ST7MDTS1-EMU2B
HDS2 Series Emulator
User Manual

Release 1.0

August 2001



Ref: DOC-ST7MDTS1-EMU2B



INSTRUCTIONS FOR USE—WARNING

This product is conform to the 89/336/EEC Directive. It complies with the ITE EN55022 standard for EMC emissions and generic 50082-1 (1992 edition) immunity standards.

This product is an FCC Class-A apparatus. In a residential environment, it may cause radioelectrical disturbances.

In addition, some parts of this emulator are not contained in an outer casing; consequently, it cannot be immune against electrostatic discharges (ESD). It should therefore be handled only in static safe working areas. Please refer to *Appendix A: EMC Conformity and Safety Requirements* on page 53 for relevant safety information.

USE IN LIFE SUPPORT DEVICES OR SYSTEMS MUST BE EXPRESSLY AUTHORIZED.

STMicroelectronics PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF STMicroelectronics. As used herein:

1. Life support devices or systems are those which (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided with the product, can be reasonably expected to result in significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can reasonably be expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Table of Contents

Chapter 1: Introduction 5

1.1 Emulator configuration 7
1.2 Emulator operation 8
1.3 Software and documentation for the emulator kit 8
1.4 About this manual.... 9
1.5 Getting assistance 9

Chapter 2: Getting Started 11

2.1 Your system requirements 11
2.2 Delivery checklist 11
2.3 Installing the hardware 13
2.4 Debuggers supporting the ST7 HDS2 emulator 18

Chapter 3: STVD7 19

3.1 Installing STVD7 19
3.2 Launching STVD7 20
3.3 About STVD7 debugging features 21
3.4 Workspaces 22
3.5 Toolchains and application files 23
3.6 Creating a workspace 26
3.7 Opening an existing workspace 28
3.8 Opening binary files 30
3.9 Opening lone programmable files (*.s19 or *.hex) 31
3.10 Changing your project settings 32
3.11 Saving workspaces 34
3.12 Debug context and Build context 36
3.13 Configuring the MCU 37
3.14 Start debugging! 42

Chapter 4: Emulator Features 43

4.1 Specific features 43
4.2 Emulator architecture 43
4.3 Output triggers 47
4.4 Analyser probe input signals 49
4.5 Front panel LEDs 50
4.6 On-chip peripherals 51
4.7 Emulation functional limitations and discrepancies 52

Table of Contents

Appendix A: EMC Conformity and Safety Requirements 53

Appendix B: Troubleshooting 55

B.1 Identifying the problem 55
B.2 Changing the parallel port setup on your PC 55
B.3 Running the hardware test 56

Appendix C: Glossary 59

Product Support 61

Getting prepared before you call..... 61
Contact list 61
Software updates..... 62
Hardware spare parts 62

Index 65

1 INTRODUCTION

Thanks for choosing ST7! This manual will help you get started with the ST7MDTS1-EMU2B.

The ST7MDTS1-EMU2B package is a development tool designed for emulation of the following microcontrollers of the ST72SCR family:

Supported Devices

ST72SCR1R4T1
ST72FSCR1R4T1
ST72SCR1F4M1
ST72FSCR1F4M1

The ST7MDTS1-EMU2B package will assist you in debugging your application hardware as well as your software. The ST7MDTS1-EMU2B kit comes with the ST7 Visual Debug software package which contains all of the necessary resources to help you design, develop and debug ST7 application software running in a real environment.



Tip: *If you come across any terms or abbreviations you do not understand, you can check their meaning in the Appendix C: Glossary on page 59.*

Note: *Please refer to Section 4.7: Emulation functional limitations and discrepancies on page 52 for description of the limitations on the current release of this emulator kit.*

First off, check that the ST7 MCU that you have picked for your application is in the list of devices (see table above) supported by this version of the ST7MDTS1-EMU2B emulator.

The emulator package is made up of two main parts:

- The Hardware Development System (**ST7-HDS2**), which is the common mainframe to all ST7 emulators.
- The **ST7MDTS1-Active Probe**, dedicated to the family, which constitutes the physical link between the emulator and your application.

Note: *When receiving the ST7MDTS1-EMU2B development tool, please refer to the Delivery checklist on page 11 to confirm that all of the contents of the package are present.*

The emulator performs two main functions:

- It replaces the microcontroller in the application, by means of an emulation probe that is plugged into the application in place of the emulated MCU.
- It controls the internal data bus of the emulated microcontroller, providing arbitration and tracing capabilities on all accesses to either of the following resources:
  - ST7-HDS2 resource,
  - ST7MDTS1-Active Probe resources,
  - Application resources.

Therefore, you can have the emulator running your software in the application as the emulated microcontroller unit (MCU) would do, associated with extensive tracing capabilities (keeping a trace of what the MCU did) and control capabilities (ability to react specifically upon defined conditions).

In this way, it is possible to obtain a full emulation of the microcontroller resources.

1.1 Emulator configuration

Figure 1 shows a general configuration for the ST7MDTS1-EMU2B emulator kit. The main ST7-HDS2 box is connected to your PC via the parallel port. Two flat cables connect the ST7-HDS2 box to the ST7MDTS1-Active Probe, to which a device adapter can be fixed via an extender or flex cable so that you can connect the emulator to your application board.

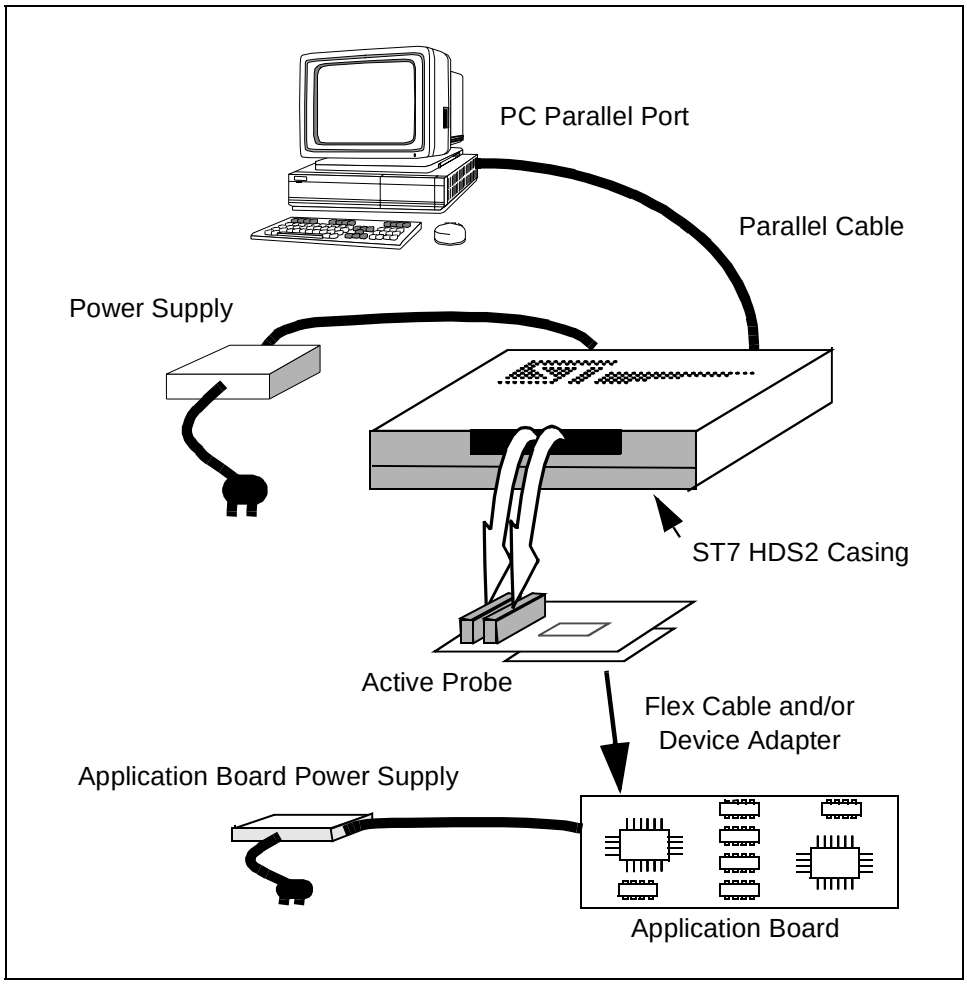


Figure 1: ST7MDTS1-EMU2B general configuration

1.2 Emulator operation

A symbolic debugger, **ST7 Visual Debug**, (also referred to as **STVD7**), is provided to control the emulator.

ST7 Visual Debug can be run on a PC under the Windows environment, and is common to all ST7 devices. ST7 Visual Debug uses a window menu-driven interface, and enables you to configure the emulator.

Chapter 3: STVD7 on page 19, explains how to install ST7 Visual Debug on your PC, and set up the emulator configuration so that you can begin your debugging session.

Once assembled and linked, the application software is ready to be downloaded into the ST7 emulator. The development station performs a real-time emulation of the target device, thus allowing high performance testing and debugging of both application hardware and software.

When the program is fully debugged, the ST7 EPROM programming board (ref.: ST7MDTS1-EPB — not provided with this emulator kit) can be used to program the emulation device with the Motorola S Record format file produced by the OBSEND formatter.

1.3 Software and documentation for the emulator kit

The “MCU on CD” CD-ROM contains:

- ST7 Tools, comprising the following software:
  - The source-level graphic debugger, STVD7, that operates with ST7-HDS2 Emulators and ST7 Development Kits or as a standalone ST7 simulator.
  - The ST7 Assembly chain, composed of an assembler, linker, librarian and formatter.
  - The ST7 Windows Epromer to program your MCU target devices.
- Third-party C compiler and toolchain demos (Hiware and Cosmic).
- ST7 application notes (with sources), training slides and exercises, and other useful reference documents in PDF format, such as:
  - Datasheets for the ST7 MCU family
  - *ST7 Programming Manual*
  - *ST7 Assembler-Linker User Manual*
  - *STVD7 User Manual*

1.4 About this manual....

Detailed instructions on how to install your emulator configuration is described in *Chapter 2: Getting Started* on page 11.

How to start debugging your application using your emulator and STVD7 is described in *Chapter 3: STVD7* on page 19.

The emulator kit's hardware features are described in *Chapter 4: Emulator Features* on page 43.

The following conventions are used in this manual:

Bold text highlights key terms, phrases and is used when referring to names of dialog boxes, windows and tabs within windows.

Bold italic text denotes menu commands (or sequence of commands), options, buttons or checkboxes which you must click in order to perform an action.

Italicized text highlights document names, variable strings, column names and field names.

`Code font` designates file names, programming commands, path names and any text you must type.

The **>** symbol is used in a sequence of commands to mean "then". For example, to open an application in Windows, we would write: "Click ***Start>Programs>ST7 Tool Chain***>....".

1.5 Getting assistance

For more information, application notes, FAQs and software updates on all the ST microcontroller families, check out the CD-ROM or our website:

http://mcu.st.com

For assistance on all ST microcontroller subjects, or if you need help with using your emulator, use the contact list provided in *Contact list* on page 61. We'll be glad to help you!

2 GETTING STARTED

2.1 Your system requirements

The ST7MDTS1-EMU2B HDS2 Emulator (both hardware and software components) has been designed to work with PCs meeting the following requirements:

- One of the following operating systems: Microsoft® Windows® 95, 98 or NT®.
- Intel® Pentium (or compatible) processor with minimum speed of 100 MHz.
- Minimum RAM of 32 MB.
- 21 MB of free hard disk space to install all of the ST7 tools.

2.2 Delivery checklist

The emulator unit, (ref.: ST7MDTS1-EMU2B), is delivered with the following (refer to *Figure 2*):

1 One emulator box containing the ST7-HDS2 main board (ref.: MB176) with two 50-wire flat cables to connect the board to the emulation probe.

2 One emulator power supply and cable.

3 Three EMC suppress ferrites.

4 One generic emulation probe—the ST7 Generic Probe (ref.: DB467).

5 One rainbow-colored logic probe cable for the logical analyser.

6 Two SMB to BNC cables for use with the output triggers.

7 An ST7MDTS1-Target Emulation Board (TEB) (ref.: DB484).

8 One parallel cable.

9 A 20 cm TQFP64 flex cable with two 40-pin connectors (for connection between the emulation probe and your application board) for the TQFP64 package (ref.: DB513A). Used with (*10*).

10 A TQFP64-0.8 socket (ref.: TET 064SA) and cover, for soldering to your application board).

11 A 20 cm SDIP32 flex cable with one 40-pin connector (for connection between the emulation probe and your application board) for the SDIP32 package (ref.: DB510B). Can be combined with (*12*).

12 A SDIP32/SO24 adapter (ref.: DB514A) comprising two parts that fit together. The topmost part fits onto the SDIP32 flex cable (*11*) while the bottom-most part can be removed to be easily soldered to your application board.

13 This manual. (Not shown.)

14 The “MCU on CD” CD-ROM containing ST7 information and software, including ST7 Visual Debug. (Not shown.)

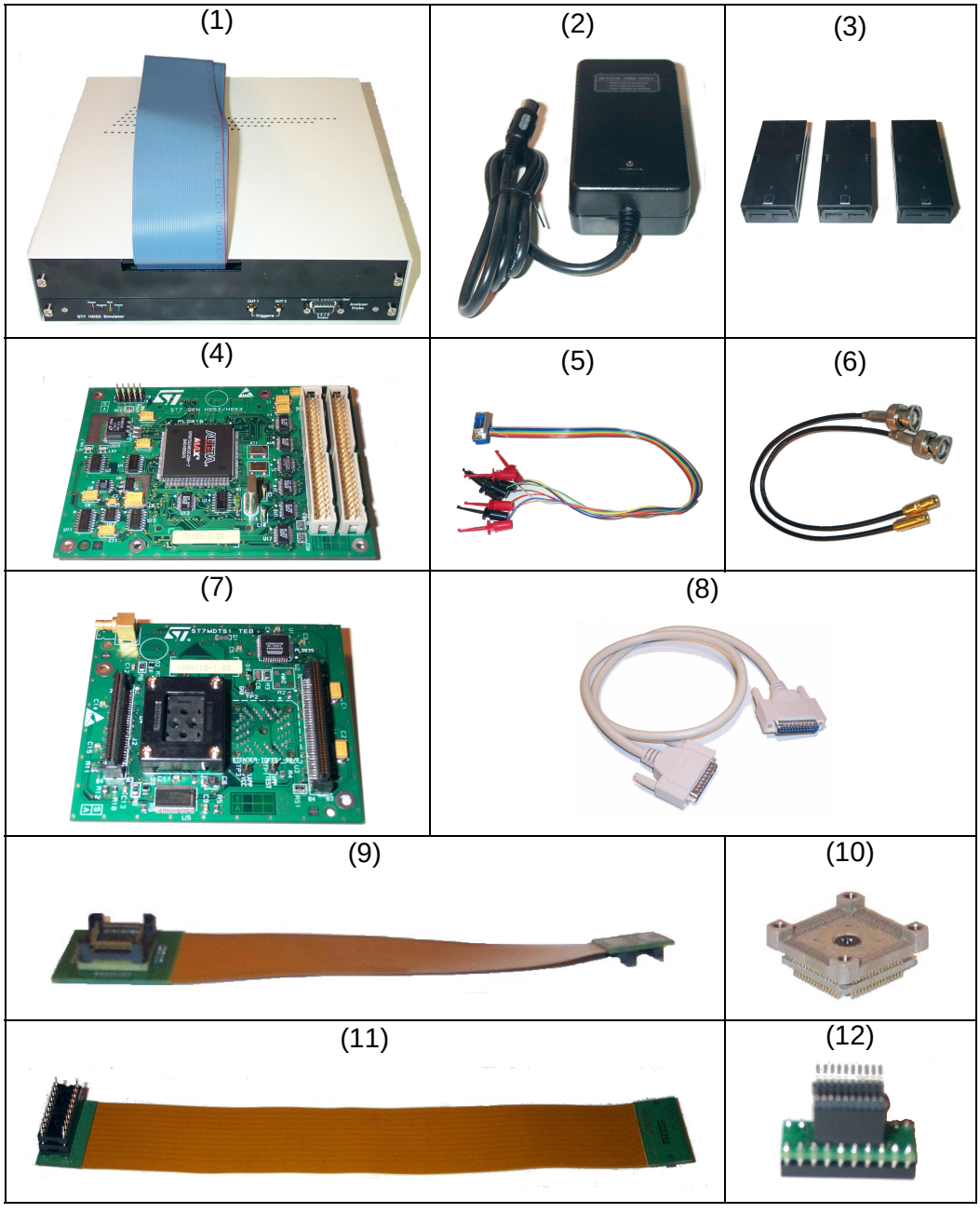


Figure 2: Main components of ST7MDTS1-EMU2B emulator kit

2.3 Installing the hardware

The ST7-HDS2 emulator is connected through the parallel port to a PC computer which runs the control software (ST7 Visual Debug) as explained later. To connect your ST7 HDS2 emulator, you will have to follow these general steps:

1 Connect the ST7-HDS2 to your PC using the parallel cable provided.

2 Connect the two flat cables of your ST7-HDS2 emulator to the emulation probe connectors.

3 Connect the emulation probe to your application board using the appropriate flex cable and/or socket adapter.

4 Connect the power supply cable between the power supply block and the power connector located on the rear panel of your ST7-HDS2 emulator.

5 Power up the emulator and then connect your application board power supply.

A connection flow diagram is shown in *Figure 3*. Step details are discussed in the following sections.

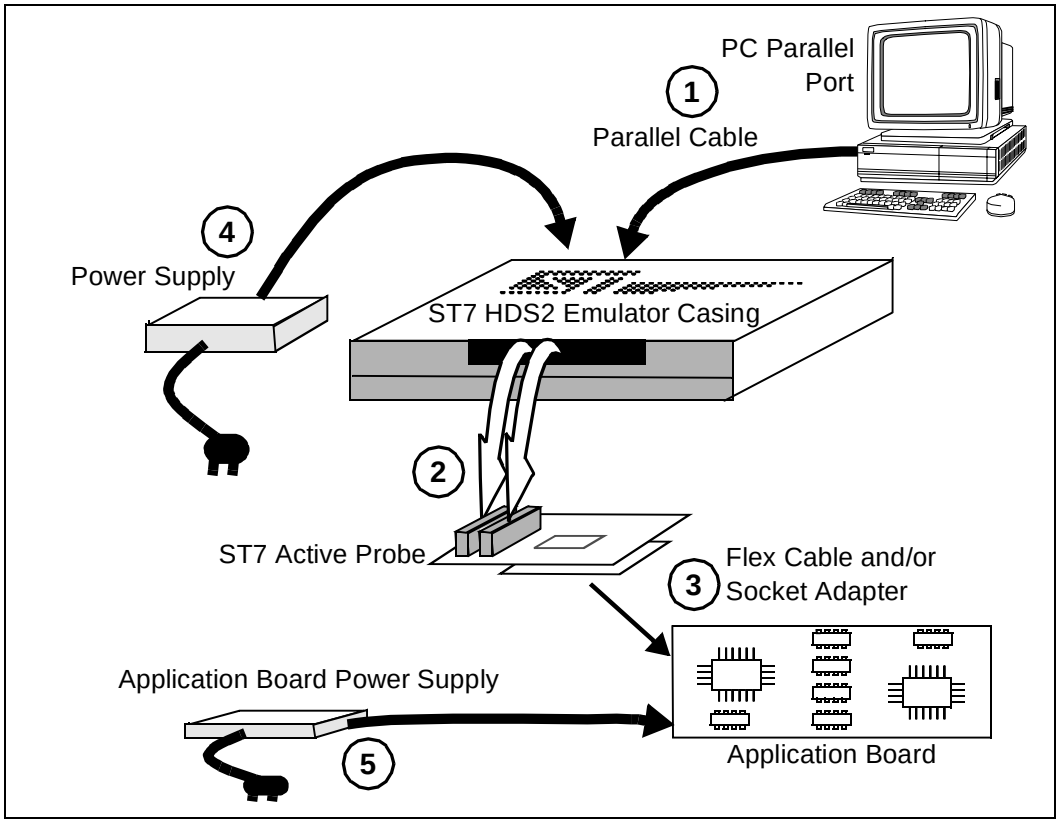


Figure 3: Connection flow

2.3.1 Step 1: Connecting the emulator to your PC

1 Shutdown and power-off the PC that is to be connected to the emulator.

2 Connect one end of the parallel cable to the emulator's rear panel 25-pin SUB-D connector and the other end to one of the PC's parallel ports (LPT1 to LPT2)—refer to *Figure 4*.

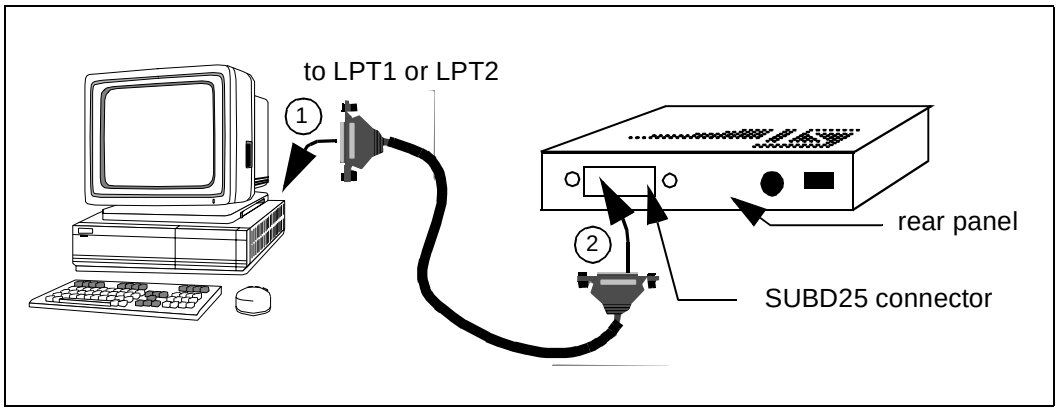


Figure 4: Connecting the emulator to the PC

2.3.2 Step 2: Connecting the probe

1 Ensure that the application and the emulator are **powered-off**.

2 Plug the two 50-wire flat cables into the J1 and J2 connections on the probe, as described below (refer also to *Figure 5*):

ST7 HDS2 emulator (ref.: MB176)	ST7 generic probe (ref.: DB467)
Upper cable	J2
Lower cable	J1

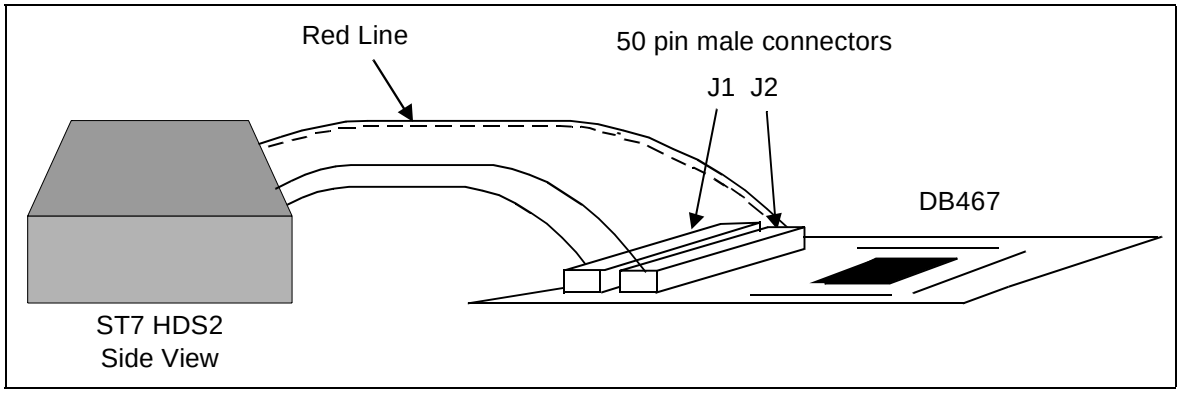


Figure 5: Connecting the emulator to the generic probe

3 **EMC-Compliant Probes:** In order to work under an EMC-compliant environment, you will have to clip two or three EMC-ferrites on the 50-wire flat cables linking the probe to the emulator box. Place these ferrites as close to the emulator window as possible. Three ferrites are provided in the package. See *Figure 6* for an illustration of where to attach the ferrites.

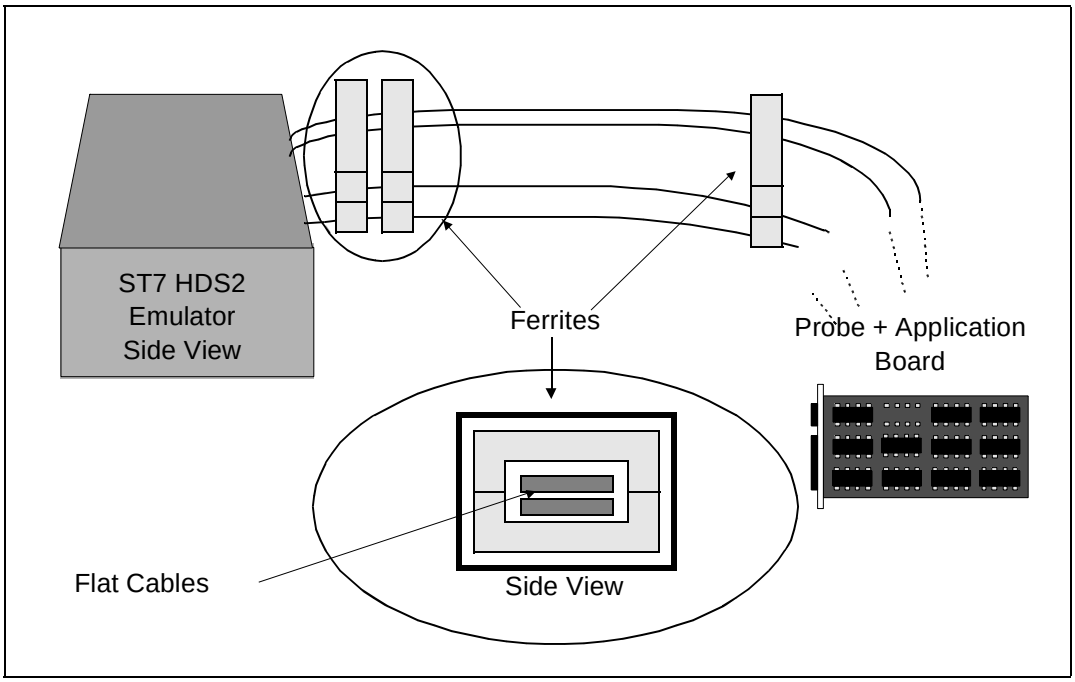


Figure 6: Making your emulation probe EMC-compliant

2.3.3 Step 3: Connecting the probe to your application board

Emulated devices of the ST7MDTS1 family are available in two packages, each package has its own connection procedure, found on the page cited in the following table:

Packages	Devices	Page No.
TQFP64	ST72SCR 1R4T1 ST72FSCR 1R4T1	16
SO24	ST72SCR 1F4M1 ST72FSCR 1F4M1	17

If you are using the TQFP64 package, proceed as follows (see *Figure 7*):

To connect the ST7MDTS1-Active Probe to your application board, a TQFP64 flex cable and a TQFP64-0.8 socket base have been provided.

1 Solder the TQFP64-0.8 socket base (ref.: TET 064SA) onto your application board. Do not use the socket cover (set it aside for future use with an actual MCU).

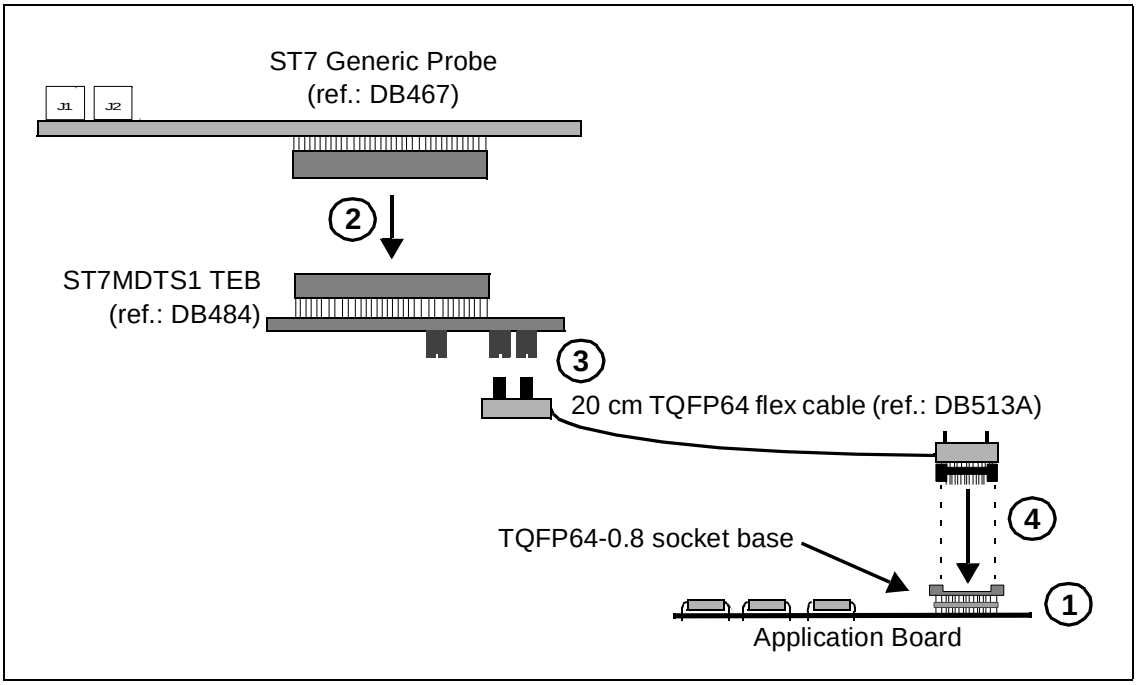


Figure 7: TQFP64 MCU package connections

2 Connect the ST7 Generic Probe (ref.: DB467) to the ST7MDTS1 TEB (ref.: DB484).

3 Connect the two 40-pin connectors of the 20 cm TQFP64 flex cable (ref.: DB513A)) to the two 40-pin connectors (labelled QFP64 FLEX ADAPTER) on the underside of the ST7MDTS1 TEB (ref.: DB484), making sure to align the pin1's of the flex cable connectors with the pin 1's of the TEB connectors.

4 Connect the TQFP64 connector on the end of the flex cable to the TQFP64 socket base, making sure to align pin 1 on the TQFP64 flex connector with pin 1 on the socket base. (Pin 1 is indicated by a chamfer on the TQFP flex connector and by a little arrow or chamfer on the socket base.)

5 Now use the four screws supplied to fasten this entire assembly together through the holes located on the TQFP flex cable connector as indicated in *Figure 7*.

If you are using the SO24 packages, proceed as follows (see *Figure 8*):

To connect the ST7MDTS1-Active Probe to your application board, a SDIP32 flex cable and a SDIP32/SO24 adapter have been provided. The SDIP32/SO24 adapter (ref.: DB514A) is comprised of two parts that fit together. The topmost part fits onto the SDIP32 flex cable (*11*) while the bottom-most part can be removed to be easily soldered to your application board.

1 Solder the bottom part of the SDIP32/SO24 adapter (ref.: DB514A) onto your application board.

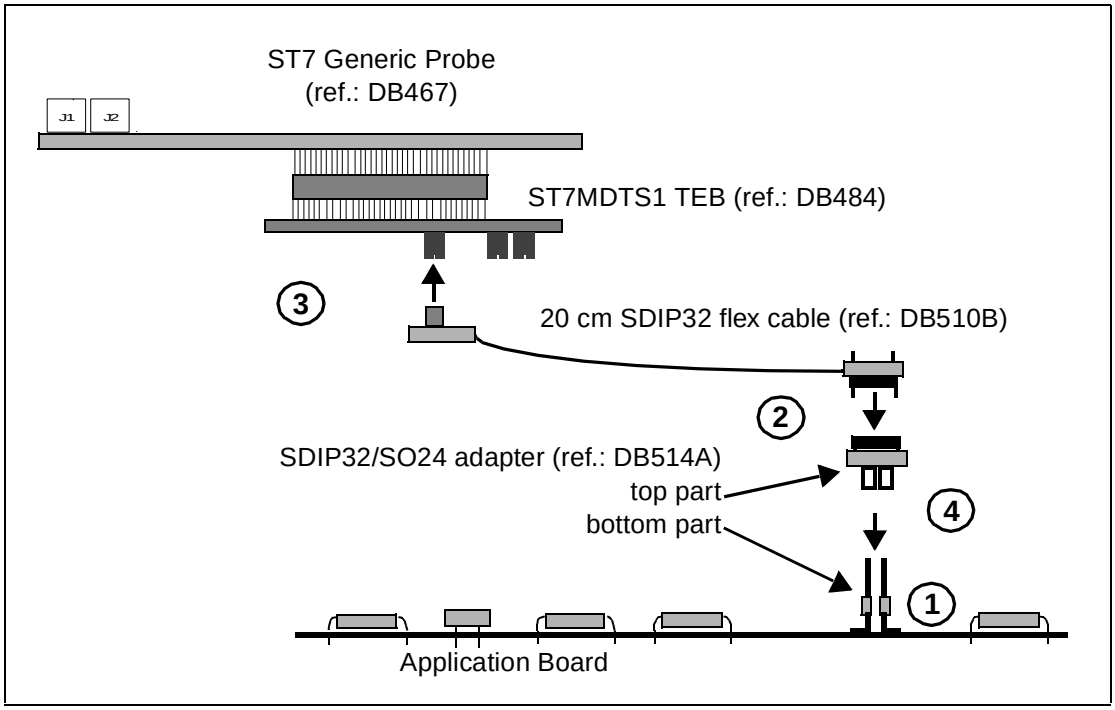


Figure 8: SO24 MCU package connections

2 Connect the top part of the SDIP32/SO24 adapter (ref.: DB514A) to the SDIP32 flex cable (ref.: DB510B).

3 Connect the single 40-pin connector on the SDIP32 flex cable (ref.: DB510B) to the single 40-pin connector (labelled SO24 FLEX ADAPTER) on the ST7MDTS1-TEB (ref.: DB484).

4 Connect the top and bottom parts of the to the SDIP32/SO24 adapter (ref.: DB514A).

2.3.4 Step 4: Connecting the power supply

Caution: *Make sure that both the ST7MDTS1-EMU2B and the application board are powered OFF before making any connections.*

1. Connect the external power supply provided with the emulator to the rear panel of the mainframe using a 5-pin DIN connector.
2. Plug the power supply into the mains using the supply cable provided.

Mains Voltage Specifications

AC Voltage	100 V to 240 V
Frequency	50 Hz to 60 Hz

2.3.5 Step 5: Powering up

1. Check the ST7-HDS2 operating voltage (110 V/220 V), indicated on the label on the power supply. Contact your dealer if this voltage does not correspond to your mains' power supply.
2. Power up the ST7 HDS2 emulator from the ON/OFF switch located on the rear panel. The LED labelled ***Power*** on the front panel should then light up.
3. Power up your application board.



Warning: Always power on your ST7 HDS2 emulator first, then power up your application board. When powering off, always power off your application board first, then power off your emulator.
NEVER have your application board under power while the emulator is powered off—this will cause serious damage to the emulator.

2.4 Debuggers supporting the ST7 HDS2 emulator

ST7 Visual Debug (also known as STVD7) is the STMicroelectronics debugger currently supporting the ST7 HDS2 emulator.

Third party debuggers supporting the ST7 HDS2 emulator include the following:

- ST7-Hiwave by HIWARE
- Zap by COSMIC
- Hitop by HITEX.

ST7 Visual Debug is free software. It is available on the STMicroelectronics website. See *Product Support* on page 61 for more information.

3 STVD7

STVD7 is an integrated development environment that allows you to edit, debug and rebuild your application all from within STVD7.

The following sections tell you:

- *Section 3.1*—how to install the STVD7 software,
- *Section 3.2*—how to launch STVD7,
- *Section 3.3*—a little about STVD7's debugging features,
- *Section 3.4*—what a workspace is,
- *Section 3.5*—what toolchains and executable files are supported by STVD7,
- *Section 3.6*—how to create a STVD7 workspace,
- *Section 3.7*—how to open existing workspaces,
- *Section 3.8*—how to open binary files,
- *Section 3.10*—how to change your project settings,
- *Section 3.11*—how to save workspaces,
- *Section 3.12*—how to switch from the build context to the debug context,
- *Section 3.13*—how to configure the target MCU in order to debug more accurately and efficiently.

3.1 Installing STVD7

Your emulator comes with the "MCU on CD" CD-ROM which contains a number of ST7 software tools. These tools run under the Windows® 95, 98, 2000 and Windows® NT® operating systems.

*Note: To install the software on "*MCU on CD*", Windows® 2000 and NT® users must have administrator privileges.*

To install and setup the ST7 software tools, follow these steps:

1 Close all other open applications on your Windows desktop.

2 Insert the "MCU on CD" into your CD-ROM drive. The CD-ROM's autorun feature will open up a welcome screen on your PC. If the autorun feature does not work, use Windows® Explorer to browse to the CD-ROM' s root folder, and double-click on `Welcome.exe`.

3 Select ***Install Your Development Tools*** from the list of options. A new screen will appear listing the different families of STMicroelectronics MCUs.

4 Use your mouse to place the cursor over the ***ST7 Tools*** option. Choose ***ST Tools***, then ***ST7 Toolchain*** from the lists that appear.

5 The install wizard will be launched. Follow the instructions that appear on the screen.

You can choose to install the complete toolchain (i.e. the appropriate version of STVD7, the Windows Epromer and the Assembler-Linker) for each type of development tool (development kit, HDS2 or EMU3 emulators or simulator), or perform a customized installation.

If you choose a customized installation, you can choose to install any or all of the STVD7 versions, and/or the Windows Epromer and/or the Assembler-Linker. **As a minimum, in order to emulate your application with your emulator, you must install STVD7 for HDS2.**

If you also install the ST7 Assembly Toolchain, you will be able to use the ST7 Assembly Toolchain as part of STVD7's integrated development environment.

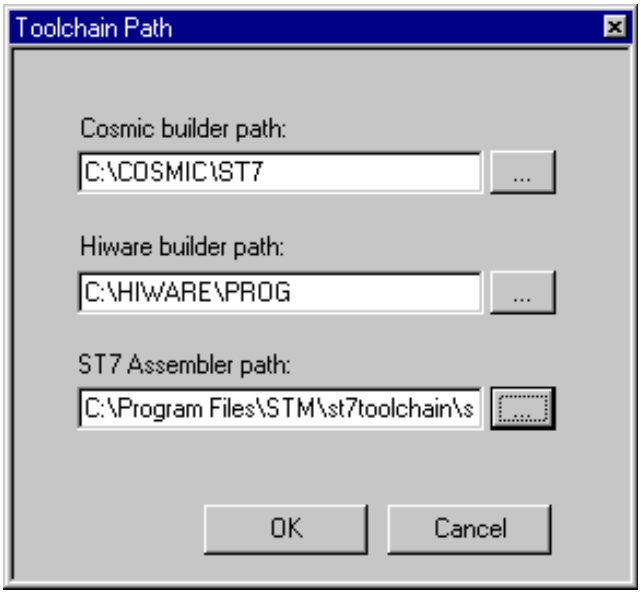
The installation is now complete. You will be prompted to reboot your computer. You should do so before launching STVD7.

3.2 Launching STVD7

1 From your Windows desktop, select ***Start>Programs>ST7 Tool Chain>Development Tools>STVD7 HDS2 emulator.***

2 The first time you open a version of STVD7 you will be prompted to enter the toolchain paths to be used by STVD7's integrated development environment.

Enter the paths for the toolchains that you use (i.e. any or all of the Hiware, Cosmic or ST7 ASM toolchains) and click ***OK***. (The default paths for each toolchain are shown below.)



3 If you choose ***Cancel,*** you will be prompted again to enter the toolchain paths the next time you launch STVD7.

*Note: You may modify the toolchain path at any time from within STVD7—simply select **Project>Toolchain Paths** from the main menu to access the dialog box above.*

3.3 About STVD7 debugging features

A number of advanced features are included in the STVD7 software:

- **Data Breakpoints** on the occurrence of a memory access via a read operation or a write operation, or both.
- **Instruction Breakpoints** on the occurrence of an opcode fetch.
- A **Logical Analyser** that allows you to control either the recording of the trace buffer, or a break in the execution of the application using a series of specific conditions (events).
- A **Trace window** to view the contents of the **trace buffer**, which permanently records in real time on 32-bits:
  - Address and data bus information.
  - Flag status and 4 external signal values.

  You can record up to 1024 executed cycles. Using trace filtering, you can filter out only those cycles you wish to record in the trace buffer. You can equally control which of the recorded cycles are displayed in the Trace window using line filtering. Addresses, data, control/status bits and 4 user signals are displayed using mnemonic and user symbols.
- Internal synchronization signals can be output to either of two **Trigger Outlets** on the front panel of the emulator (OUT1 or OUT2). This feature enables you to count events using an external equipment, when optimizing software for example, or to synchronize an oscilloscope when debugging hardware.
- **Hardware Events** can be used to control the sending of signals to the trigger outputs.
- You can choose the output that you wish the signal to be sent to (i.e. either OUT1 or OUT2).
- A **Hardware Test** function that allows you to perform a number of hardware tests on the Development Board, at your choosing. Refer to *Running the Hardware Test* for more information.
- A powerful **online help** facility can be invoked at any time to give additional information about the commands, the processor or the emulator kit.

3.4 Workspaces

STVD7 organizes project development and debugging into workspaces. Workspaces allow you to store application and project settings and save them as a `*.wsp` file, so that each time you wish to work on the project, you will find all of the settings exactly as you left them.

Creating a workspace is the first thing that you need to do when using STVD7 for the first time or when starting any new project. You must have an open workspace to work with STVD7. How to create a new workspace is described in detail in *Section 3.6* on page 26. Sample workspaces for each supported toolchain are provided so that you can familiarize yourself with STVD7 (for a listing of sample workspaces, see *Table 1* on page 24).

Each workspace is comprised of three information sets: the project settings, the visual environment and the debugging context.

- The **project settings** consists of the information necessary for a successful build of an application (commands to run, makefile file etc....). Your workspace's project settings include the definition of your application toolchain (see *Section 3.5* on page 23).
- The **visual environment** consists of the open windows elements along with their current layout, bookmarks and other features. The visual environment is composed of two environments, one in the **Build context** and one in the **Debug context** (see *Section 3.12* on page 36).
- The **debugging information** includes information on breakpoints, memory mapping, advanced breakpoints programs, trace etc..

3.5 Toolchains and application files

A quick summary of development toolchains and application file types supported by STVD7 will help you in setting up your workspace.

Three different development toolchains are currently supported by the STVD7. Each type of toolchain has its own application and executable file types, project environment and building tools (i.e. linkers and convertors):

- The **ST7 macroassembler toolchain** from STMicroelectronics, which generates either `.s19` or `.hex` executable files with various intermediate files, such as `.map` or `.lst` files.
- The **Hiware C or Assembler toolchain**, which generates `.abs` executable files with various intermediate files, such as `.o` or `.dbg` files.
- The **Cosmic C or Assembler toolchain** which generates `.elf` executable files with various intermediate files, such as `.o` or `.st7` files.

When you set up a workspace, you will need to define the following project settings:

- **The toolchain to be used—**Hiware, Cosmic or ST7 macroassembler.
- **The executable file** (`*.abs`, `*.elf`, `*.s19` or `*.hex` depending on toolchain—refer to *Table 2* on page 25).
- **The maker program** for the toolchain. The maker program can be a part of the toolchain software (such as Hiware's `maker.exe`) or you can choose to use a generic maker such as `Nmake.exe` or `Gmake.exe` (which is provided with the STVD7).
- **The maker batch file** (`*.mak` or `*.bat`). This is a file which you create for each application which spawns the compilation and/or link step each time you wish to build or rebuild. In it, you define the conditions for recompiling, re-linking or both.
  Default `*.mak` or `*.bat` files are often included with the toolchains—for example, `maker.mak` is included with the Hiware toolchain and simply recompiles your application if it detects that the file has been saved since the start of your debugging session. The STVD7 software includes sample `*.mak` and/or `*.bat` files for each toolchain—these are listed in *Table 1*.

Table 1: Sample files included with STVD7

Toolchain	Sample Workspace (with default path	Sample Make and/or Batch files (with default path[1])	Description of Make/Batch File
ST Macro assembler	`.../realtim/realtim.wsp`	`.../realtim/tim_rtc.bat`	Batch file that forces a recompile of the application.
	`.../spim11/spim11.wsp`	`.../spim11/spim11.bat`	Batch file that forces a recompile of the application.
Cosmic	`.../c/cosmic/sample.wsp`	`.../c/cosmic/sample.mak`	Recompiles only if one (or more) of the application files has been resaved.
		`.../c/cosmic/sample.bat`	Batch file that forces a recompile of the application.
Hiware	`.../c/hiware/sample.wsp`	`.../c/hiware/build.mak`	Recompiles only if one (or more) of the application files has been resaved.
		`.../c/hiware/rebuild.mak`	Forces a recompile of the application.

1) The full default path is: `C:/Program Files/Stm/st7toolchain/stvd7/hds/sample/...`

3.5.1 About executable files

The user should verify that the options to include debug information were active during creation of the project files. *Table 2* on page 25 summarizes the way each toolchain functions and lists the different file types (source files, intermediate files and executable files) used and produced by the toolchain. The **executable file types** and **intermediate file types** necessary to exploit fully the STVD7 capabilities are listed.

Table 2: Toolchain steps and their output files

Toolchain:	ST Macroassembler	Hiware	Cosmic
Compile or Assemble Step:			
Source File Types	`.asm`	`.c, .asm`	`.c, .s`
Required Options	`asm -li macrost7.asm`	<NONE>	`+debug`
Resulting File Types	`.obj, .lst`	`.o, .dbg`	`.o`
Linker Step:			
Required Options	`lyn macrost7.obj, macrost7` `asm macrost7.asm -sym -fi=macrost7.map`	<NONE>	<NONE>
Resulting File Types	`.map, .lst`	`.abs`	`.st7`

Converter Step:	`obsend macrost7, f, macrost7.s19, srec`
	or
	`obsend macrost7, f, macrost7.hex, intel`
		not applicable	`cvdwarf`

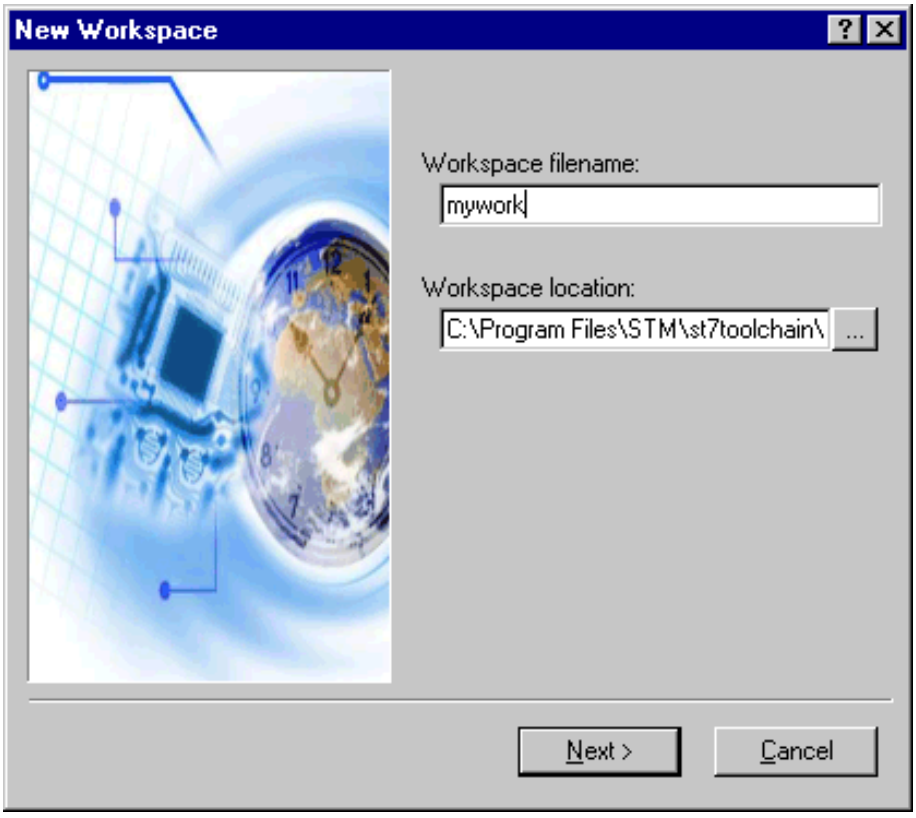
Resulting executable file:	`.s19 or .hex`	`.abs, .elf`	`.elf`
Necessary Intermediate Files:	`.map, .lst`	`.o, .dbg`	<NONE>

The **executable file(s)**, **source files** and any necessary **intermediate files** (these are listed above and contain debug information necessary to the STVD) should be located in the same project directory. You do this when you define your workspace.

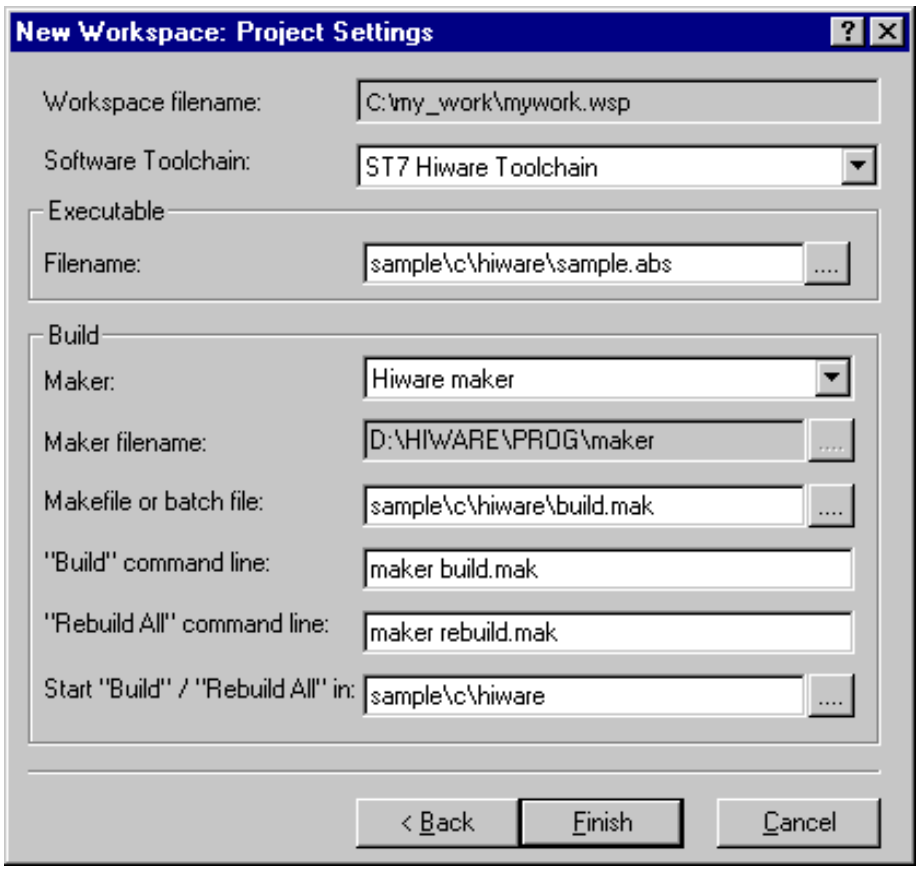
Note: It is always preferable to have access to all of the files generated by the development toolchain. However, you can load `*.s19` *or* `*.hex` *binary files directly and have limited debugging capabilities (refer to Section 3.8 on page 30).*

3.6 Creating a workspace

1 Select ***File>New Workspace***. This command opens a new window where you define the name of your workspace and the directory in which you want to work.



2 Then, click ***Next>***. The **New Workspace: Project Settings** dialog box appears:



Here you enter your software toolchain, your executable filename and your build parameters either by typing or using the drop boxes.

3 Select the toolchain and enter the name of your application's executable file. For example, if you wish to use the Hiware toolchain for ST7, your executable file will be of type `*.abs` (refer to *Table 2* on page 25)—click on the browse button  to browse to the folder where your executable file is saved and select it.

4 Next, choose the type of maker your application uses from the drop down list. In the example above, we have chosen the default Hiware maker, `maker.exe`. STVD7 will automatically look for this maker file in the folder you defined as the Hiware toolchain path.

5 Finally, you must define a make file or a batch file. There are several sample files provided with STVD7 (see *Table 1* on page 24). Here we have chosen

`build.mak` as the default make file, used when the ***Build*** command is issued, and `rebuild.mak` as the make file to use when the ***Rebuild*** command is issued.

6 After you have finished defining your project settings, click ***Finish***.

Once the workspace is opened, the Workspace window displays its contents.

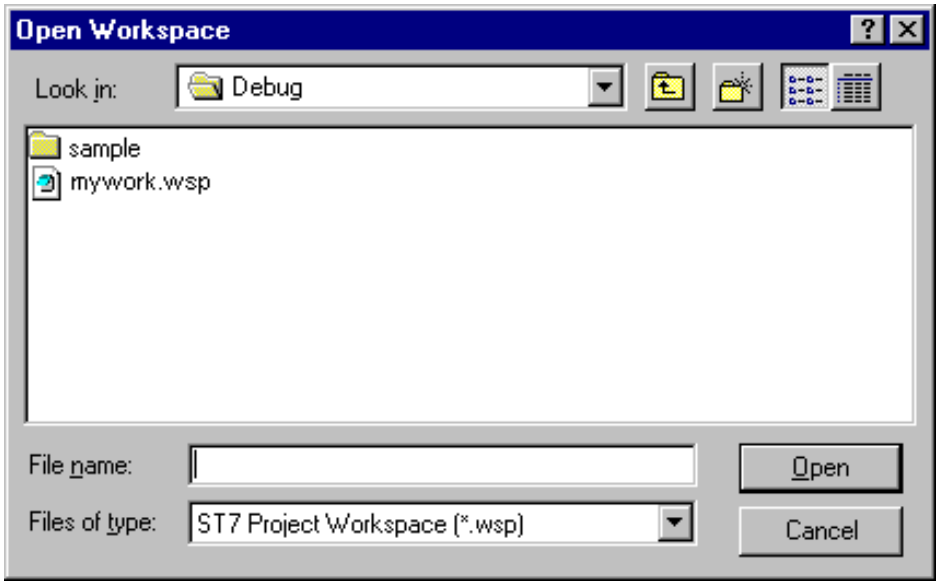
When you create a new workspace, the first time you switch to Debug context (see *Section 3.12* for an explanation of STVD7 contexts), the MCU Configuration window will automatically open to prompt you to choose you target MCU and confirm or modify its option and memory configuration (see *Section 3.13* on page 37).

3.7 Opening an existing workspace

If you have already created a workspace, you simply need to open it in order to load all of your project settings into the STVD7.

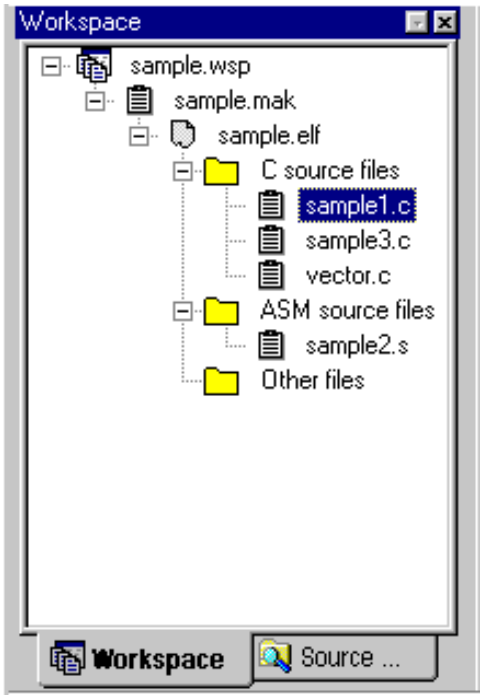
Note: There are a number of sample workspaces provided with STVD7 that you can open to get familiar with STVD7. These samples are listed in Table 1 on page 24.

1 From the main menu, select ***File>Open Workspace.***
This command opens a window where you can browse to any folder you wish, and select an existing workspace.



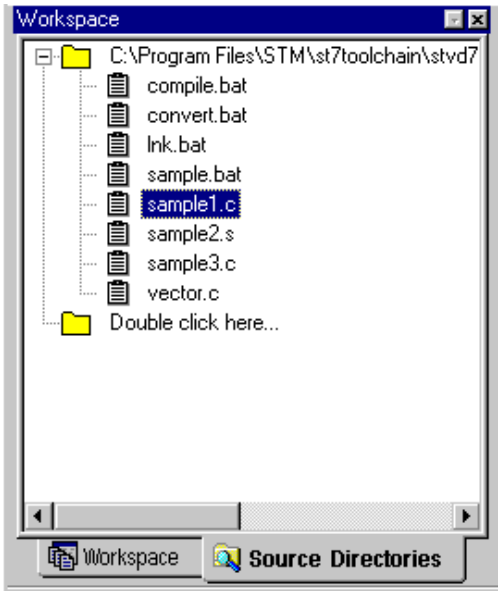
2 The Workspace window opens.

When a workspace is opened, all of the predefined project settings are loaded into the STVD7. The **Workspace** window will show a structured representation of the project. For example, `mywork.wsp` shows that it uses `build.mak` as the make file and `sample.abs` as the executable file.



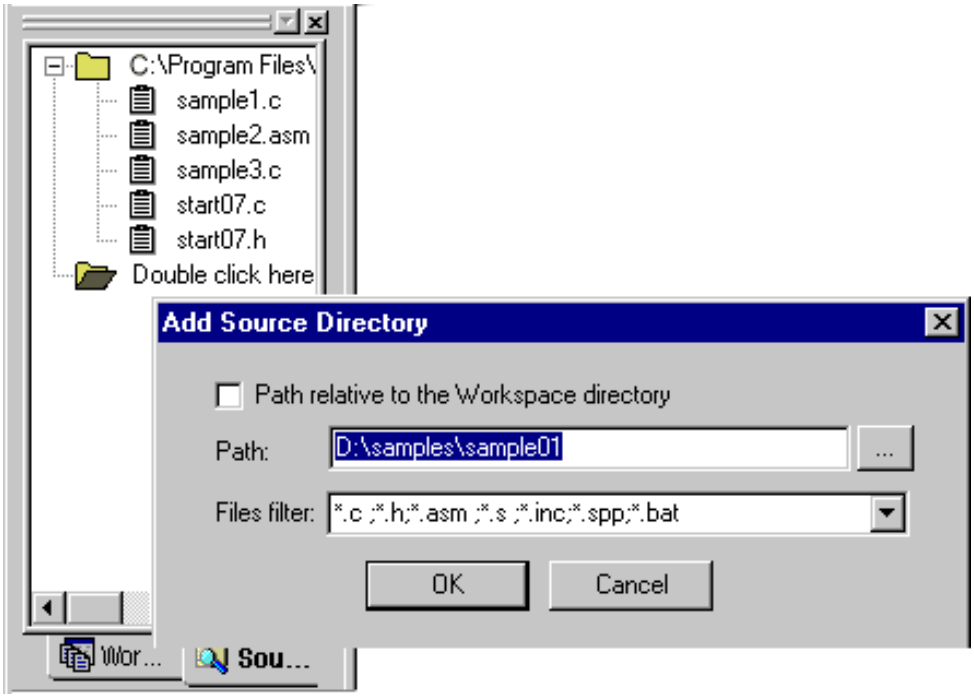
Note: *Although the name of the executable file is shown in the Workspace window, it has not yet been loaded into the emulation memory—see page 30.*

If you click on the **Source Directory** tab, the window will show every source and intermediate file type (`*.c`, `*.s`, `*.asm`, `*.h` or `*.o`) in the selected directory.



3 If there are no source files shown in the Source Directory tab of the Workspace window, or you wish to list additional files stored in another folder, you may browse to them by clicking the ***Double Click here...*** folder. The **Add Source Directory** window pops up allowing you to enter or browse for a new directory, and filter out the file types of interest. You may also choose to specify a

directory that is relative to the workspace directory by clicking on the ***Path relative to the Workspace directory*** option.



4 To load the executable file, as well as any intermediate files, click the Debug icon or the Reset Chip icon . The application and symbols will be loaded. Before you can start debugging, you must set the target hardware device by configuring the MCU.

3.8 Opening binary files

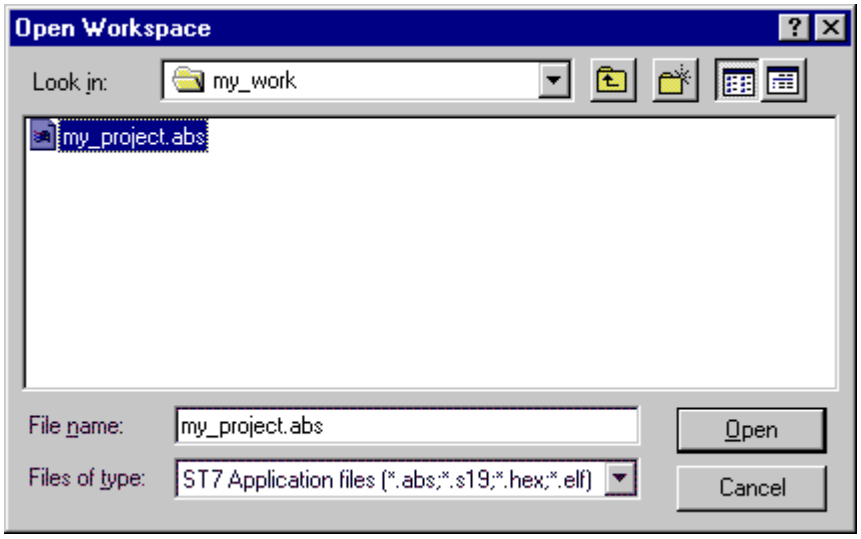
If you do not have access to the source or intermediate files generated by a toolchain, you may also load **`*.abs`**, **`*.s19`**, **`*.hex`** or **`*.elf`** files on their own using the ***Open Workspace*** command.

Note: *If you try to open* `*.s19` *or* `*.hex` *files using the following procedure, STVD7 assumes that these files have been generated using the ST7 Assembler-Linker toolchain, and that it will find the *.map files in the same directory. If you have ONLY the* `*.s19` *or* `*.hex` *files available, instead use the procedure given in Section 3.9: Opening lone programmable files (*.s19 or *.hex) on page 31.*

The range of debugging features available when you open a binary file only will be very restricted. You will only have access to basic debugging windows, such as the Disassembly and Memory Windows.

1 Launch STVD7 and select ***File>Open Workspace*** from the main menu.

2 Browse to the folder where your binary file is stored, and select ***ST7 Application files (*.abs, *.s19, *.hex, *.elf)*** in the *Files of type* field.



3 Select your binary file (`*.abs`, `*.s19`, `*.hex` or `*.elf`) and click ***Open***.

The binary code in the executable file will be loaded into STVD7 and you will be able to access the Disassembly window and the Memory window. A workspace file (of the same name as the binary file, but with an extension `.wsp`) will be created automatically.

3.9 Opening lone programmable files (*.s19 or *.hex)

If you do not have access to them `*.map` file generated by the ST7 toolchain, you may also load isolated `*.s19` and `*.hex` files from within STVD7.

The range of debugging features available when you open these files will be very restricted. You will only have access to the **Disassembly Window** and the **Memory window**.

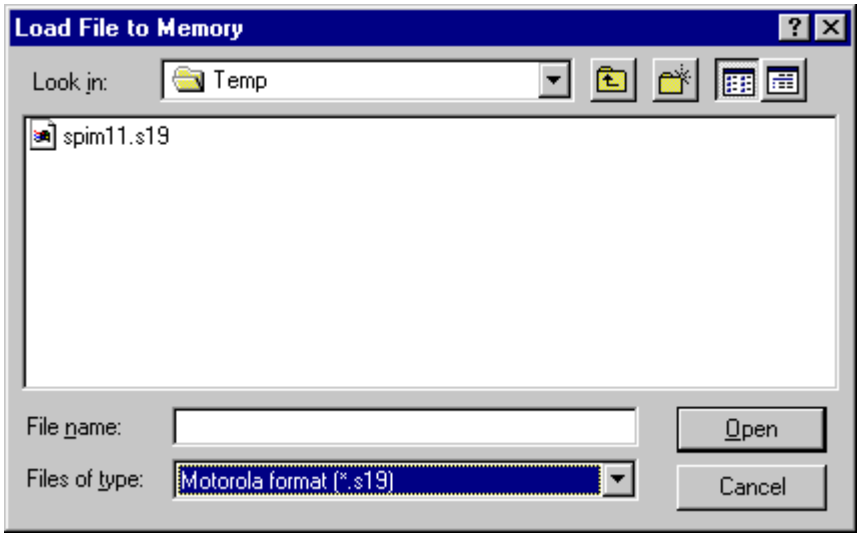
1 Launch STVD7 and select ***Debug>Start Debugging*** from the main menu or click on .

2 Open the Memory window by selecting ***View>Start Debugging*** from the main menu.

3 With the cursor in the Memory window, right-click the mouse to open the Memory contextual menu.

4 In the Memory contextual memory, select ***File>Restore Layout***. The Load File to Memory window opens.

5 Browse to the folder where your programmable file is stored, and select either the **Motorola format (*.s19)** or the **Intel format (*.hex)** in the Files of type field.

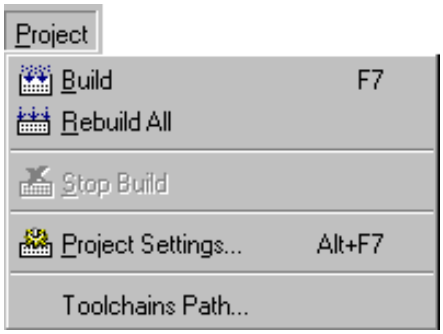


6 Select your programmable file (`*.hex` or `*.s19`) and click ***Open***.

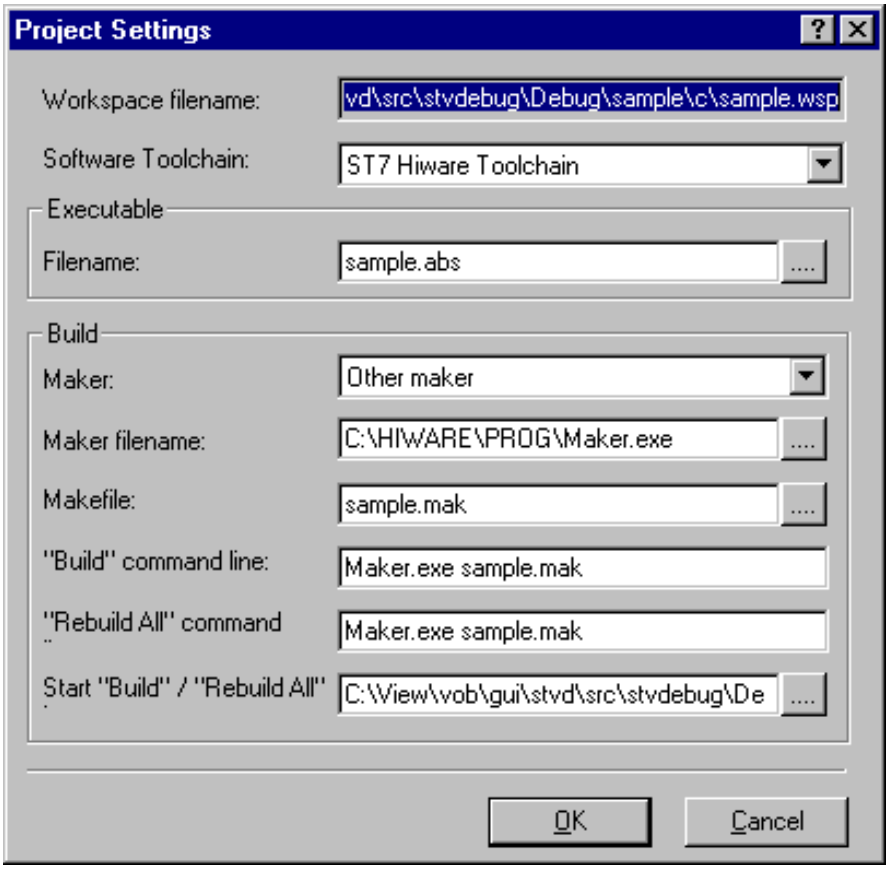
The binary code in the `.s19` or `.hex` file will be loaded into STVD7 and you will be able to access the Disassembly window and the Memory window. A workspace file (of the same name as the programmable file, but with an extension .wsp) will be created automatically.

3.10 Changing your project settings

The Project menu contains the ***Build*** and ***Rebuild All*** commands you need to recompile your application after having made changes to it in the course of debugging. You may also access your project or toolchain settings in the event you wish to change them.

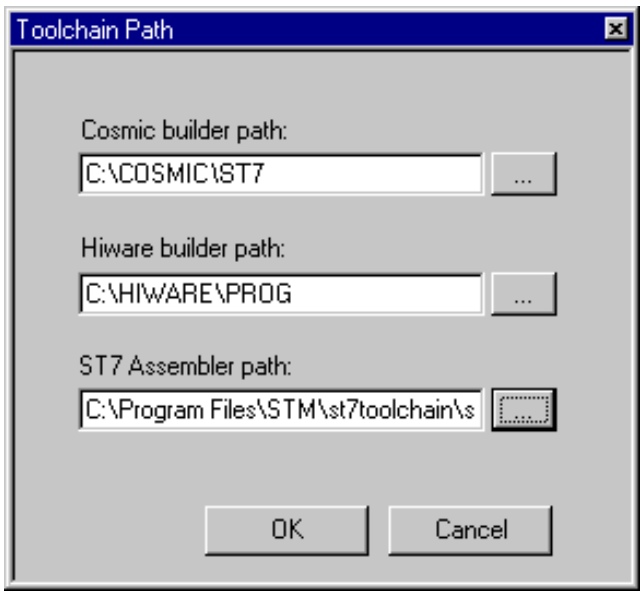


From the main menu, select ***Project>Project Settings***.



You can change your settings here and continue running your application. When you exit STVD7, the system will ask you if you want to save these settings in the workspace you have been working in. If you choose ***yes***, these will become your new workspace settings; if you choose ***no***, these settings will be lost.

The ***Toolchain Path...*** item invokes the following window:



In this window, you can define your builder and/or Assembler paths. Clicking  opens a browser window.

3.11 Saving workspaces

Whenever the current workspace is closed, it is automatically saved. This can happen either when exiting STVD or opening or creating a new workspace.

In addition to this, a workspace can be explicitly saved with the ***File>Save Workspace...*** or ***File>Save Workspace as...*** commands.

The user is given the choice of which of the workspace elements to include in the saved file. Either the **visual environment** or the **debugging information** may be saved alone, or both may be saved together. This is configured as follows:

1 From the main menu, select ***Tools>Options***.

2 In the Options window that opens (see *Figure 9 on page 35*), select the **Workspace** tab.

3 Choose whether you wish your saved workspace to include either the visual environment or the debugging information or both.

4 Select which windows will appear docked when a project is opened by checking the appropriate check boxes in the *Floating windows in the main frame* area. Only windows currently docked in the main window can be included.

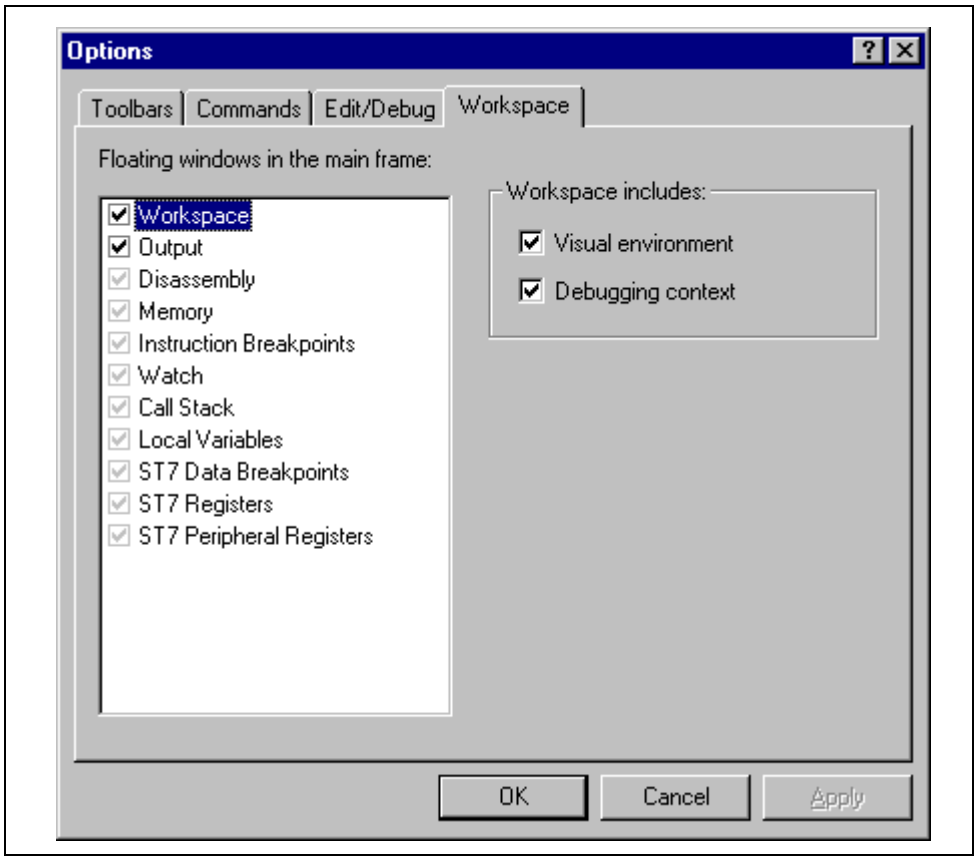


Figure 9: Workpace tab of Options window

5 Click ***Apply*** to confirm your settings.

6 Click ***OK*** close the dialog box.

In addition, open file contexts and current window positions are saved when the workspace is closed. This feature restores the workspace window, window layout and file views to that which was current when STVD7 was closed. The toolbar layout, plus customized toolbar content is also saved and restored with the workspace (options set via the tabs entitled **Toolbars** and **Commands**).

By default (i.e. when saved automatically) the workspace is saved as file <*application*>.wsp. The name of the file corresponds to the name used for the executable file (for example, <*application*>.abs for a Hiware executable file).

*Note: Using the **Configuration Setup** dialog box (available from the MCU Configuration dialog box), you can also control what type of MCU configuration information is restored from a workspace file (*.wsp).*

3.12 Debug context and Build context

There are two STVD7 contexts, the **build context** and the **debug context**. Until now, in creating a workspace, and defining your project settings, you have been in the build context. To proceed step—configuring your MCU—you need to change to the debug context.

Briefly, the two contexts are different in that:

- In the build context, you can open and close workspaces and build or re-build the application executable file.
- In the debug context you set the emulated MCU configuration (this step is described in *Section 3.13* on page 37) and debug the executable file created while in the build context.

3.12.1 Build Context

The build context is the context set when starting STVD7. In this context, it is not necessary to be connected to an emulator and the debug commands are not available. You can also edit the source files of an application and perform the use the ***Build*** command to perform compile and link actions in an interactive and iterative way to re-build the application executable file.

3.12.2 Debug Context

In this context, the following debug actions can be carried out:

- Loading, running and stopping the application.
- Defining the MCU configuration (MCU options and memory mapping).
- Viewing source and disassembled code.
- Setting instruction breakpoints with a counter and/or condition.
- Setting data breakpoints.
- Viewing local variables, memory and ST7 registers.
- Viewing history of execution from the trace buffer or with the Call Stack feature analyzing the performance of a piece of code.

3.12.3 Switching between contexts

The switch between contexts usually occurs when the ***Start Debugging*** and ***Stop Debugging*** commands are used:



From the main menu, choose ***Debug>Start Debugging*** or ***Stop Debugging*** or click on the **Start Debugging** or **Stop Debugging** icons shown at right.

While debugging, the editor allows source files to be modified. To switch to the Build context perform either a ***Build*** or ***Rebuild*** action or use the ***Stop Debugging*** command.



3.13 Configuring the MCU

After you create or open a workspace, the next step you must perform before starting your STVD7 debugging session is to define and configure the target device (MCU) that you wish to emulate.

The target device is defined and configured from the MCU Configuration window.

1 First, ensure that you are in **Debug context** by clicking on . (STVD7 has two contexts: Debug context and Build context—these are described in *Section 3.12*.)

Note: The first time you enter into the Debug context after having created a new workspace, the MCU Configuration window will be opened automatically.

2 Select ***Tools>MCU Configuration*** from the main menu. The MCU Configuration window will open.

An example of a typical MCU Configuration window is shown in *Figure 10*.

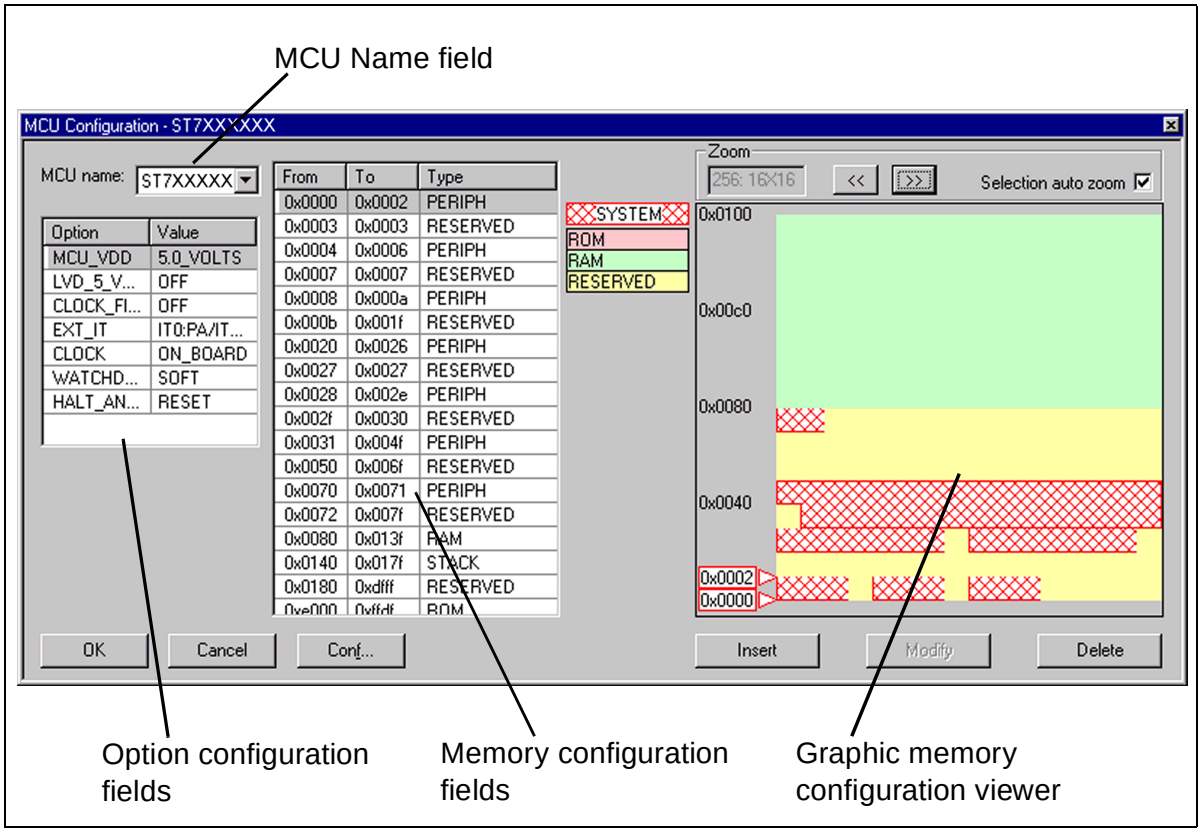


Figure 10: MCU Configuration window

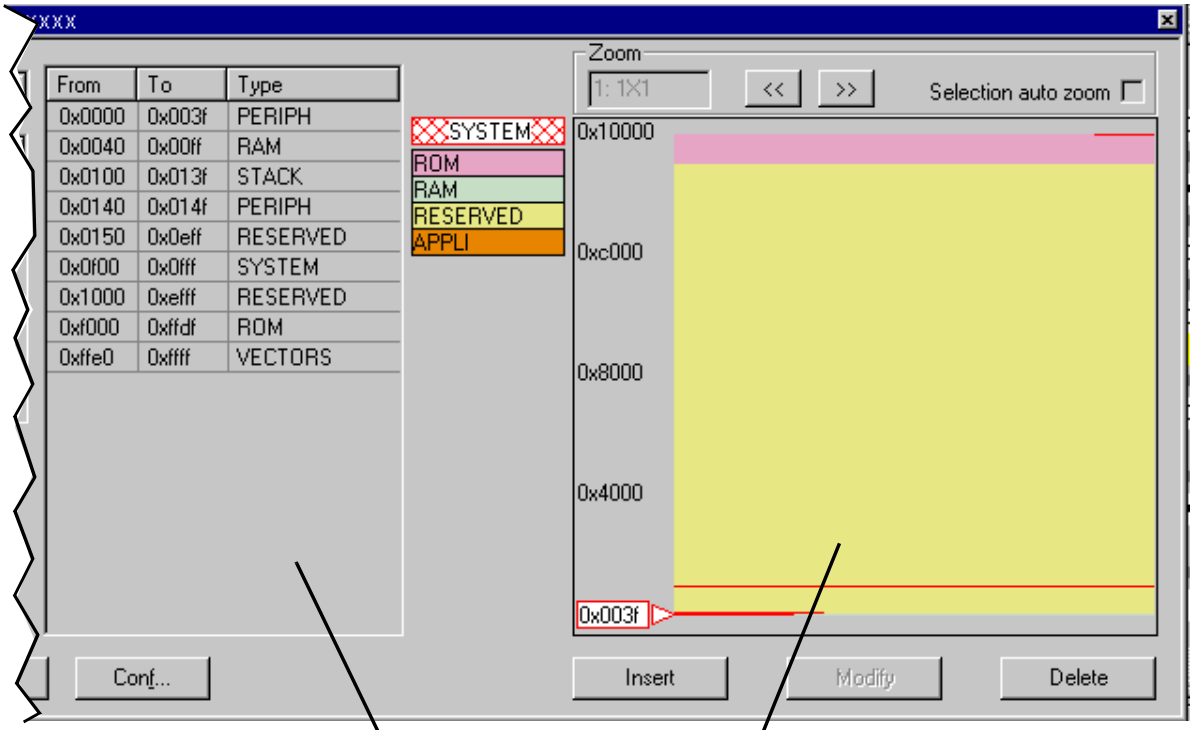
Note: The options shown in the above example may not be available for your particular target MCU.

3 **Set the Target MCU**. In the MCU name field, select the target device for which the application is intended from the dropdown box. Once a target MCU has been chosen, the *Option configuration* and the *Memory configuration* fields will show the default values for this device.

4 **Configure the MCU Options and On-Chip Peripherals.** All of the configurable options on your target hardware device are listed in the *Option configuration* fields. Beside each *option*, a default *value* is given. You may change this value by clicking on it and choosing a new value from the drop down list. This allows you to configure your target device's options and on-chip peripherals. Depending on the MCU selected, the default settings in the *Option configuration* fields will change. It is up to you to configure those options that will impact your application so that the emulator accurately emulates your target device.

Note: For more information about the configurable options available on your target hardware device, please consult your target MCU's datasheet.

5 **Configuring the MCU Memory.** The default memory settings depend on the MCU selected. However, you can configure the memory settings as you wish if your application requires non-default settings. This feature would enable you, for instance, to temporarily increase the ROM size during the development phase of your application.



There are two methods for configuring the memory settings on the MCU: by typing in the start and stop addresses of each memory zone into the **memory configuration window**, and by graphically moving the memory zone boundaries in the **graphic memory configuration viewer** (see *page 40* for more instruction).

Memory zone types

The left column of the **memory configuration window** indicates the address range of each memory zone. The right column indicates the memory type of each zone. Depending on your target MCU, the available memory types may be: Peripherals, RAM, ROM, Stack, System, EEPROM, Reserved, Vectors, Application. Some of these zones can have their type and size modified, others cannot be modified. Their definitions and properties are explained as follows:

- **Peripherals**: Microcontroller internal or rebuilt peripherals registers. Their properties are defined as in the microcontroller datasheet. This memory cannot be modified.
- **RAM**: Random-Access-Memory of the microcontroller. This memory type can be modified.
- **ROM**: Read-Only Memory of the microcontroller. Write protected. This memory type can be modified.
- **Stack**: Stack of the microcontroller. This memory type cannot be modified.
- **System**: The emulator uses this space for emulation management. This memory type cannot be modified.
- **EEPROM**: This memory is internal to the microcontroller and is located inside the emulation device. The programming of this zone is done according to an automaton found in the datasheet. This memory type cannot be modified.
- **Reserved**: This memory zone is reserved as on the microcontroller. It is not allocated to any use and is write protected. This memory type cannot be modified.
- **Vectors**: This memory zone contains the user interrupt vectors zone. It is write protected. This memory type can be modified.
- **Application**: This memory type is microcontroller-specific. The user can add memory or peripheral resources on its hardware. It is not available on every emulator. Properties are linked to the user hardware. This memory type can be modified.

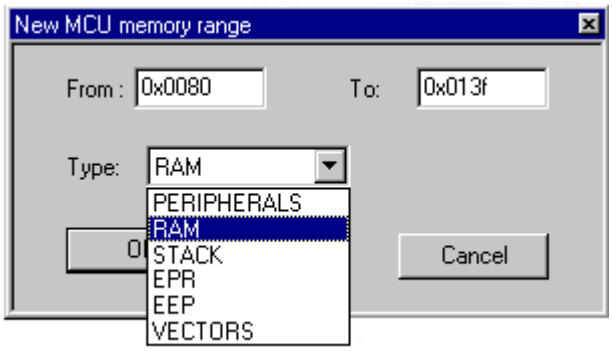
For most target MCUs, you may modify the following types of memory zone: **RAM**, **ROM**, **Reserved** and **Application**. This feature would enable you, for instance, to temporarily decrease the RAM zone, increase the size of the ROM (to exceed what is available on the real microcontroller) during the first stages of development. Once your program is functional, you can start to optimize its size by reducing your code and returning these zones to their original size. There are two different actions you may perform on the memory configuration:

- change the type of an entire existing zone.
- define a new zone of any type wherever possible.

To change an existing memory zone:

1 Select the memory zone to be modified.

2 Click on the ***Modify*** button at the bottom of the window. A **New MCU Memory Range** dialog box will open, allowing you to change either the address range and/or the memory type of the memory zone.



To create a new zone of any type:

1 Click on the ***Insert*** button. The **New MCU Memory Range** dialog box will appear.

2 Enter the address range of the new memory zone in the *From* and *To* fields.

3 Select the type of the new memory zone in the *Type* field.

4 Click ***OK*** to validate your choice.

The new memory zone will then appear in the **MCU Configuration** window **unless** you tried to create a new zone in a non-modifiable memory space (such as Stack or EEPROM).

To use the Graphic Memory Configuration viewer:

1 In the memory configuration window, click on the zone whose boundaries you wish to move.

2 Check the ***Selection auto zoom*** box in the upper right-hand corner. The graphical view of the memory configuration will be scaled so that the zone you have selected is easily visible.

3 At the upper and lower boundary of the zone, at the left-hand side of the graphical viewer, you will see a small triangle and rectangular box giving the boundary addresses of the memory zone. You can change a boundary address by dragging and dropping the triangle with the mouse to its new location. The triangle can be moved either up or down, left or right in the graphical viewer.

The MCU configuration that you specify will, by default, be saved in a workspace file (`*.wsp`) for the project. The next time the application is opened, the STVD will automatically set the MCU configuration (as well as the layout of opened windows

and other debug information) to the same conditions you had when you left the last debugging session.

If you do not wish your MCU configuration information to be saved in the workspace file, you must alter the default Configuration Setup options by clicking on the ***Conf...*** button.

3.14 Start debugging!

Once in **debug context**, you are now ready to start debugging your application using the emulator.Full documentation on how to:

- control your STVD7 work environment
- use its integrated editor
- use the many debugging windows and features

is available from the online help and the online STVD7 user manual, located under ***Help*** in the main menu.

4 EMULATOR FEATURES

The features common to all ST7 emulators are:

- real-time emulation capability (internal frequency from internal 0.5 MHz up to 8 MHz),
- full memory emulation (up to 64 Kbytes),
- real-time trace with 3 event conditions allowing selective recording,
- hardware breakpoint capability on instruction *fetch*,
- hardware breakpoint capability on address,
- breakpoint capability on invalid address access,
- breakpoint capability if trace is full,
- 1 K x 32-bit real-time trace (address, data, ctrl),
- 6 different modes to configure trace access by combining 3 event conditions,
- selective trace recording capability,
- 2-trigger output capability,
- ability to use either the on-probe oscillator, or an external source (via the front panel input) as an external clock source,
- 4 probe inputs to display application signals in the trace.

4.1 Specific features

The following features are specific to the ST7MDTS1-Active Probe:

- clock source selection,
- an application power supply follower which allows this emulator to run with application V_{CC} from 4.0 V to 5.5 V.

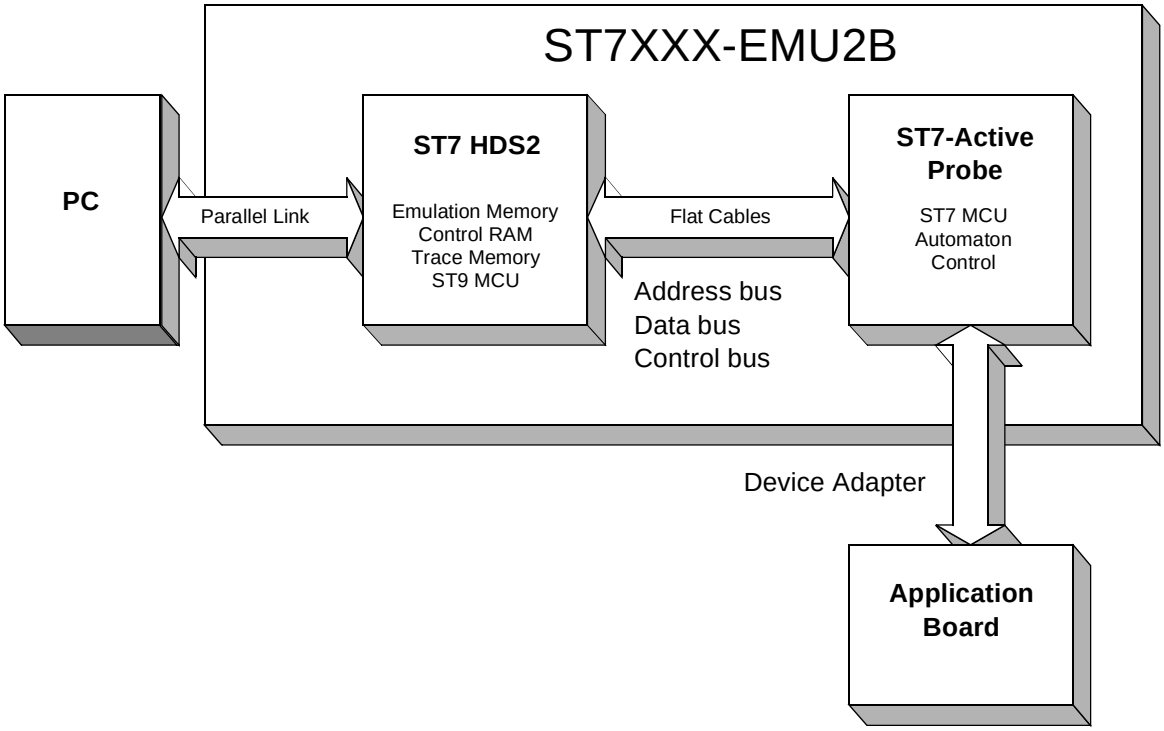
Note: When the probe is not connected to an application board or if the application board isn't powered (application V_{CC}< 4.0 V), the V_{CC} default value is 4.0 V.

4.2 Emulator architecture

The ST7MDTS1-EMU2B emulator is composed of 2 parts:

- The **ST7 HDS2** (Hardware Development System) contains all of the common resources necessary to emulate any ST7 device (such as memory and the link interfaces with the PC). This board is connected to the PC via a parallel link and to the second part by two 50-pin connectors.

- The **ST7MDTS1-Active Probe** contains the specific resources for the emulated ST72SCR devices and is used as a link between the ST7 HDS2 and your application.

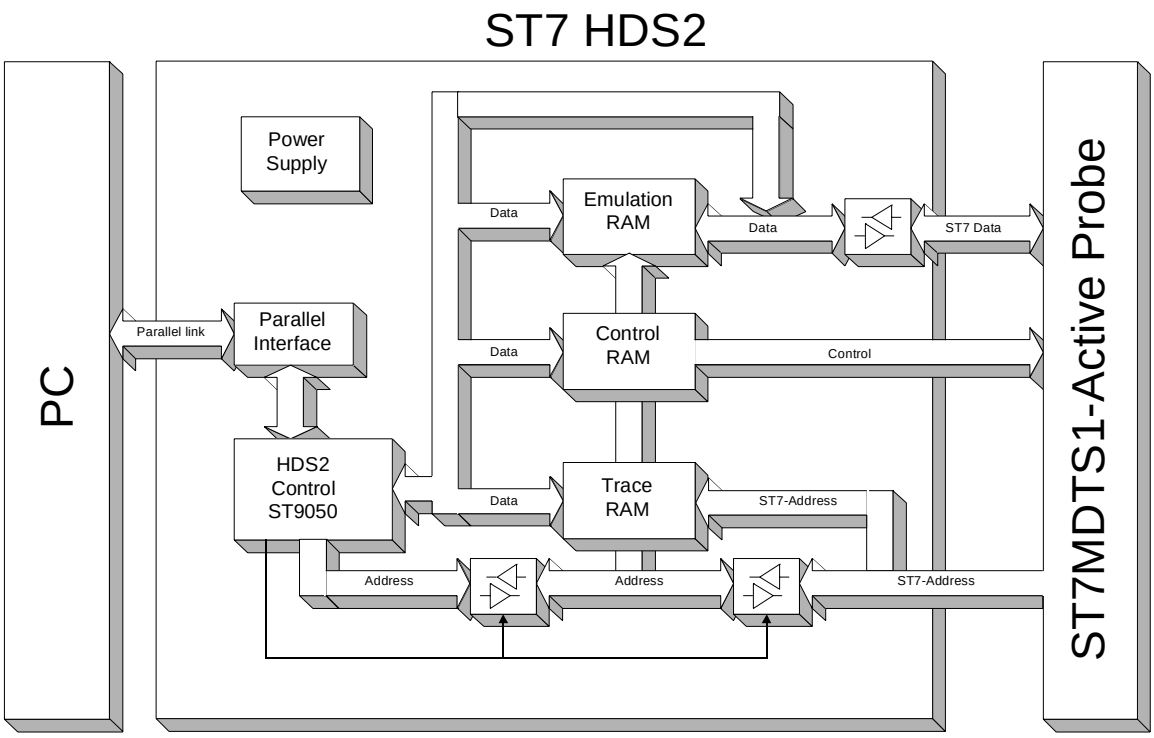


4.2.1 ST7 HDS2 hardware

The hardware functions provided by this component are listed below:

- **HDS2 CPU:** Used to control the ST7xxx-HDS2 board and manage common HDS2 features such as the parallel link with the PC.
- **RAM memory**:
  - 64 Kbytes for ROM and RAM emulation.
  - 64 Kbytes as break points control and Mapping.
  - 1 K x 32-bit as trace memory.
- **Hardware breakpoint control logic** to manage breakpoints from the 16-bit address bus.
- **Logical analyser control logic** to manage sophisticated recording and break events in the trace.
- **PC link:** parallel interface for communication with PC.

- **ST7MDTS1-Active Probe interface** — 3 buses connect the ST7 HDS2 to the **ST7MDTS1-Active Probe**:
  - Address bus (16-bit) of the ST7 emulation chip used for RAM addressing and trace.
  - DATA bus (8-bit) of the ST7 emulation chip.
  - Control bus to manage ST7MDTS1-Active Probe hardware-like breakpoint features.



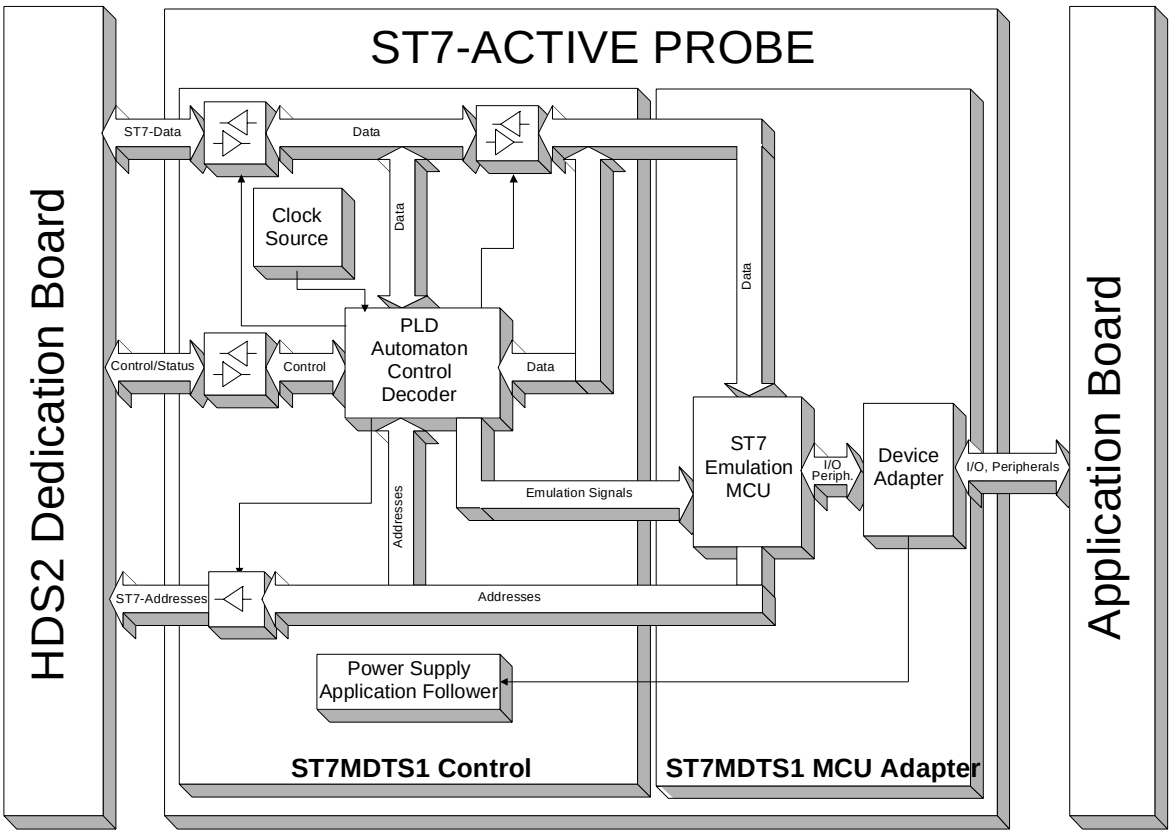
4.2.2 ST7MDTS1-Active Probe hardware

The hardware functions provided by the ST7MDTS1-Active Probe are:

- **Probe Emulation MCU:** This is an ST7 microcontroller similar to those of the emulated target device(s), which runs in emulation mode. It acts as the ST7 core and gives access to all on-chip peripherals.
- **Control logic:** Control logic is provided to manage the software execution by the user (i.e. program running and halting).
- **Application V_{CC} follower:** The probe emulation MCU is supplied with the same voltage as the application (i.e. must be in the range 4.0 V to 5.5 V).
- **ST7 HDS2 interface:** All of the communication buses connecting the active

probe to the ST7 HDS2 board are buffered:

- ST7 Address bus (16-bit) of the ST72C171 in emulation mode.
- Data bus (8-bit) of the ST72C171 in emulation mode.
- ST7 emulation chip control bus for trace recording, breakpoints and memory mapping.



4.3 Output triggers

Your ST7 HDS2 emulator has two output triggers, OUT1 and OUT2. The OUT1 and OUT2 outlets are available via SUB-click connectors located on the front panel of the ST7 HDS2 emulator box. Two SMB to BNC cables are included in your emulator package for use with these triggers.

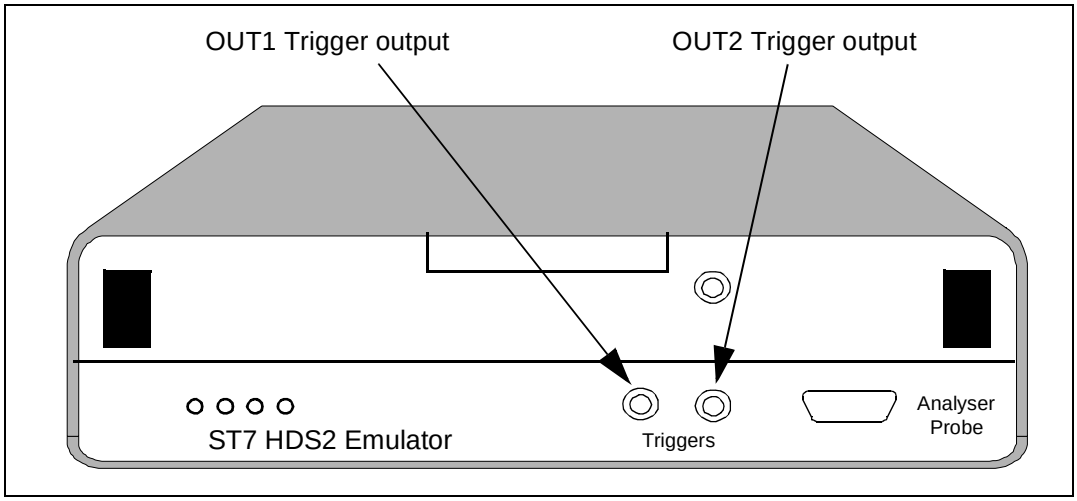
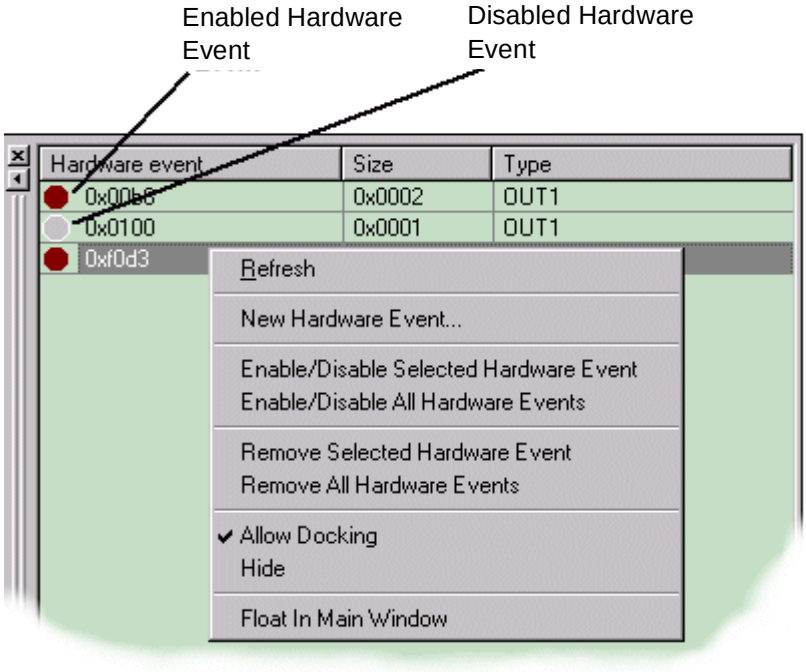


Figure 11: Output triggers on ST7 HDS2 emulators

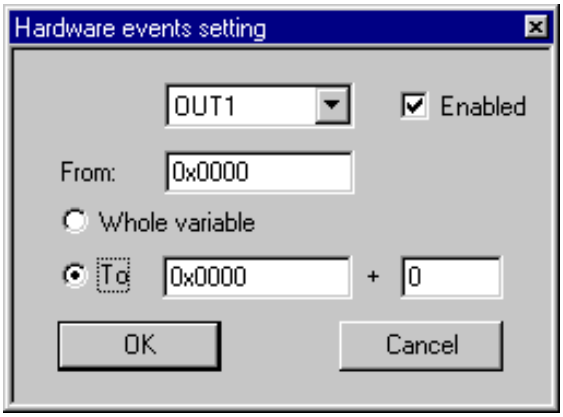
You can program the output signals to these triggers using ST7 Visual Debug:

1 From the main menu in ST7 Visual Debug, select ***View>Hardware Events***. The Hardware Events window will open in your workspace.



2 Right-click the mouse while the mouse pointer is anywhere in the **Hardware Events** window.

3 Choose ***New Hardware Event*** from the contextual menu. The **Hardware event settings** dialog box will open as below.



4 Choose the trigger output that you wish the signal to be sent to (i.e. OUT1 or OUT2) and check the ***Enabled*** box.

5 You may trigger output signals by setting an event on any of the following:

- **a whole variable**—creating an *event for synchronization*, which enables you to preset the pulse synchronization for external equipment connected to the output trigger.
- **a single address**—also creates an *event for synchronization* (see above).
- **a range of addresses**—creating an *event to measure time*, which enables you to measure the time elapsed during a subroutine execution.

A positive impulse is emitted on OUT1 and OUT2 when a specific condition is met. This impulse lasts for one Clock cycle.

4.4 Analyser probe input signals

The ST7 HDS2 allows you to use 4 external input signals (TTL level). These signals are on pins 6,7,8,9 of the Analyser Probe connector located on the front panel of the emulator as shown below.

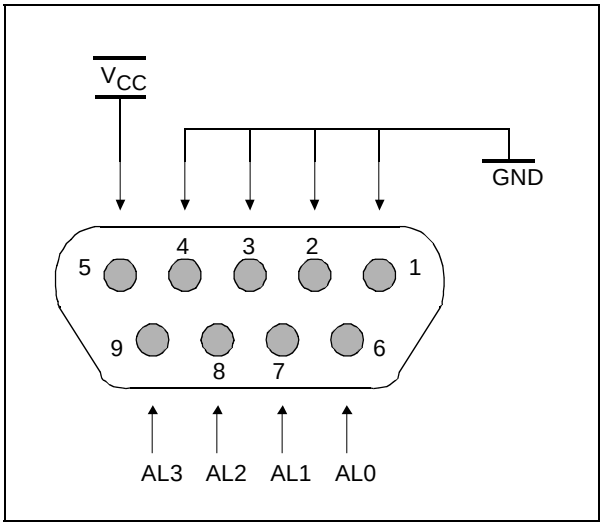


Figure 12: Analyser probe connector

You can view these probe inputs using ST7 Visual Debug. From the main menu, select ***View>Trace***. The input signal values are listed under the *Sig* column (AL3..0).

ST7 Visual Debug's **Logical Analyser** allows you to use these input signals to define trace filtering or output trigger events. From the main menu, select ***Tools>Logical Analyser*** to open the dialog box. A full description of how to use this facility to control trace recording or trigger output signals is given in the ST7 Visual Debug online help.

A rainbow-colored logic probe cable is included in your emulator package to connect your application to these inlets. Each red connector is to be connected to your signal. Each black connector is to be connected to the reference ground for the signal.

Colors are attributed as follows:

- AL0 is to be taken between the RED (signal) and BROWN (ground) wires.
- AL1 is to be taken between the YELLOW (signal) and ORANGE (ground) wires.
- AL2 is to be taken between the BLUE (signal) and GREEN (ground) wires.

- AL3 is to be taken between the GREY (signal) and PURPLE (ground) wires.

4.5 Front panel LEDs

Four LEDs on the front panel of the HDS2 box indicate the state of the development tool during emulation:

- **Power (Green)**—indicates that the 5 V power supply is ON.
- **Run (Yellow)**—indicates that the ST7 is running (not in RESET, WFI and HALT mode).
- **System (Red)**—not used with this emulator.
- **Reset (Red)**—not used with this emulator.

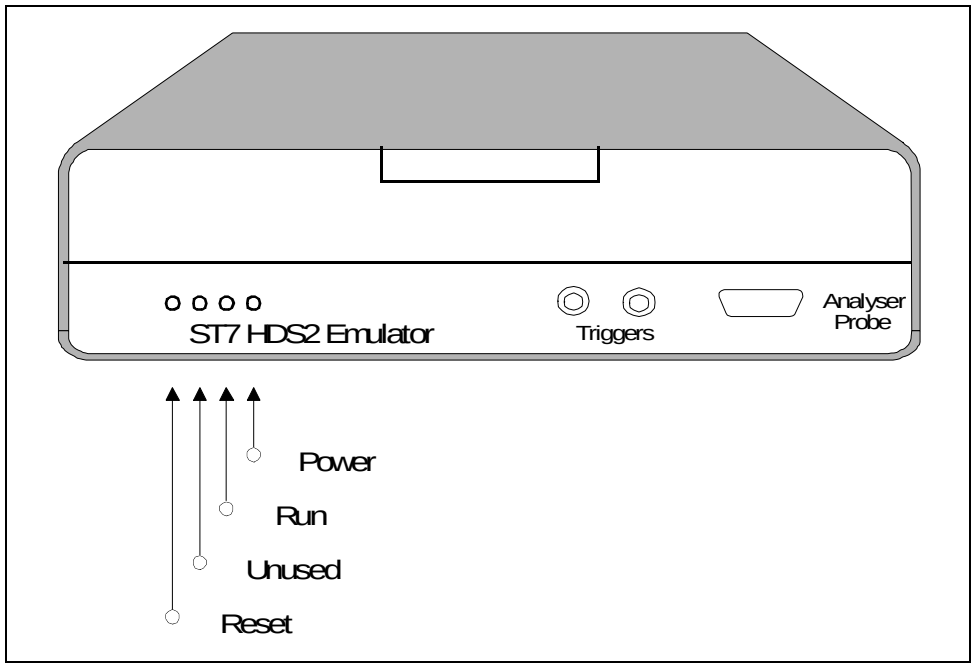


Figure 13: LEDs on emulator front panel

4.6 On-chip peripherals

You can configure certain on-chip peripherals in ST7 Visual Debug's **MCU Configuration** dialog box (refer to *Section 3.6: Creating a workspace* on page 26) so that the emulator accurately emulates your target device.

The on-chip peripheral options supported by the emulator are:

4.6.1 Clock

The emulator can work with two clock sources:

- **4 MHz** internal oscillator.
- A user-provided **External Clock** input SUB-Click located on the DB484 board of the active probe as shown in *Figure 14*.

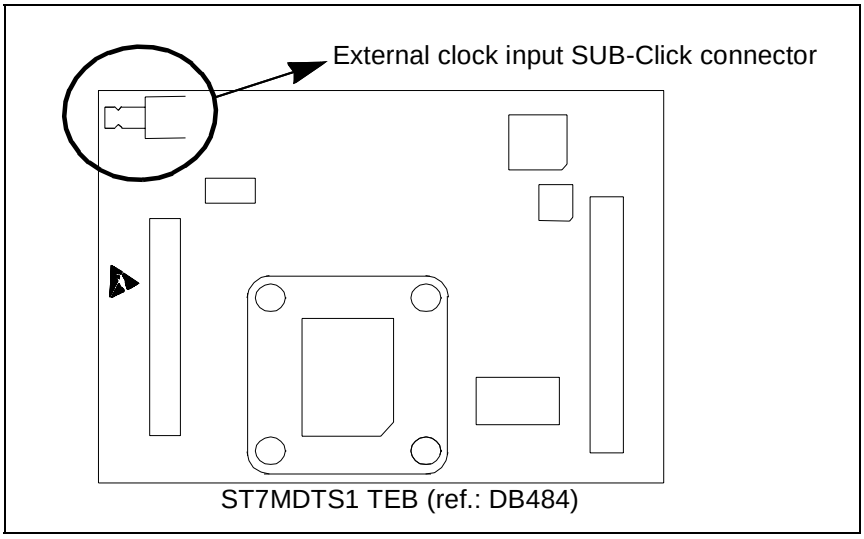


Figure 14: Location of external clock input connector

The SUB-click **external clock** connector on the probe can be used with the two SMB to BNC connectors provided in the emulator package. Voltage at these connectors must range between 0 V and 5 V. The levels are TTL.

However, if additional connectors are needed, you can purchase them at the dealers or manufacturers listed in the section entitled *Hardware spare parts* on page 62. (This list is not exhaustive.)

Note: Refer to Section 4.7: Emulation functional limitations and discrepancies on page 52 for information on the application clock available on your emulator.

4.6.2 Watchdog

This option allows you to choose whether the watchdog timer is enabled by software or by hardware.

When the **Software** option is chosen, the watchdog has to be enabled by software. When the **Hardware** option is chosen, the watchdog is always enabled.

Refer to the datasheet for your ST7 MCU for more information on the watchdog timer.

4.6.3 ISO clock source

This option allows you to determine if the ISO clock is generated by the divider (48 MHz / 12 = 4) or if it is generated by the oscillator.

4.6.4 Interrupt controller

This option bit enables or disables the nested interrupt controller. Two modes of interrupt management are possible on the ST7MDTS1 family of devices (nested or concurrent), see the devices datasheet for a detailed explanation of these two modes. When the nested mode is enabled the concurrent mode is disabled and vice versa.

4.7 Emulation functional limitations and discrepancies

The following is a list of functional limitations and discrepancies between certain features of the ST7MDTS1-EMU2B emulator and its actual target devices:

4.7.1 Power supply

The application supply follower allows this emulator to run with an application V_{DD} ranging between **4.0 V** to **5.5 V**. If the application isn't powered, or the V_{DD} < 4.0 V, the power supply is maintained at 4.0 V. If your application is powered by a voltage greater than 5.5 V, the emulator will limit this value internally to 5.5 V.

4.7.2 Clock probe

This clock source is not available on this emulator.

APPENDIX A: EMC CONFORMITY AND SAFETY REQUIREMENTS

This emulator respects the EMC requirements of the European guideline 89/336/EEC under the following conditions:

- Any tester, equipment, or tool used at any production step or for any manipulation of semi-conductor devices must have its shield connected to ground.
- All ferrites provided with the emulator kit must be attached as described in the hardware installation instructions of the relevant user manual(s).
- Your emulator must be placed on a conductive table top, made of steel or clean aluminum, grounded through a ground cable.
  All manipulation of finished goods must be made at such a grounded worktable.
- The worktable must be free of all non-antistatic plastic objects.
- It is recommended that you wear an antistatic wrist or ankle strap, connected to the antistatic floor covering or to the grounded equipment.
- If no antistatic wrist or ankle strap is worn, before each manipulation of the powered-on emulator, you must touch the surface of the grounded worktable.
- It is recommended that antistatic gloves or finger coats be worn.
- It is recommended that nylon clothing be avoided while performing any manipulation of parts.

APPENDIX B: TROUBLESHOOTING

B.1 Identifying the problem

If the following occurs:	Then...
Error Message (when starting the STVD7 for HDS Emulator): **“No message received from emulator.”**	Ensure that: • The parallel cable is connected between the emulator and one of the PC’s parallel ports (LPT1 or LPT2). Note that the use of switch boxes between the parallel port connector of your PC and the emulator are not recommended. • The development board is powered on. • The parallel cable used is the one supplied with the kit by STMicroelectronics. If none of the above items has been overlooked, this may mean that your parallel port connection needs to be reconfigured. Please refer to *Section B.2: Changing the parallel port setup on your PC* on page 55.
Error Messages (when starting the STVD7 for HDS Emulator): **"Communication error with EMULATOR board.”** or **“SYSTEM ERROR DETECTED by EMULATOR BOARD: RESET CPU.”**	Ensure that: • The flat cables linking the ST7MDTS1-Active Probe and the emulator box are properly connected. • The selected configuration file matches the connected ST7MDTS1-Active Probe configuration. If it doesn’t, from within ST7 Visual Debug, open the **MCU Configuration** dialog box by selecting ***Tools>MCU Configuration*** from the main menu. Choose the correct MCU target device in the dropdown list, then click ***OK*** to save your changes to the ****.wsp*** file for your debugging session. Exit STVD7 and power off the emulator. Power on the emulator and restart STVD7 to ensure correct installation.

B.2 Changing the parallel port setup on your PC

Under certain circumstances, you may receive the following error message:

"Connection Error (LPT1/LPT2): Interconnection failure. Verify your input/output cable."

This may mean that the setup of the LPT1 or LPT2 port on your PC is not compatible with the ST7MDTS1-EMU2B emulator.

To set up the port correctly:

1 Shut down and restart your PC in order to enter the BIOS setup.

2 Follow the messages displayed on the screen and when prompted, press the key required to enter the BIOS setup (usually a function key or the ESC key).

3 Select the parallel ports menu. (This may be listed under I/O ports.)

4 Change the Mode of the LPT port that you have connected the development board to (i.e. either LPT1 or LPT2) to one of the following compatible modes, according to the following table:

Operating System	Compatible Parallel Port Modes
Windows 95	ECP, EPP, Bidirectional or Centronics
Windows 98	EPP, Bidirectional or Centronics
Windows NT4	ECP, EPP, Bidirectional or Centronics

5 Save your changes and exit the BIOS setup.

B.3 Running the hardware test

The **Hardware Test** in the STVD7 for HDS2 lets you check that your emulator is correctly connected, configured and working. You can test components of the development board individually, or all at the same time.

If problems occur during debugging (such as bad debugger responses and unexpected behavior), you should check for hardware problems using the Hardware Test function, and if any are detected, contact your STMicroelectronics sales representative (see *Product Support* on page 61).

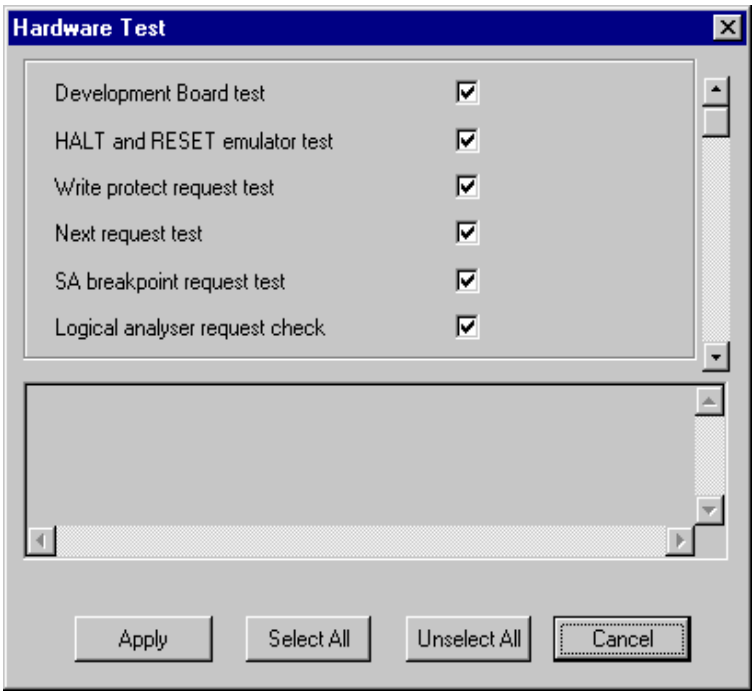
You may open the **Hardware Test** dialog box by:

- selecting, from the Main Menu, ***Emulator>Hardware Test***, or by
- clicking on the Hardware Test icon  in the **Emulator** toolbar.

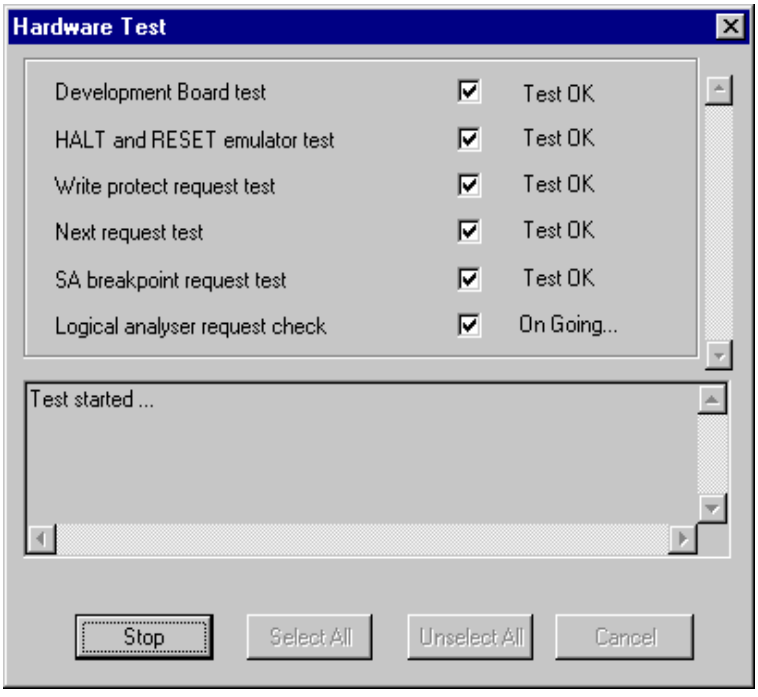
Caution: *Be cautious in performing a Hardware Test on the emulator while an application is open. The opened application WILL BE corrupted by the hardware testing process. If you find that your application has been corrupted, simply close the application, and reopen it.*

The Hardware Test dialog box shows a list of different tests that can be performed on the emulator.

Check the box of each test that you wish to perform (they are all checked by default) and click **Apply** to start the hardware test.



The Hardware tests will be performed one by one, and the results summarized in the dialog box as shown on the right:



APPENDIX C: GLOSSARY

Application board

This is the printed circuit board onto which you wish to connect the target ST7 MCU. It should include a socket or footprint so that you can connect the application board to your emulator or development kit using the probe and the appropriate device adapter. This allows you to emulate the behavior of the ST7 MCU in a real application in order to debug your application program.

Device adapter

Device adapters are included in your emulator kit to allow you to connect the emulator to your application board. The type of device adapter depends on the target device's packaging. Many MCUs come in more than one different package, and you should therefore use the device adapter that corresponds to the type of package you have chosen for your application.

DIL

Dual In Line. Designates a type of device package with two rows of pins for thru-hole mounting. Sometimes also called DIP (Dual In-line Package).

ECP

Extended capabilities port communication standard.

EPP

Enhanced parallel port communication standard.

Footprint

Designates the dimensions of the location of a component on a printed circuit board or in a socket. It depends on the number of pins, their size, type and positioning. The footprint of each ST7 device is specified in the datasheet in the section titled *Package Mechanical Data.*

MCU

Microcontroller Unit. Otherwise referred to as the "target device" throughout this manual. This is the core product (or family of products) for which the Development Kit is designed to act as an emulator and programming tool. In general terms, an MCU is a complete computer system, including a CPU, memory, a clock oscillator and I/O on a single integrated circuit.

ST7MDTS1-Active Probe

A printed card having connector pins that allow you to connect the Emulator to the MCU socket of the user application board. Using the active probe allows the HDS2 emulator to function as if it were the target device embedded in your application. The probe is connected to the emulator by two flat cables.

Program counter (PC)

The program counter is the CPU register that holds the address of the next instruction or operand that the CPU will use.

RC network

Resistor-capacitor network.

SDIP

Serial Dual In-line Package.

SO

Small outline. Designates a type of device package with two rows of pins for SMD or socket mounting.

ST7 Visual Debug (STVD7)

A graphic debugger software package that allows you to debug applications destined for the ST7 family of MCUs, either using a built-in simulator function, a Development Kit or an HDS2 Emulator.

Target device

This is the ST7 device that you wish to use in your application, and which the development kit will emulate for you.

User application board

Designates your application board.

PRODUCT SUPPORT

If you experience any problems with this product or if you need spare parts or repair, contact the distributor or ST sales office where you purchased the product.

Getting prepared before you call

Collect the following information about the product before contacting ST or your distributor:

1 Name of the company where you purchased the emulator kit.

2 Date of purchase.

3 Order Code: Refer to the side of your emulator kit box. The order code will depend on the region for which it was ordered (i.e. the UK, Continental Europe or the USA).

4 Serial Number: The serial number is located on the rear panel of the emulator box.

5 Target Device: The sales type of the ST7 microcontroller you are using in your development.

Contact list

*Note: For **American and Canadian customers** seeking technical support the US/Canada is split in 3 territories. According to your area, contact the following sales office and ask to be transferred to an 8-bit microcontroller Field Applications Engineer (FAE).*

Canada and East Coast

STMicroelectronics
Lexington Corporate Center
10 Maguire Road, Building 1, 3rd floor
Lexington, MA 02421
Phone: 781-402-2650

Mid West

STMicroelectronics
1300 East Woodfield Road, Suite 410
Schaumburg, IL 60173
Phone: 847-517-1890

West coast

STMicroelectronics, Inc.
30101 Agoura Court
Suite 118
Agoura Hills, CA 91301
Phone: 818-865-6850

Europe

France (33-1) 47407575
Germany (49-89) 460060
U.K. (44-1628) 890800

Asia/Pacific Region

Japan (81-3) 3280-4120
Hong-Kong (852) 2861 5700
Sydney (61-2) 9580 3811
Taipei (886-2) 2378-8088

Software updates

You can get software updates from the ST Internet web site **http://mcu.st.com**. For information on firmware and hardware revisions, call your distributor or ST using the contact list given above.

Hardware spare parts

Most of the hardware you will require is included in the emulator kit. However, some special applications may require additional parts, such as connecting an external clock, or you may need additional sockets for your application board.

Below is a list of manufacturers and dealers of SMB and BNC connectors that can be used with our product.

European manufacturer and product references:

Radiall
For worldwide sales locations, visit Radiall's website at:
www.radiall.com

The EXTERNAL clock male connector on the emulation probe has the following commercial reference:

- In SMB range: Ref.: 114665.

Adaptable Female connectors that fit this connector are:

- SMB upright range
  Ref.: 114005 for cable 2,6.
  Ref.: 114003 for cable 4,2.
  Ref.: 114009 for cable 3,8.
- SMB kneed range
  Ref.: 114165 for cable 2,6.
  Ref.: 114163 for cable 4,2.
- SMB to BNC range
  Ref.: 191214. Adapter SMB female / BNC male.
  Ref.: 191215. Adapter SMB female/ BNC female.

USA manufacturer and product references

R-Tek
411 Quentin Road
Palatine, IL 60067
Phone: (847) 934-7900
Fax: (847) 934-7946

Adaptable female connector part numbers:

- CCAX00168-2: cable length 2 ft, with SMB plug to BNC plug.
- CCAX00168-3: cable length 3 ft, with SMB plug to BNC plug.

Index

A

Active Probe
- architecture 45
- definition of..... 60
- hardware 45

analyser probe signals 49
application board
- definition of..... 59

C

clock
- selecting source 51

clock probe 52
configuration
- analyser probe input signals 49
- output triggers 47

connections
- emulator power supply 18
- emulator to PC 14
- probe to emulator..... 14

D

DB467 11
DB484 11

E

ECP
- definition of..... 59

EMC compliance 15
EMC compliancy
- requirements for..... 53

emulator kit
- configuration of 7
- delivery checklist..... 11
- functional limitations/discrepancies..... 52
- installing software for 19
- main functions of..... 6
- operation of..... 8
- software and documentation for..... 8

F

ferrites
- attaching to cables.....15

finished goods
- manipulation of 53
- safety requirements 53

H

hardware
- installation.....13

hardware test.....56
HDS2 emulators
- main features.....43

I

input signals.....49
installation
- hardware.....13
- STVD7 19

interrupt controller.....52
ISO clock source 52

L

LEDs.....50
load
- binary files 30

M

MB176 11
MCU
- emulated.....5
- on-chip peripherals 51

MCU configuration.....37
MCU memory
- configuring 39
- types 39

O

on-chip peripherals 51
- clock 51
- interrupt controller.....52
- ISO clock source 52
- watchdog 52

output triggers.....47

Index

P

parallel port
troubleshooting connection problems ... 55
PC
system requirements ... 11
peripherals
configuring target ... 38
power supply ... 52
powering up
device sequence ... 18
important warning ... 18
project settings
modifying ... 32

R

RAM
minimum ... 11
ROM size ... 39

S

safety requirements ... 53
SDIP
definition of ... 60
SO
definition of ... 60
SO20 package ... 17
software
updates ... 62
ST7MDTS1 EMU2B
architecture ... 43
specific features of ... 43
ST7MDTS1-Active Probe
main features ... 43
STVD7
about ... 21
build context ... 36
contexts ... 36
creating a workspace ... 26
debug mode ... 36
installing ... 19
main features ... 21
MCU configuration ... 37
opening binary files ... 30
opening workspaces ... 28
supported application files ... 23
supported toolchains ... 23
switching between contexts ... 37
toolchain paths ... 20
workspaces ... 22
support
contact numbers for ... 61
for development kit ... 61
web address ... 9

T

target device
definition of ... 60
supported ... 5
TQFP64 package
connecting ... 16
troubleshooting ... 55
connection error ... 55

U

user application board
definition of ... 60

W

watchdog option ... 52
working environment recommendations ... 53
workspaces
creating new ... 26
saving ... 34

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without the express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics.

Intel® is a U.S. registered trademark of Intel Corporation.

Microsoft®, Windows® and Windows NT® are U.S. registered trademarks of Microsoft Corporation.

©2001 STMicroelectronics - All Rights Reserved.

Purchase of I^2C Components by STMicroelectronics conveys a license under the Philips I^2C Patent. Rights to use these components in an I^2C system is granted provided that the system conforms to the I^2C Standard Specification as defined by Philips.

STMicroelectronics Group of Companies

Australia - Brazil - China - Finland - France - Germany - Hong Kong - India - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain Sweden - Switzerland - United Kingdom - U.S.A.

http://www.st.com