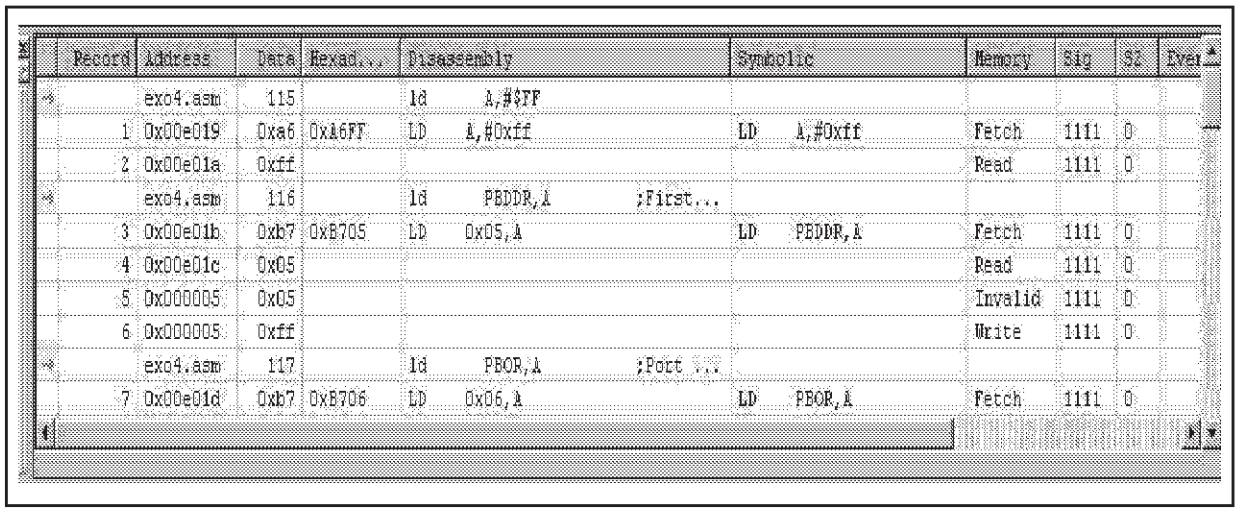
Specific information can be filtered in the trace buffer using the Logic Analyser.

Various information is displayed in this trace window:

- Record: trace record numbering starts at 1 up to 1024 (cycles)

- Address: memory location accessed
- Data: hexadecimal value on the bus
- Hexadecimal: the instruction in hexadecimal format
- Disassembly: the instruction in assembly language mnemonics
- Symbolic: the instruction in assembly language mnemonics with symbolic operands
- Memory: the type of memory accessed (R/W/invalid/Fetch)
- Sig: value of input signal (Analyser probe, front panel of the HDS emulator)
- Event: the name of the Logical Analyser event (HDS)
- TRIGIN: the value of the TRIGIN signal (DVP)
- Probe: the values of the signals output from the probe pins (DVP)

Figure 19. Trace Window



Record	Address	Data	Hexad...	Disassembly	Symbolic	Memory	Sig	S2	Even
→	exo4.asm	115		ld A,#$FF					
1	0x00e019	0xa6	0xA6FF	LD A,#0xff	LD A,#0xff	Fetch	1111	0	
2	0x00e01a	0xff				Read	1111	0	
→	exo4.asm	116		ld PBDDR,A ;First...					
3	0x00e01b	0xb7	0xB705	LD 0x05,A	LD PBDDR,A	Fetch	1111	0	
4	0x00e01c	0x05				Read	1111	0	
5	0x000005	0x05				Invalid	1111	0	
6	0x000005	0xff				Write	1111	0	
→	exo4.asm	117		ld PBOR,A ;Port ...					
7	0x00e01d	0xb7	0xB706	LD 0x06,A	LD PBOR,A	Fetch	1111	0	

2.13 LOGICAL ANALYSER

The Logical Analyser is used to create advanced breakpoints (conditional ones on complex events) or to filter traces (depending on the event).

3 events are available: Event 1, Event 2 and Event 3. An event corresponds to the flag of a specific address or symbol and/or a specific value on the data bus, and/or a specific binary signal value.

The Advanced Breakpoints mode is used to stop the application after one of the following events occurs:

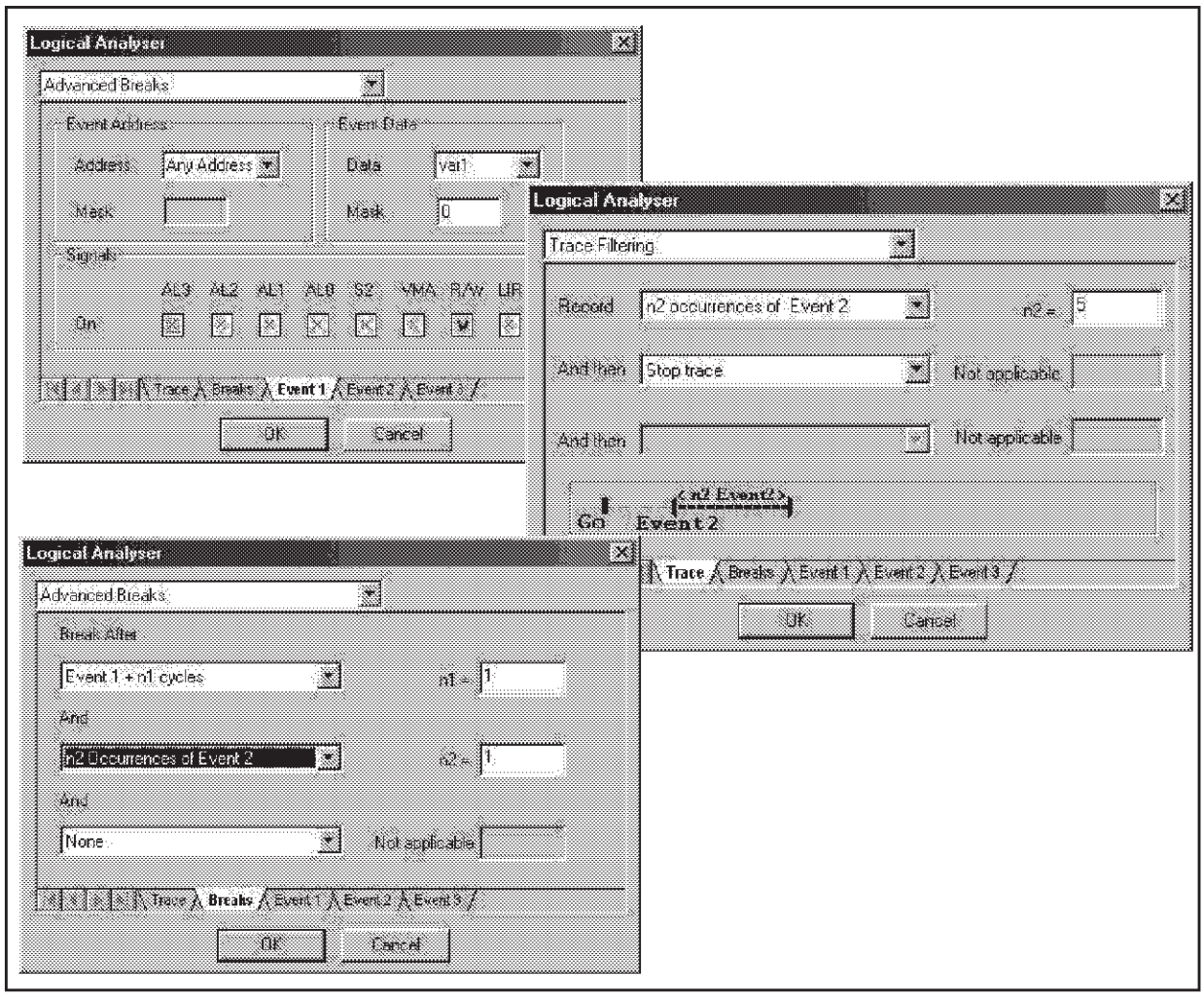
- Event1 + n1 cycles (Event 1 has to be defined and a value for n1 has to be given)
- n2 occurrences of Event 2 (Event 2has to be defined and a value for n2 has to be given)
- Both previous conditions can be linked with an AND operator.

Trace Filtering mode is used to record the following in the trace buffer:

- Events 1 and 3 cause trace buffer recording to be stopped after n1 or n3 cycles. All events up to Event 1 (or 3) +n1 (or n3) cycles can be recorded.
- Event 2 causes each access made to a specified area (associated to an event) to be recorded n2 times.
- A Permanent recording is also available.

This feature is used to trace complex events (links such as “And then” may be possible between events...).

Figure 20. Logic Analyser Windows



Not all the features have been described in this document. Refer to the STVD7 User Manual for more information about the ST7 Debugger or the ST7 Training Exercise Manual (available on the Free Software page on ST internet site: http://mcu.st.com) to practice.

3 TROUBLESHOOTING

A Troubleshooting section relating to each of the Hardware tools (Development Kits or Emulators) is given at the end of each of the corresponding user manuals.

Nevertheless, there is an additional anomaly that may occur on certain PCs (IBM Thinkpad laptops, for example):

If the message ”Connection error (LPTx): Communication error” appears when trying to connect with the hardware tool, first make sure that the parallel cable is correctly connected. Then, if this anomaly still occurs, enter the BIOS of the PC and change the I/O parallel port address settings: address 378 must be specified (uncheck the automatic settings box and choose user settings, basic 0001 configuration). When the new parameters are set, reboot the PC, in compliance with the PC operating system, so that the modifications are taken into account.