

ST7MDT2-EMU2B HDS2 Series Emulator User Manual

Release 1.2

June 2000



Ref: DOC-ST7MDT2-EMU2B

Obsolete Product(s) - Obsolete Product(s)

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	7
1.3 Software and Documentation for the Emulator Kit	8
1.4 About this Manual....	8
1.5 Related Documentation	9
1.6 Getting Assistance	9
Chapter 2: Getting Started	11
2.1 Your System Requirements	11
2.2 Delivery Checklist	11
2.3 Installing the Hardware	12
2.4 Debuggers Supporting the ST7 HDS2 emulator	20
2.5 Using the TQFP64 Device Adapter (Ref.: DB379)	20
2.6 Accessing Device Pins	22
Chapter 3: STVD7	25
3.1 Installing STVD7	25
3.2 Launching STVD7	26
3.3 About STVD7 debugging features	27
3.4 Workspaces	28
3.5 Toolchains and application files	29
3.6 Creating a workspace	32
3.7 Opening an existing workspace	34
3.8 Opening binary files	36
3.9 Changing your project settings	37
3.10 Saving workspaces	39
3.11 Debug context and Build context	41
3.12 Configuring the MCU	42
3.13 Start debugging!	47
Chapter 4: Emulator Features	49
4.1 Main Features of the ST7 HDS2 Emulator series	49
4.2 Specific Features	49
4.3 Emulator Architecture	49
4.4 Output Triggers	53
4.5 Analyser Probe Input Signals	55

Table of Contents

4.6	Front Panel LEDs	56
4.7	On-Chip Peripherals	57
4.8	Emulation Functional Limitations and Discrepancies	59
Appendix A: Troubleshooting		63
A.1	Identifying the Problem	63
A.2	Changing the Parallel Port Setup on Your PC	64
A.3	Running the Hardware Test	65
A.4	QFP64/TQFP64 & QFP44/TQFP44 Footprint Issues	67
Appendix B: Hardware Schematics		69
B.1	Component layouts	69
B.2	Device adapter pin-matching diagrams	74
Appendix C: Glossary		77
Product Support		79
	Getting prepared before you call.....	79
	Contact List	79
	Software updates	80
	Hardware spare parts	80
Index		83



1 INTRODUCTION

Thanks for choosing ST7! This manual will help you get started with the ST7MDT2-EMU2B.

The ST7MDT2-EMU2B package is a development tool designed for emulation of the following microcontrollers of the MDT2 family:

Supported Devices
ST 72511 R
ST 72512 R
ST 72311 R
ST 72532 R
ST 72124 J
ST 72314 J
ST 72334 J
ST 72314 N
ST 72334 N

The ST7MDT2-EMU2B package will assist you in debugging your application hardware as well as your software. The ST7MDT2-EMU2B kit comes with a new debugger software package—ST7 Visual Debug—which contains all of the necessary resources to help you design, develop and debug ST7 application software running in a real environment.

Note: If you come across any terms or abbreviations you do not understand, you can check their meaning in the Glossary on page 77.

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 ST7MDT2-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 **ST7MDT2-Active Probe**, dedicated to the family, which constitutes the physical link between the emulator and your application.

Note: When receiving the ST7MDT2-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,
 - ST7MDT2-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 ST7MDT2-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 ST7MDT2-Active Probe, to which a device adapter can be fixed so that you can connect the emulator to your application board.

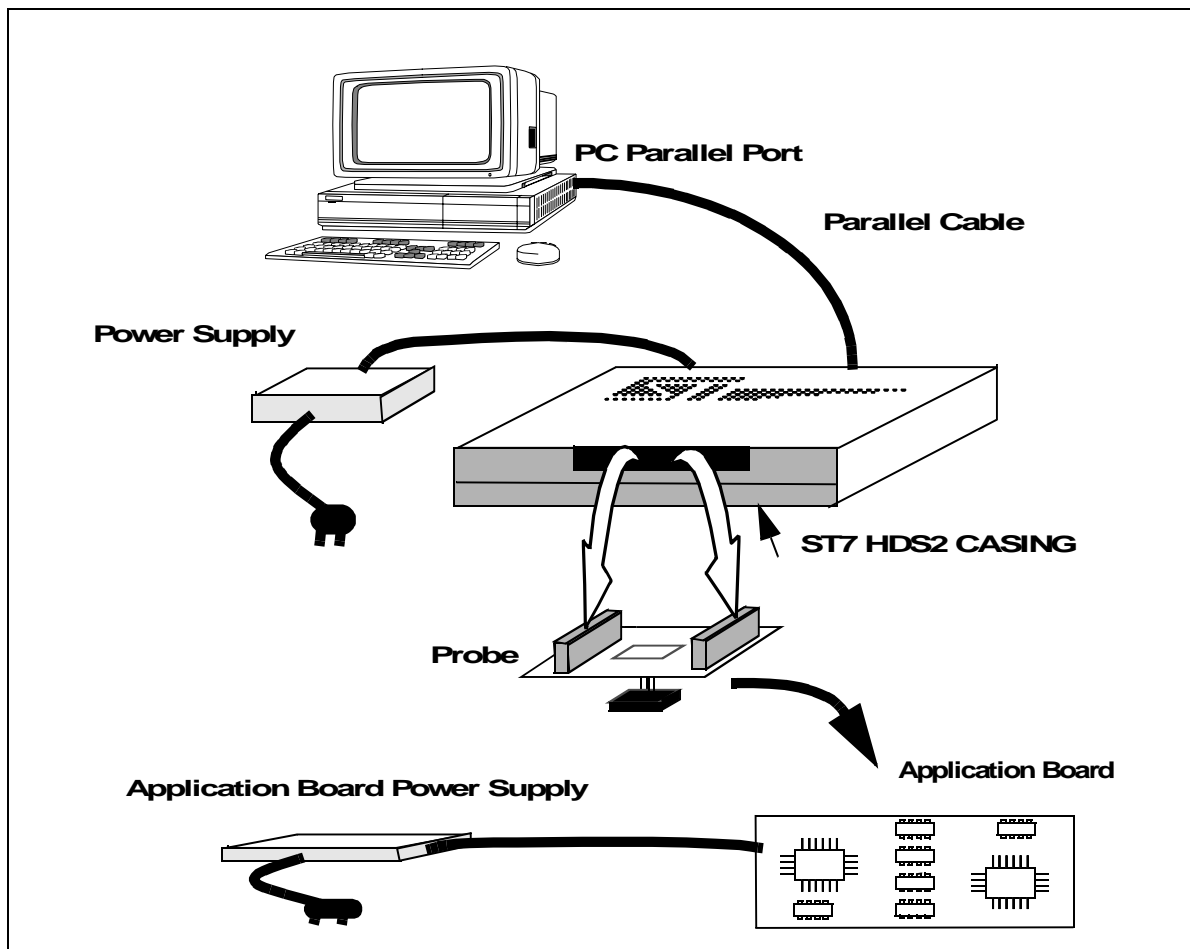


Figure 1: ST7MDT2-EMU2B General Configuration

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 25, 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.: ST7MDT2-EPB2 —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, ST7 Visual Debug, 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, this manual (in PDF version), and other useful reference materials.
- Datasheets for the ST7 MCU family.

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 ST7 Visual Debug is described in *Chapter 3: STVD7* on page 25.

The emulator kit's hardware features are described in *Chapter 4: Emulator Features* on page 49.

1.5 Related Documentation

To get all the essential information about your ST7 MCU and the software that comes on the CD-ROM with the emulator kit, you will need to refer to these documents (also contained on the CD-ROM):

- ST7-Family Data Sheets
- *ST7-Family 8-bit MCUs Product Overview* (Ref. BKST7/2)
- *Software Tools for the ST7 Family* (Ref. Doc-ST7ASMLK-SW)
- ST7-Family Programming Manual

1.6 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 *Product Support* on page 79. We'll be glad to help you!



0

2 GETTING STARTED

2.1 Your System Requirements

The ST7MDT2-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.: (ST7MDT2-EMU2B), is delivered with the following (refer to *Figure 2*):

- 1 One emulator box containing the ST7-HDS2 main board (ref.: MB176).
- 2 One parallel cable.
- 3 Two 50-wire flat cables to connect the ST7-HDS2 main board to the emulation probe.
- 4 One emulation probe—the ST7MDT2-Active Probe (ref.: DB407).
- 5 A TQFP64 device adapter (ref.: DB389) for connecting the ST7-Active Probe to your application board. Used with (f) below. See *QFP64/TQFP64 & QFP44/TQFP44 Footprint Issues* on page 67 for additional information on QFP64/TQFP64 issue.
- 6 A QFP64 socket adapter (ref.: DB200). (This fits between the TQFP64 device adapter (ref.: DB389) and the QFP64 Yamaichi socket.)
- 7 A Yamaichi TQFP64 socket (ref.: IC149).
- 8 A SDIP56 (ref.: DB408) device adapter for connecting the ST7-Active Probe to your application board.
- 9 A SDIP56 to SDIP42 (ref.: DB326) device adapter.
- 10 A TQFP64 device adapter for use in Device Mode (ref.: DB379) plus its User Manual. (Not shown.)
- 11 One User Manual for the ST7 Family Software Development Tools (assembler, linker, and formatter). (Not shown.)
- 12 This manual. (Not shown.)

- 13 A CD-ROM containing ST7 information and software, including ST7 Visual Debug and a Windows® Epromer. (Not shown.)

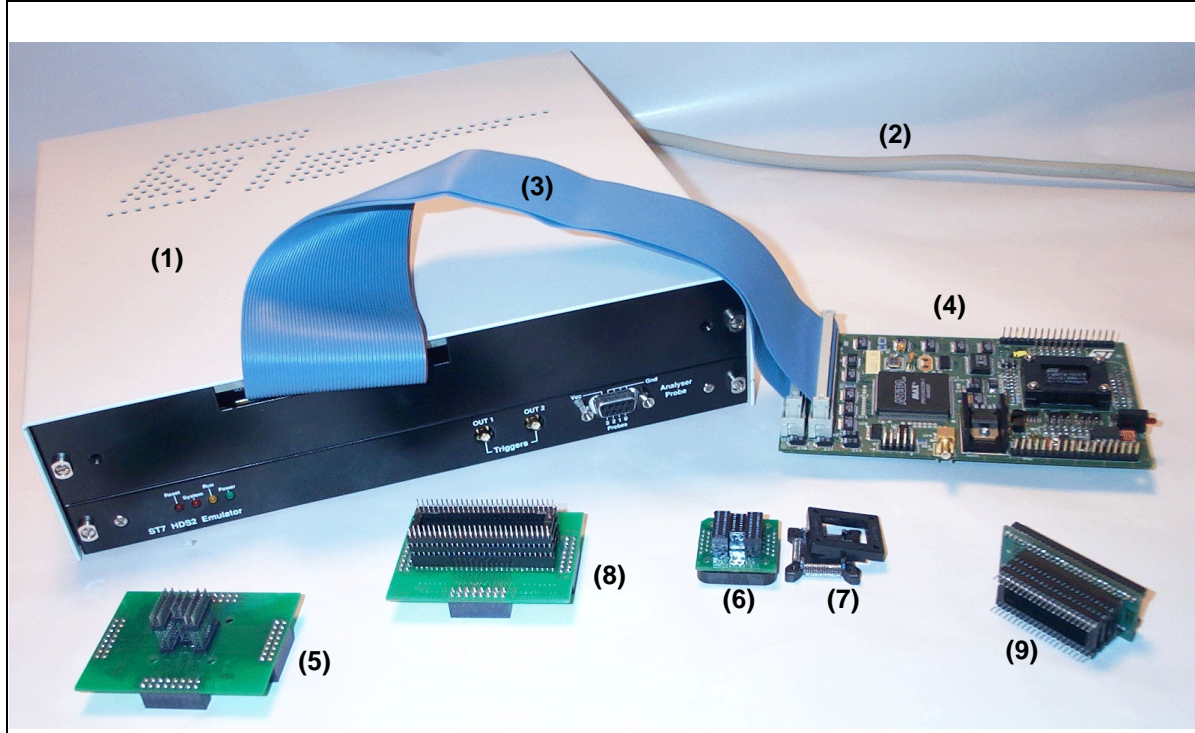


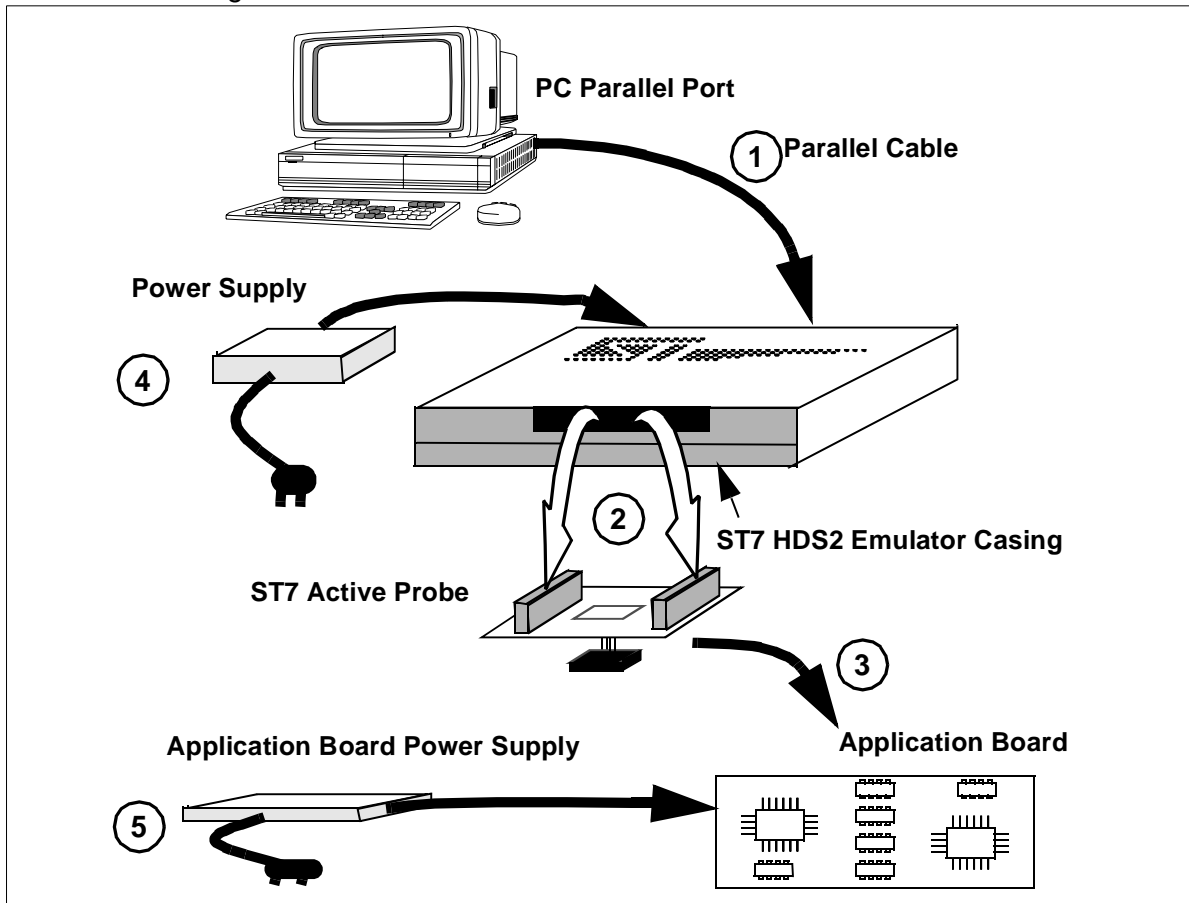
Figure 2: Main Components of ST7MDT2-EMU2B Emulator Kit

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 on your application to the appropriate 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 power supply.

A connection flow diagram is shown hereafter. Step details are discussed in the following sections.



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 3*.

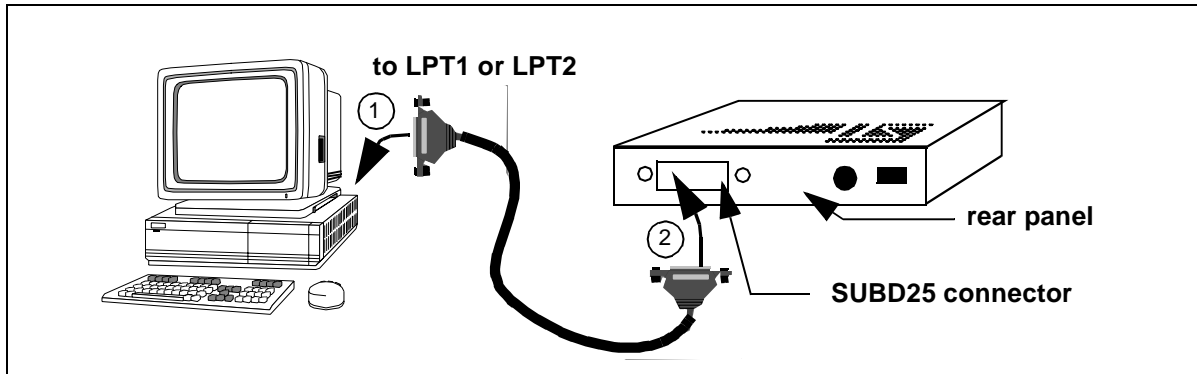


Figure 3: Connecting the Emulator to the PC

Step 2: Connecting the HDS2 and the probe

- 1 Ensure that the application and the emulator are **powered-off**.
- 2 Plug the two 50-wire flat cables into J1 and J2 as described below:

ST7-HDS2 Emulator (ref.: MB176)	ST7MDT2-Active Probe (ref.: DB407)
Upper cable	J2
Lower cable	J1

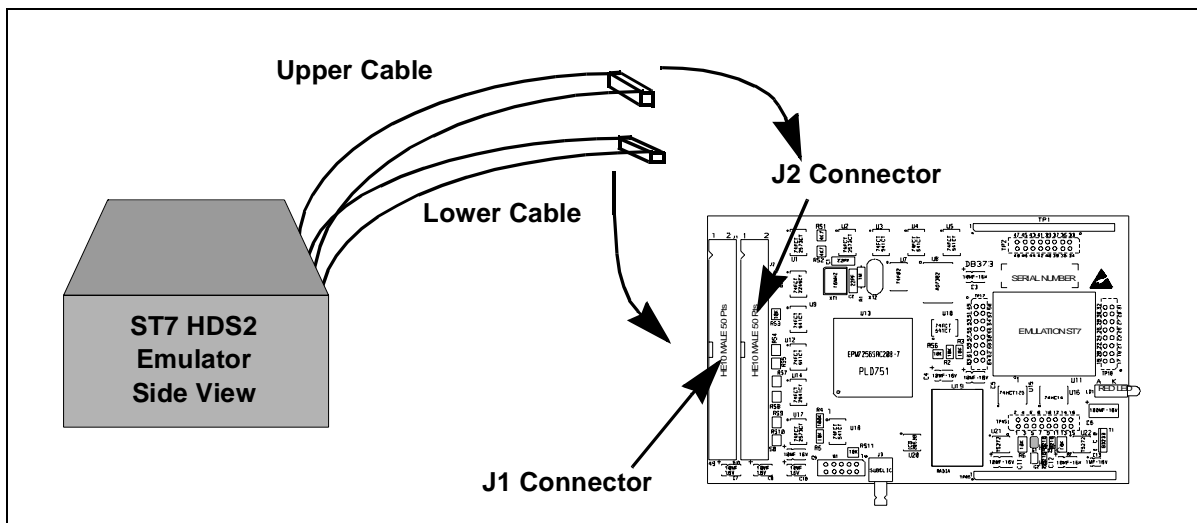


Figure 4: Connecting the Emulator to the ST7-Active Probe

3 EMC-Compliant Probes: In order to work under an EMC-compliant environment, you will have to clip one or two EMC-ferrite on each 50-wire flat cable linking the probe to the emulator box. Place these ferrites as close to the emulator window as possible. Four ferrites are provided in the package. See *Figure 5* for an illustration of where to attach the ferrites.

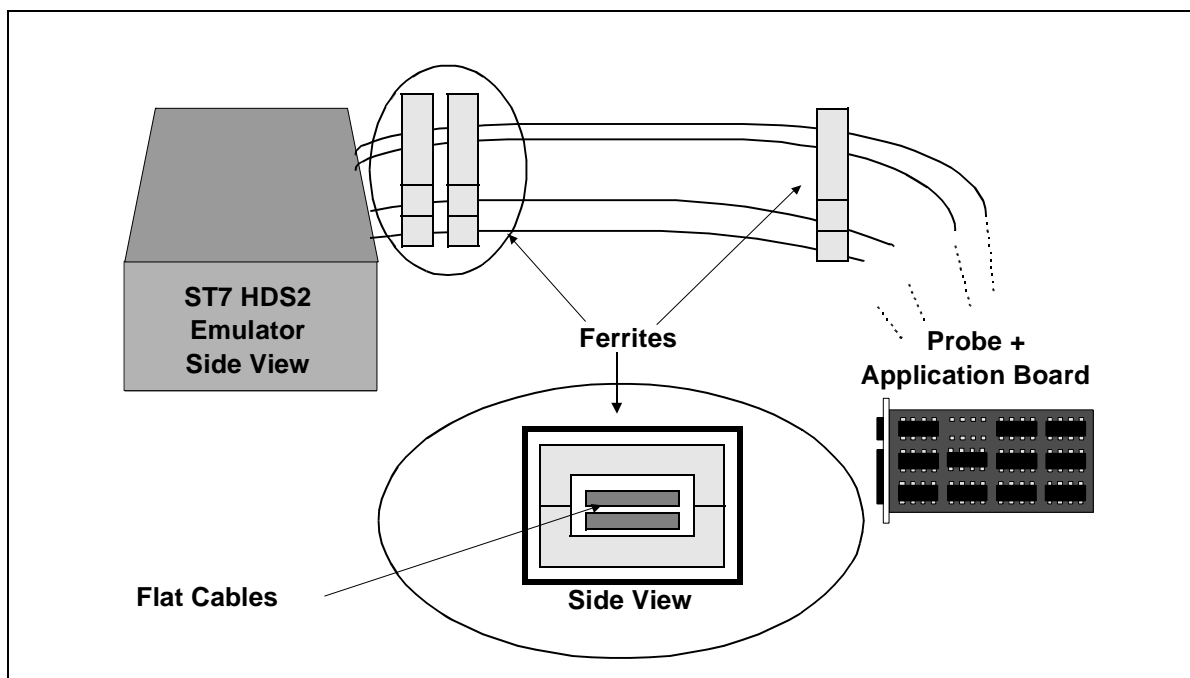


Figure 5: Making your Probes EMC-Compliant

Step 3: Connecting the Probe to your Application Board using Device Adapters

Important Notes Concerning ST7MDT2 devices and their packages

Emulated devices of the ST7MDT2 family are available in several packages as the following table shows.

Each package has its own connection procedure, found on the page cited in the table below:

Devices	Packages	Page No.
ST 72511 R ST 72311 R ST 72512 R ST 72532 R	TQFP64	16
ST 72314 N ST 72334 N	TQFP64 SDIP56	16 18

Devices	Packages	Page No.
ST 72314 J	SDIP42	18
ST 72334 J	TQFP44	Probe not provided. See <i>page 63</i> .
ST 72124 J		

Note the following constraints:

- **The TQFP44 package is not supported by the probe provided in this emulator kit**—refer to *page 63* for more information.
- **Special precautions are required if you are using the TQFP64 package.** Because there is no Yamaichi socket available specifically for the **TQFP64** footprint (a QFP64 Yamaichi socket is furnished instead), there are special footprint precautions to be taken when designing your application board (see *QFP64/TQFP64 & QFP44/TQFP44 Footprint Issues* on page 67). In addition, two TQFP64 device adapters are provided in this emulator kit. The first (ref.: DB389) allows you to connect the emulator (i.e. the ST7MDT2-Active Probe) to your application board. The second (ref.: DB379) allow you to connect an actual MCU device to the Yamaichi socket (i.e. without soldering the MCU directly to the footprint)—see *Using the TQFP64 Device Adapter (Ref.: DB379)* on page 20).

A) If you are using the TQFP64 package, proceed as follows:

- **If you have already designed your application board using a true TQFP64 footprint**, see the troubleshooting table entry entitled *QFP64/TQFP64 Incompatibility* on page 64.
- **If you haven't yet designed your application board**, and want to know how best to do so, see the section entitled *QFP64/TQFP64 & QFP44/TQFP44 Footprint Issues* on page 67.
- **If you have designed your application board using a hybrid QFP64/TQFP64 compatible footprint** (how to do this is described in *QFP64/TQFP64 & QFP44/TQFP44 Footprint Issues* on page 67), continue with the following instructions.



Note:

A Yamaichi QFP64 socket and its cover are provided in the package. Before going through the procedure, make sure that the emulator kit you were delivered includes the socket, its cover and its screws and washers.

To connect the ST7MDT2-Active Probe to its TQFP64 device adapter and then to your application, follow these steps (see *Figure 6*):

- 1 Solder the Yamaichi QFP64 socket base onto your application board. Do not use the socket cover (set it aside for future use with an actual MCU).

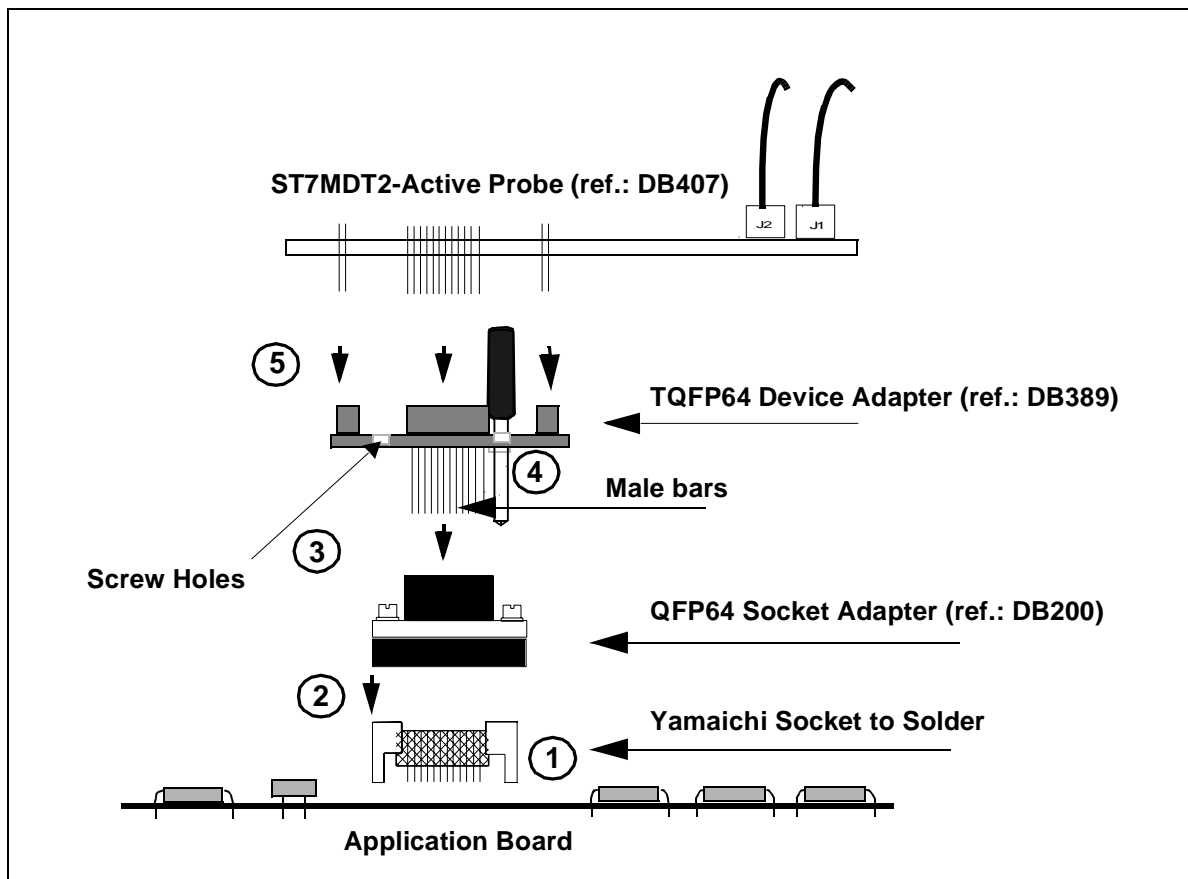


Figure 6: TQFP64 MCU Package Connections

- 2 Place the QFP64 socket adapter (ref.: DB200) upon the socket base, aligning pin 1 of the socket adapter with pin 1 on the socket base. (Pin 1 is indicated by a chamfer on the QFP64 socket adapter and by a little arrow or chamfer on the socket base.)
- 3 Now plug the male bars of the TQFP64 device adapter (ref.: DB389) onto the socket adapter (i.e. into the 2 x 8 pin connector located between the two 2 x 12 pin connectors). There is only one connection scheme. Handle the device adapter and its bars carefully.
- 4 Using four screws, fasten the TQFP64 device and socket adapter assembly onto the Yamaichi socket through the holes located on the upper surface of the TQFP64 device adapter.

- 5 Once screwed in you can connect the ST7MDT2-Active Probe (ref.: DB407) to the TQFP64 device adapter using the four female connectors on the TQFP64 device adapter and the corresponding male pins on the ST7MDT2-Active Probe. The pins numbered 1 on these two boards must correspond. W1 connectors on both boards must vertically correspond.

If you require supplementary sockets, their commercial reference numbers are given below:

Yamaichi socket WITH positioning pins	QFP/TQFP 64: IC149-064-108-S5
Yamaichi socket WITHOUT positioning pins	QFP/TQFP 64: IC149-064-008-S5

Note: Once your code is debugged, you may want to program some devices and test them in your application without the emulator. Another TQFP64 Device Adapter (ref.: DB379) is included in the emulator kit to allow you to connect an actual MCU to your application without removing the Yamaichi socket. Refer to *Using the TQFP64 Device Adapter (Ref.: DB379)* on page 20.

B) If you are using the SDIP56 or SDIP42 MCU package, use this procedure:

Note: A SDIP56 device adapter and a SDIP56 to SDIP42 adapter are provided in the emulator package. Before going through the procedure, make sure that the package you were delivered includes these items.

To connect the ST7MDT2-Active Probe to its SDIP56 or SDIP42 adapter and then to your application, follow these steps (see *Figure 7*):

- 1 Solder a SDIP56 or a SDIP42 pin socket onto your PCB.
- 2 Plug the SDIP56 adapter (ref.: DB408) into the ST7MDT2-Active Probe (ref.: DB407). Find the W1 connector on both boards. They must correspond vertically.
- 3 **This step is for the SDIP42 adapter ONLY.** Plug the SDIP56 to SDIP42 adapter (ref.: DB326) onto the SDIP56 pin socket of the SDIP56 adapter. Pins 1 of both must correspond. Refer to *Hardware Schematics* on page 69 to find the layout of the adapter.

- Now plug the ST7MDT2-Active Probe and its connected adapter(s) onto your application board.

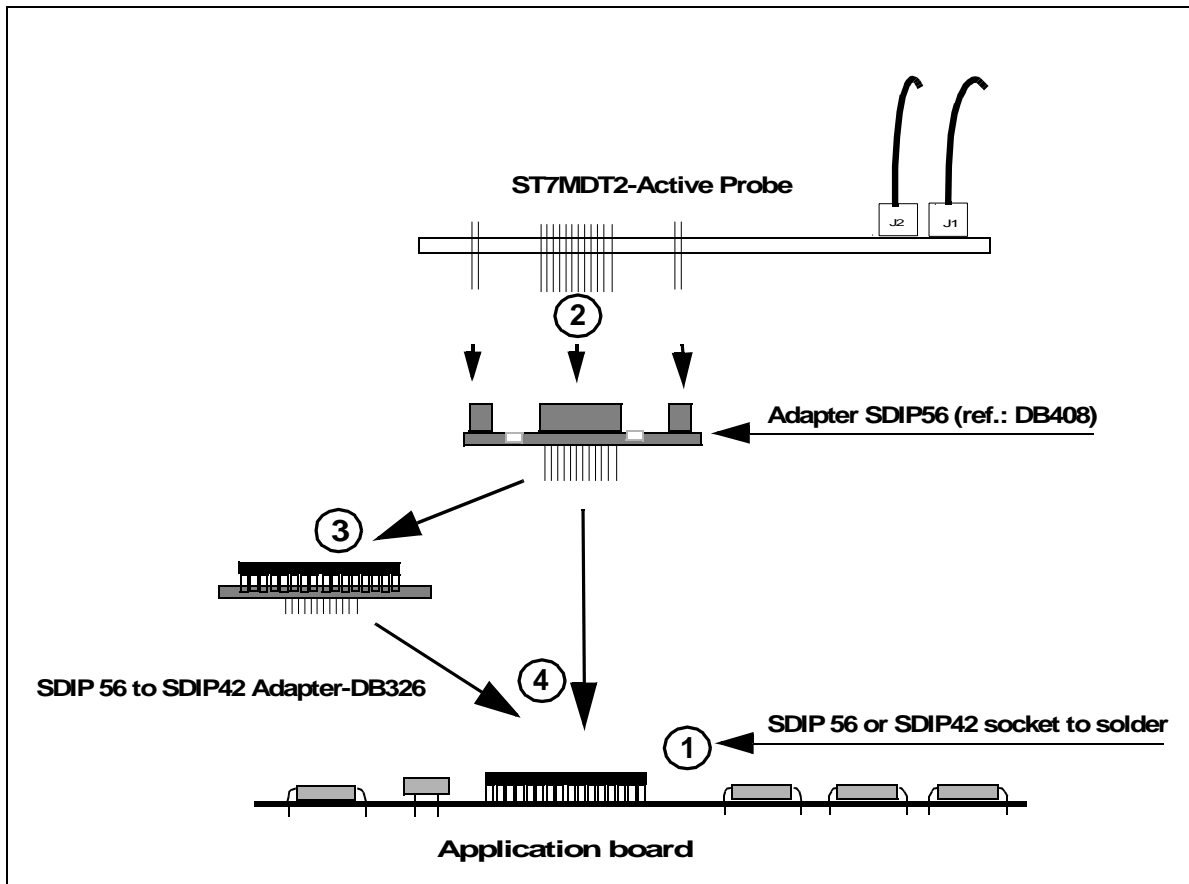


Figure 7: SDIP42 or SDIP56 MCU Package Connections

Step 4: Connecting the power supply

Warning: Make sure that both the ST7MDT2-EMU2B and the application board are powered OFF before making any connections.

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

Mains Voltage Specifications	
AC Voltage	100 V to 240 V

Mains Voltage Specifications	
Frequency	50 Hz to 60 Hz

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 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.

Note: Remember that while your application V_{DD} supply must be in the 3.5 V to 5.5 V range to comply with the actual target MCU, the emulator only supports V_{DD} in the range of 3 V to 5.5 V. For more information, refer to *Emulation Functional Limitations and Discrepancies* on page 59.

2.4 Debuggers Supporting the ST7 HDS2 emulator

The debuggers currently supporting the ST7 HDS2 emulator are:

- ST7 Visual Debug (also known as STVD7) by STMicroelectronics
- Hilight for ST7 (HIWARE)
- HI-WAVE for ST7 (HIWARE)
- Zap for ST7 (COSMIC)

ST7 Visual Debug is free software. It is available on the STMicroelectronics website. See *Product Support* on page 79 for more information

2.5 Using the TQFP64 Device Adapter (Ref.: DB379)

Once your code is debugged, you may want to program some devices and test them in your application without the emulator. This mode is called 'Device Mode'.

However, because the Yamaichi socket furnished is a QFP64 type, you will not be able to insert a TQFP64 device and ensure perfect contact. For this reason, this emulator kit includes a TQFP64 device adapter (ref.: DB379) which can be connected onto the QFP64 socket adapter (ref.: DB200) upper face connectors as follows:

- 1 Remove the ST7MDT2-Active Probe (ref.: DB407) and its device adapter (ref.: DB389) and replace them by the TQFP64 device adapter (ref.: DB379).

- 2 On the upper face of this adapter you will find a low insertion socket for your programmable device.

Note: On the upper face of this adapter you will also find 64 numbered pins to access the device pins voltage. These pins are numbered as they appear on the devices user datasheets. If you have already designed your application board with a real (i.e. non-hybrid) TQFP64 footprint, see QFP64/TQFP64 Incompatibility on page 64.

Figure 8 provides a summary of the different hardware configurations depending on your application board QFP64/TQFP64 footprint. On the left are the configurations used when you designed your application board with the compatible footprint. On the right are the configurations when you used the TQFP64 device footprint (incompatible with our emulation socket).

d

Note: If you plan to use epoxy devices, such as EQFP64, please contact STMicroelectronics Microcontroller Development Tools Sales Support.

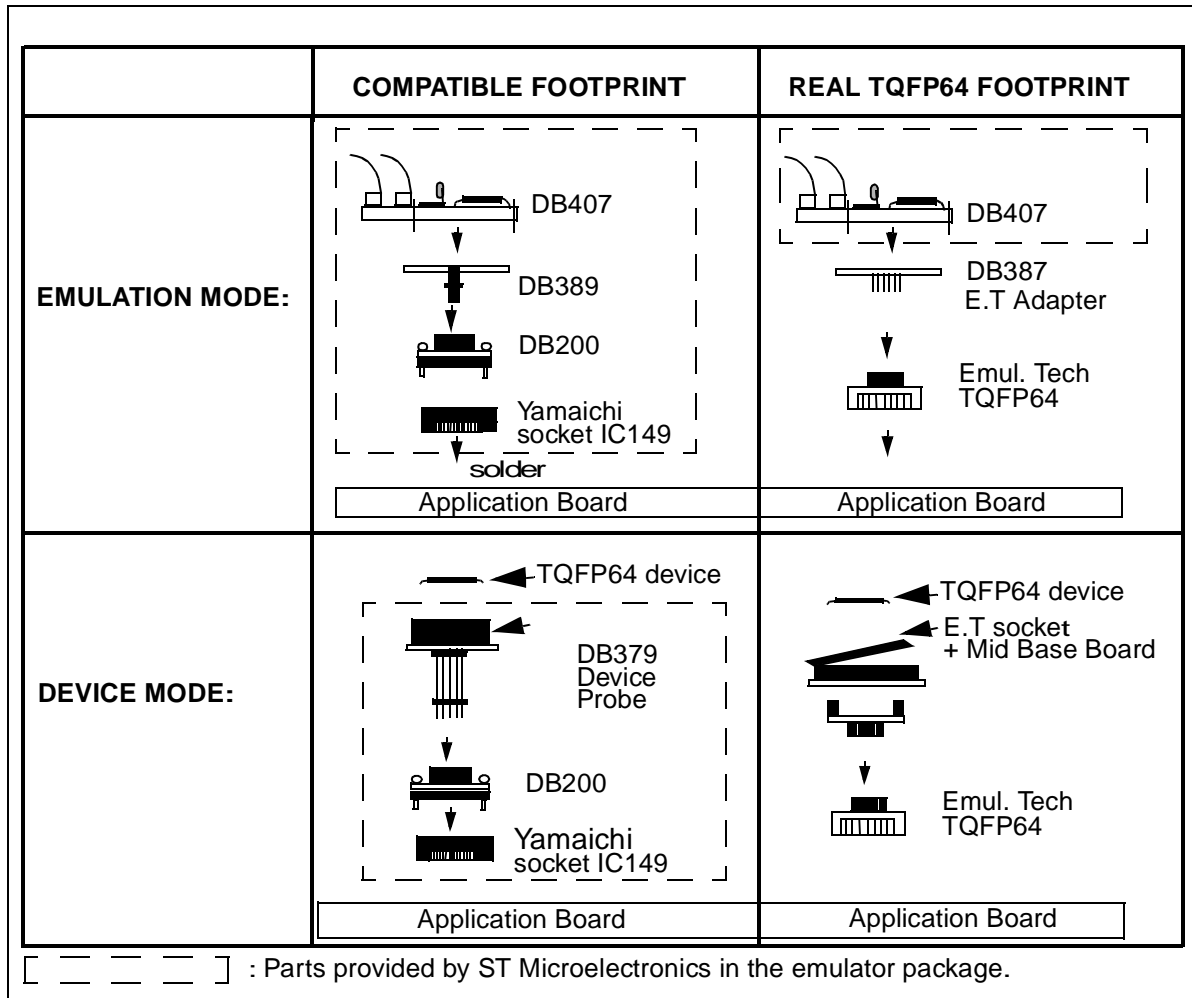


Figure 8: Hardware Configurations for QFP64/TQFP64 Footprints

2.6 Accessing Device Pins

In the event you wish to access the device pins to, for example, monitor pin voltage once your debugging bench is set, you will find a 64-pin connector made up of 4 connectors on the upper face of the ST7MDT2-Active Probe.

These pins are numbered as they appear on the TQFP64 devices user datasheets. The physical location of these connectors is shown in *Figure 9*.

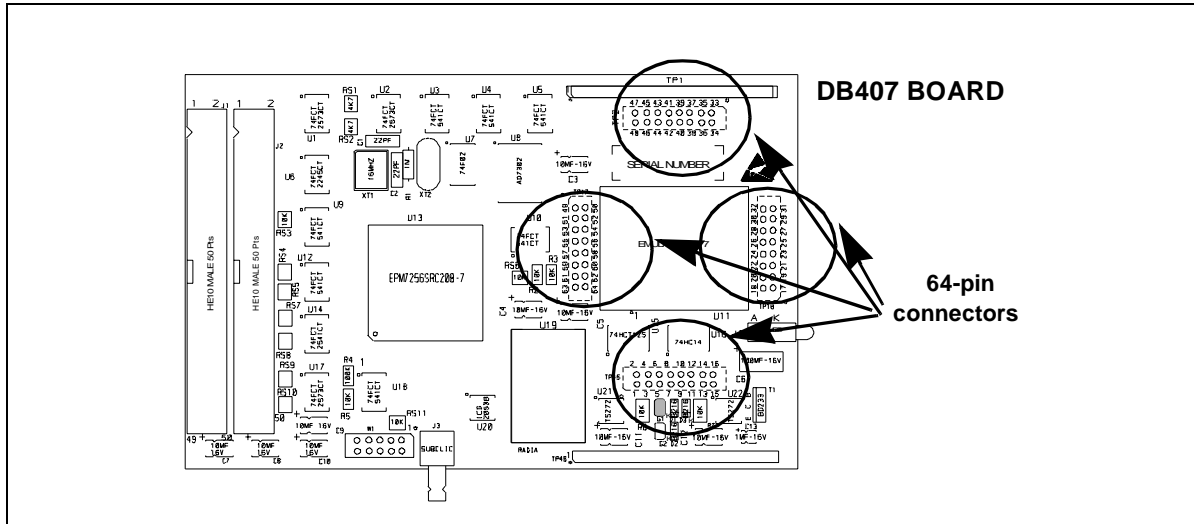


Figure 9: Pin Connector Location

The pin number diagrams on *page 74* shows the correspondence between the 64-pin connectors and their corresponding real pin number in SDIP56, SDIP42 and TQFP44. You can photocopy these diagrams on a cardboard sheet, cut along the dashed edges and place them on the top of the ST7MDT2-Active Probe. You will then find the correct pin numbering for the package.

d

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 application 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 3.9*—how to change your project settings,
- *Section 3.10*—how to save workspaces,
- *Section 3.11*—how to switch from the build context to the debug context,
- *Section 3.12*—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 and Windows NT[®] operating systems.

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.

Note: As a minimum, in order to use 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.

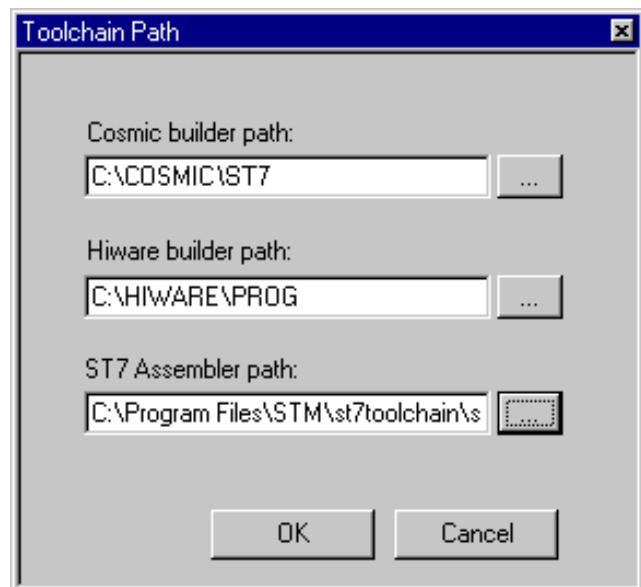
The installation is now complete. You will be prompted to reboot your computer. You should do so before launching STVD7.

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 32. 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 30).

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 29).
- 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 3.11* on page 41).
- 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 file types, project environment and building tools (i.e. linkers and converters):

- The **ST7 macroassembler toolchain** from STMicroelectronics, which generates either `.s19` or `.hex` application files with various intermediate files, such as `.map` or `.lst` files.
- The **Hiware C or Assembler toolchain**, which generates `.abs` application files with various intermediate files, such as `.o` or `.dbg` files.
- The **Cosmic C or Assembler toolchain** which generates `.elf` application 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 application file** (`*.abs`, `*.elf`, `*.s19` or `*.hex` depending on toolchain—refer to *Table 2* on page 31).
- **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 application file.
	.../spim11/spim11.wsp	.../spim11/spim11.bat	Batch file that forces a recompile of application file.
Cosmic	.../c/cosmic/sample.wsp	.../c/cosmic/sample.mak	Recompiles only if the application file has been resaved.
		.../c/cosmic/sample.bat	Forces a recompile of application file.
Hiware	.../c/hiware/sample.wsp	.../c/hiware/build.mak	Recompiles only if the application file has been resaved.
		.../c/hiware/rebuild.mak	Forces a recompile of application file.

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

3.5.1 About application files

The user should verify that the options to include debug information were active during creation of the project files. *Table 2* on page 31 summarizes the way each toolchain functions and lists the different file types (source files, intermediate files and application files) used and produced by the toolchain. The **application 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 Application File:	.s19 or .hex	.abs	.elf
Necessary Intermediate Files:	.map, .lst	.o, .dbg	<NONE>

The **application 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.

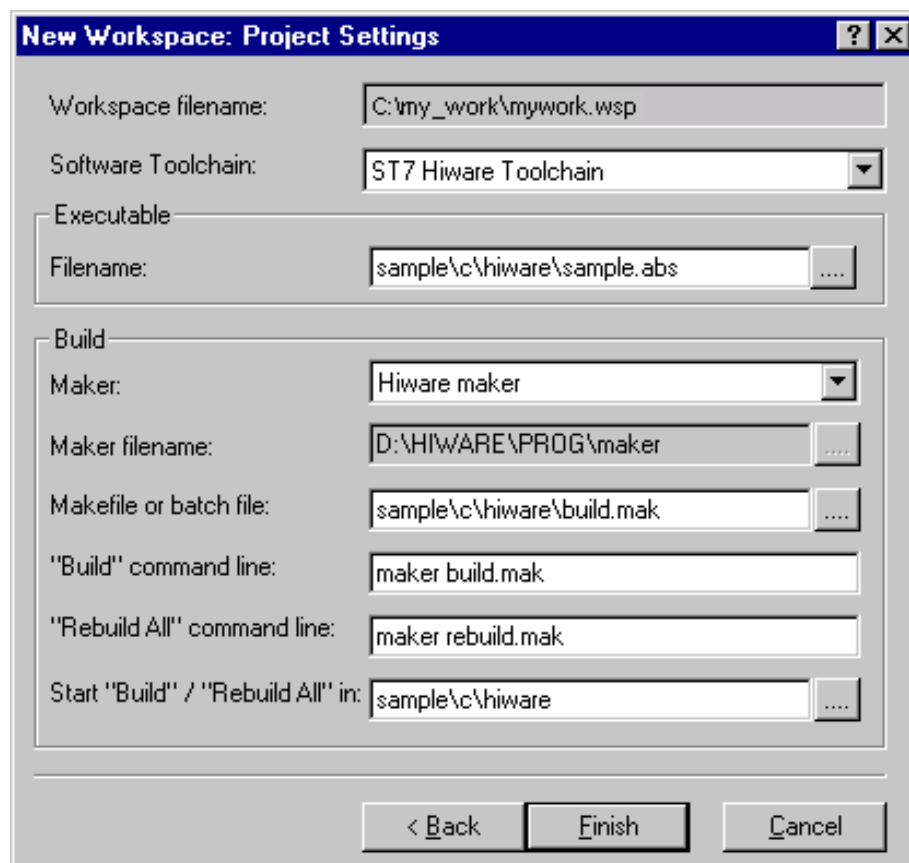
Note: *It is always preferable to have access to all of the files generated by the development toolchain. However, you can load *.s19 or *.hex binary files directly and have limited debugging capabilities (refer to Section 3.8 on page 36).*

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 31)—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 30). 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 3.11* for an explanation of STVD7 contexts), the MCU Configuration window will automatically open to prompt you to choose your target MCU and confirm or modify its option and memory configuration (see *Section 3.12* on page 42).

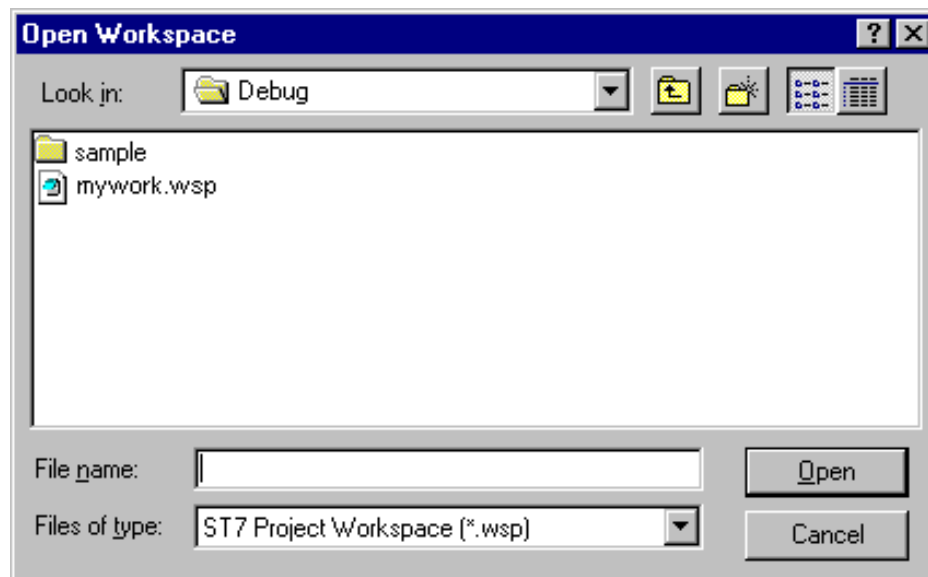
3.7 Opening an existing workspace

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 30.

- 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.

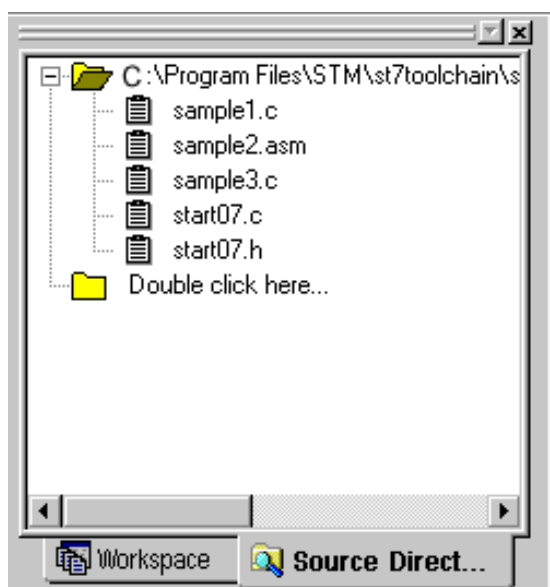
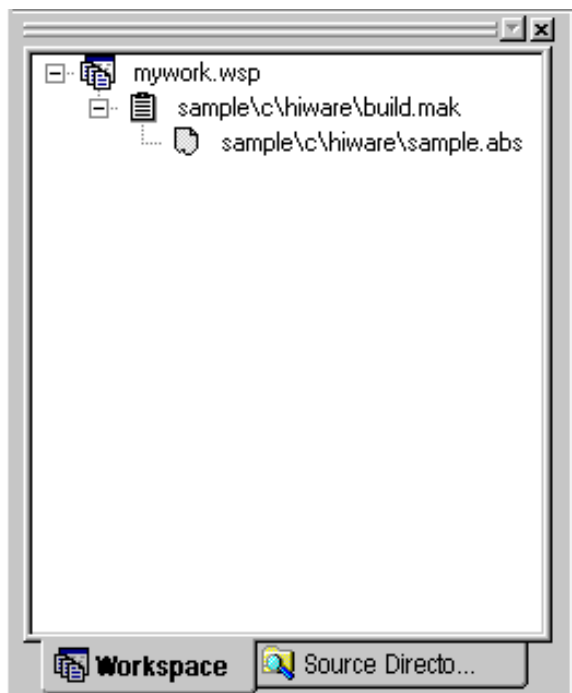


2 The Workspace window opens.

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 application file.

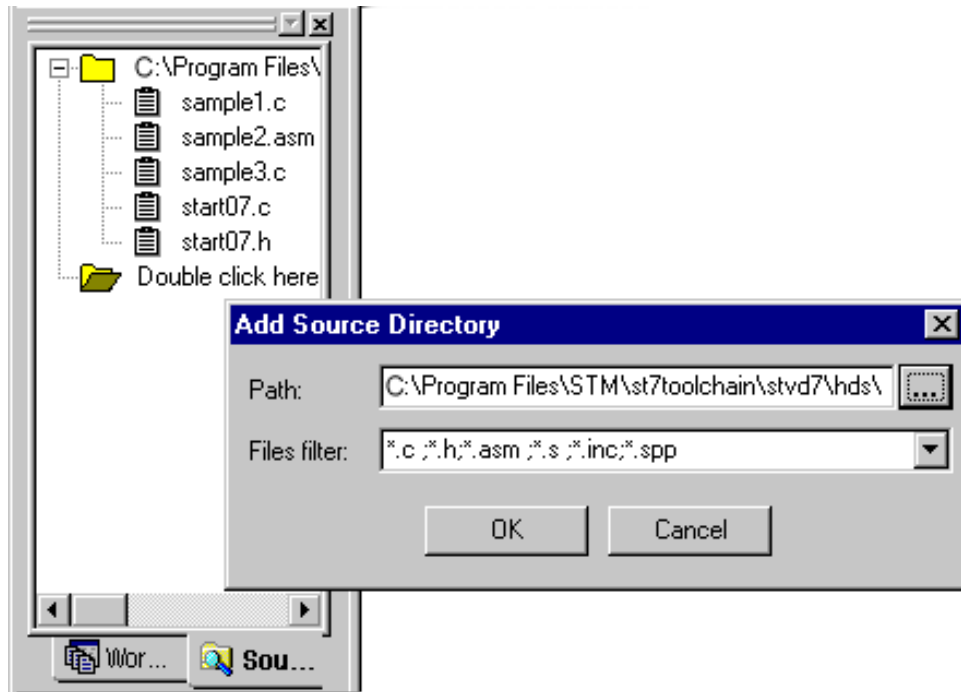
Note: Although the name of the application file is shown in the Workspace window, it has not yet been loaded into the emulation memory—see page 36.

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.



- 4 To load the application 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.

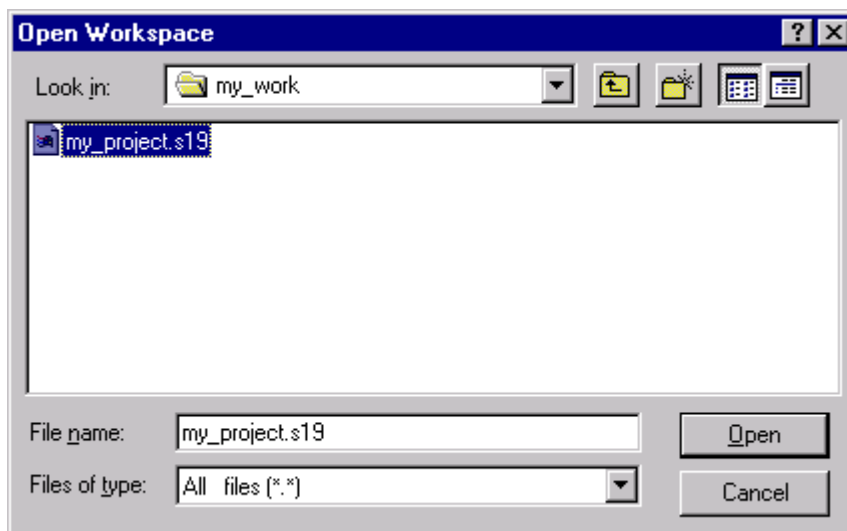
3.8 Opening binary files

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

Note: The range of debugging features available when you open a binary file only will be very restricted. You will only have access to the Disassembly Window.

- 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 **All files (*.*)** in the *Files of type* field.



- 3 Select your binary file (* .hex or * .s19) and click **Open**.

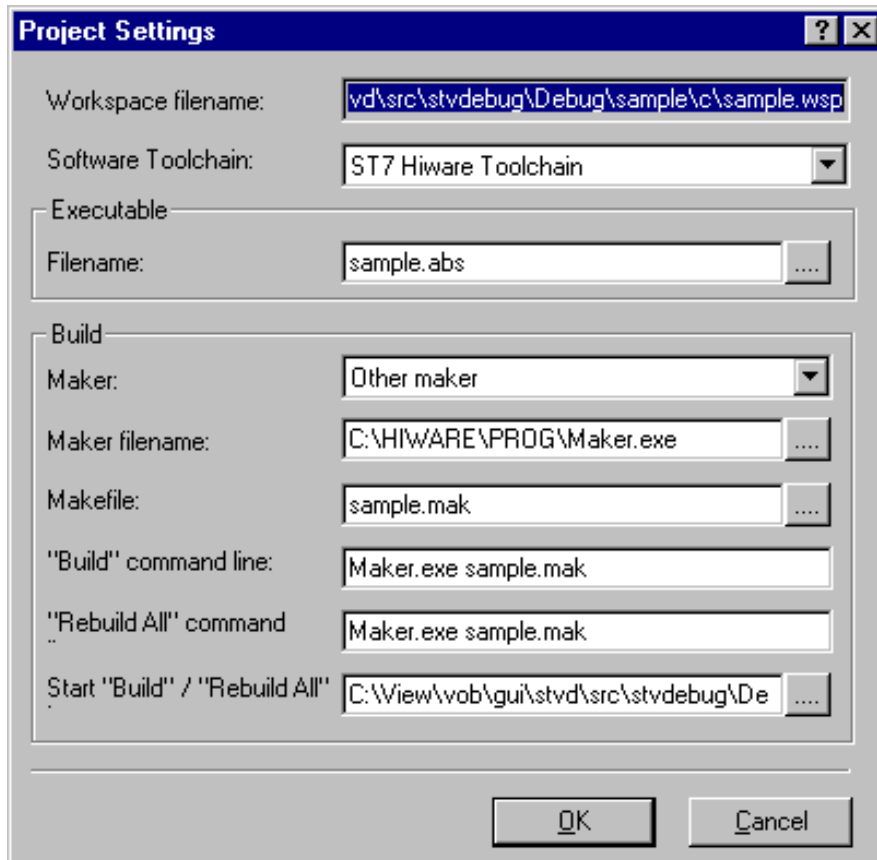
The binary code in the .s19 or .hex file will be loaded into STVD7 and you will be able to access the Disassembly Window. A workspace file (of the same name as the binary file, but with an extension .wsp) will be created automatically.

3.9 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.

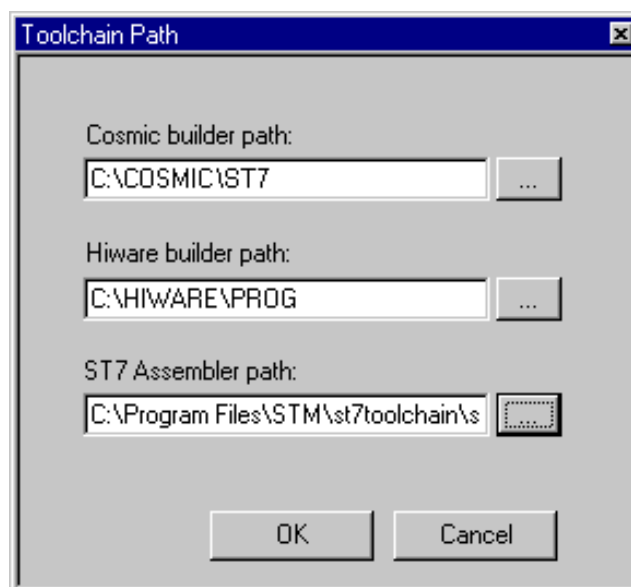



From the main menu, select **Project>Project Settings**.



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.

3.10 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 10 on page 40*), 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.

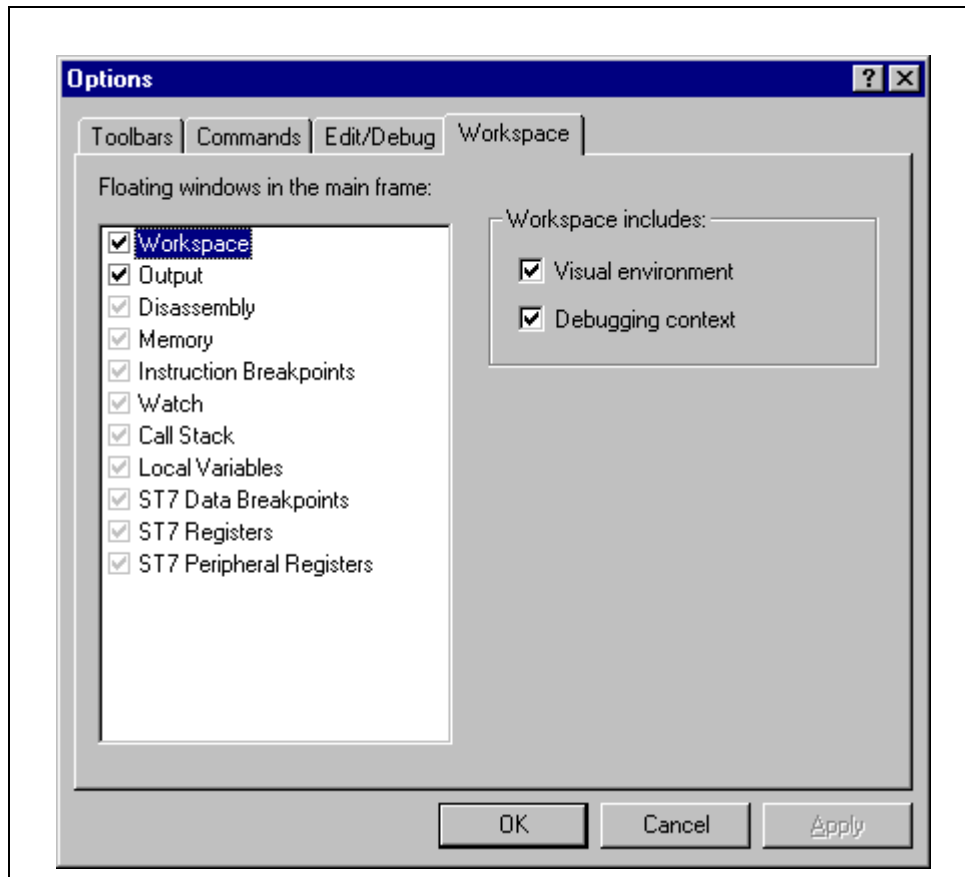


Figure 10: Options window

- 1 Click **Apply** to confirm your settings.
- 2 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 application 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).

3.11 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 3.12* on page 42) and debug the executable file created while in the build context.

3.11.1 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.

3.11.2 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.

3.11.3 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.



3.12 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 3.11.*)

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 11*.

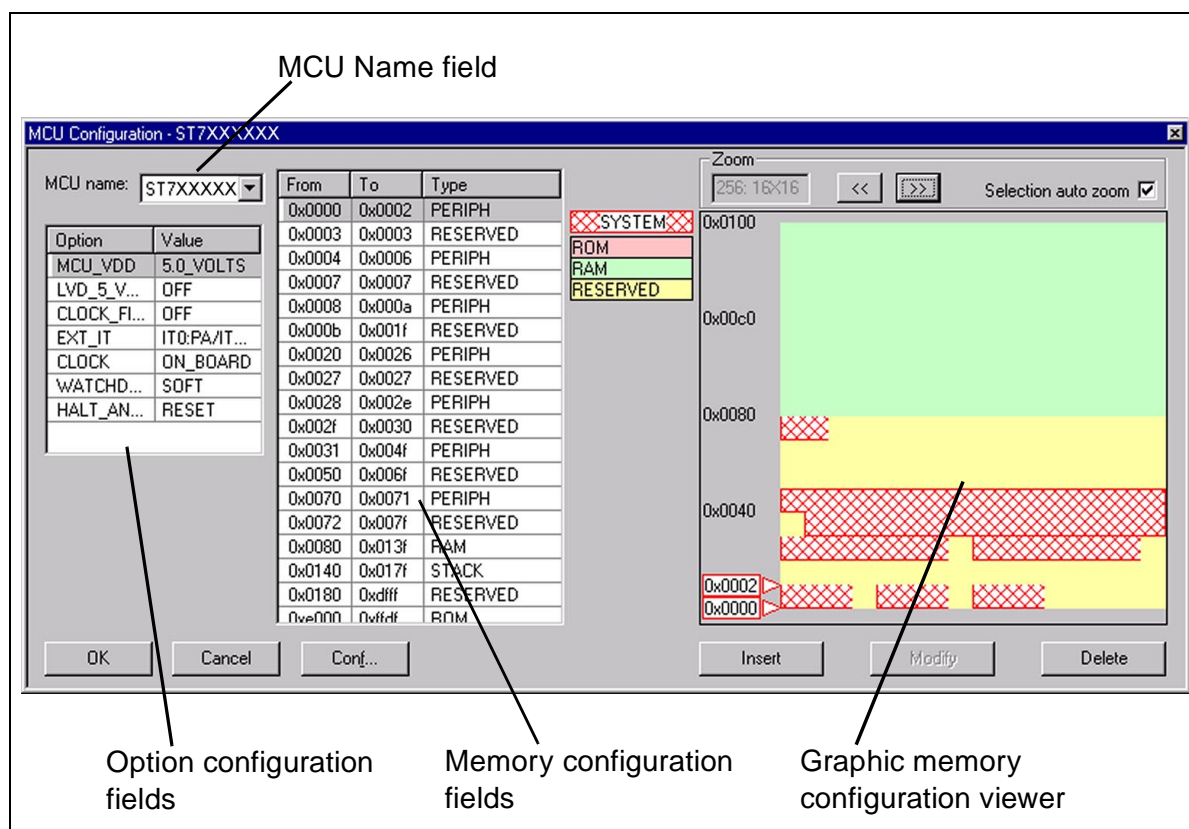


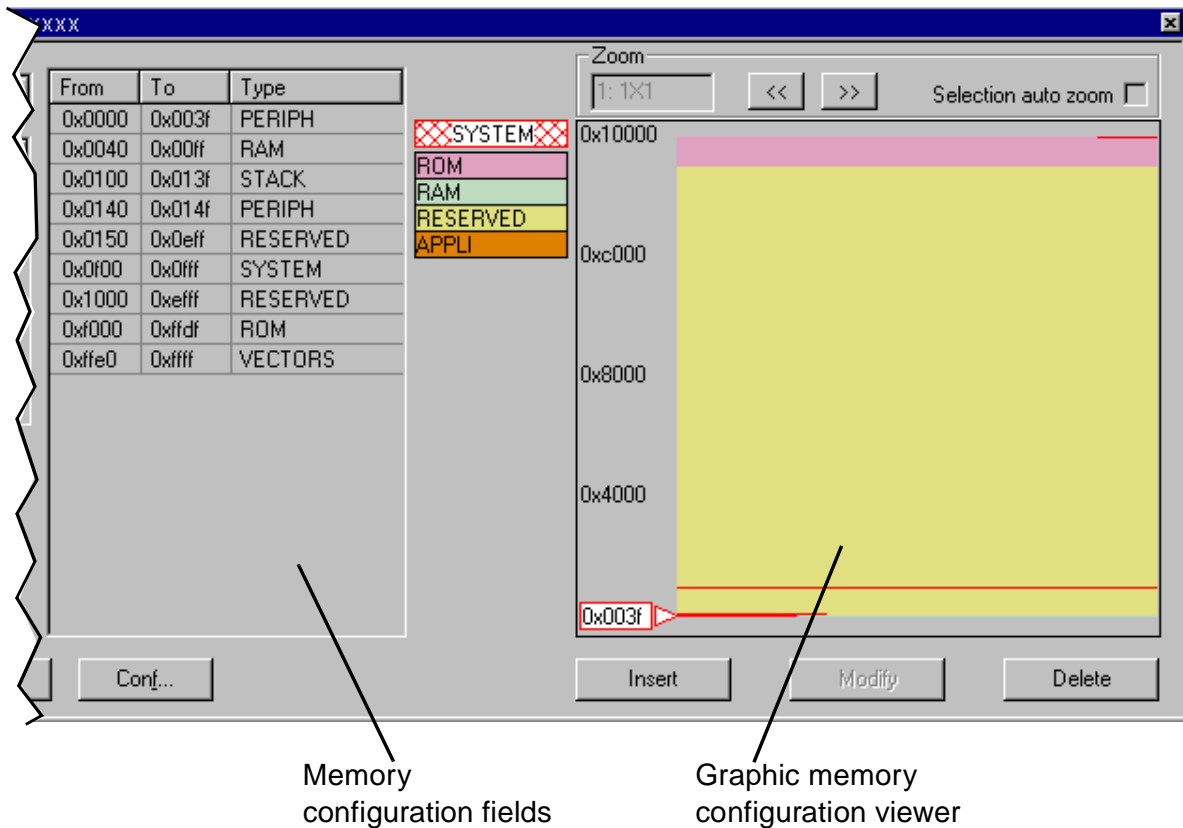
Figure 11: MCU Configuration window

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 46 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:

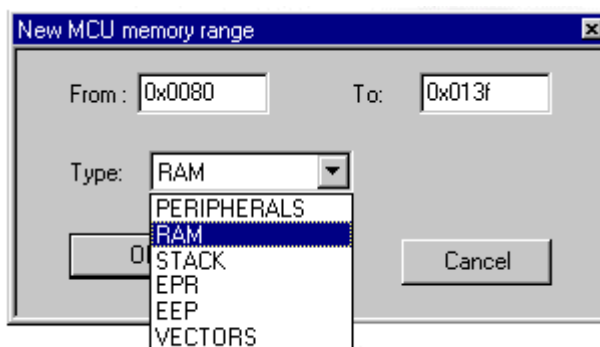
- **Peripherals:** Microcontroller internal or rebuilt peripherals registers. Their properties are defined as in the microcontroller user manual. This memory cannot be modified.
- **RAM:** Random-Access-Memory of the microcontroller. This memory type can be modified.
- **ROM:** Read-Only Memory of the 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 user manual. 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.

3.13 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.



d

4 EMULATOR FEATURES

4.1 Main Features of the ST7 HDS2 Emulator series

The features described below are common to all ST7 emulators:

- Real-time emulation capability (internal frequency from internal 0.5 MHz up to 8 MHz).
- Full memory emulation (up to 64 KB)
- 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
- 1K 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
- Can use as an external clock source either the on-probe oscillator, or an external source via the front panel input
- 4 probe inputs to display application signals in the trace

4.2 Specific Features

The features described below are specific to the ST7MDT2-Active Probe:

- Clock source selection.
- The application power supply follower allows this emulator to run with application V_{CC} from 3 V to 5.5 V. When the probe is not connected to an application board or if the application board isn't powered (application $V_{CC} < 3$ V), the V_{CC} default value is 3 V.
- An external LVD management cell is included.

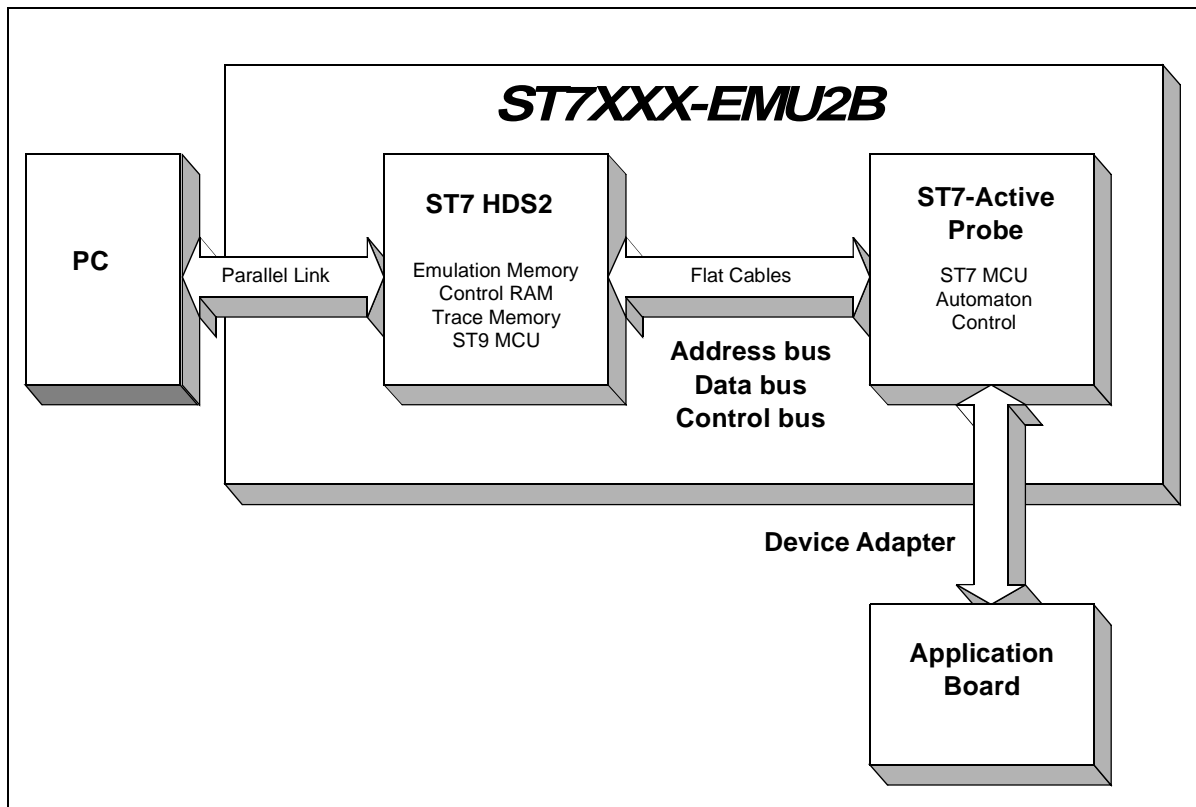
4.3 Emulator Architecture

The ST7MDT2-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 **ST7MDT2-Active Probe** contains the specific resources for the emulated ST7MDT2 devices and is used as a link between the ST7 HDS2 and your application.

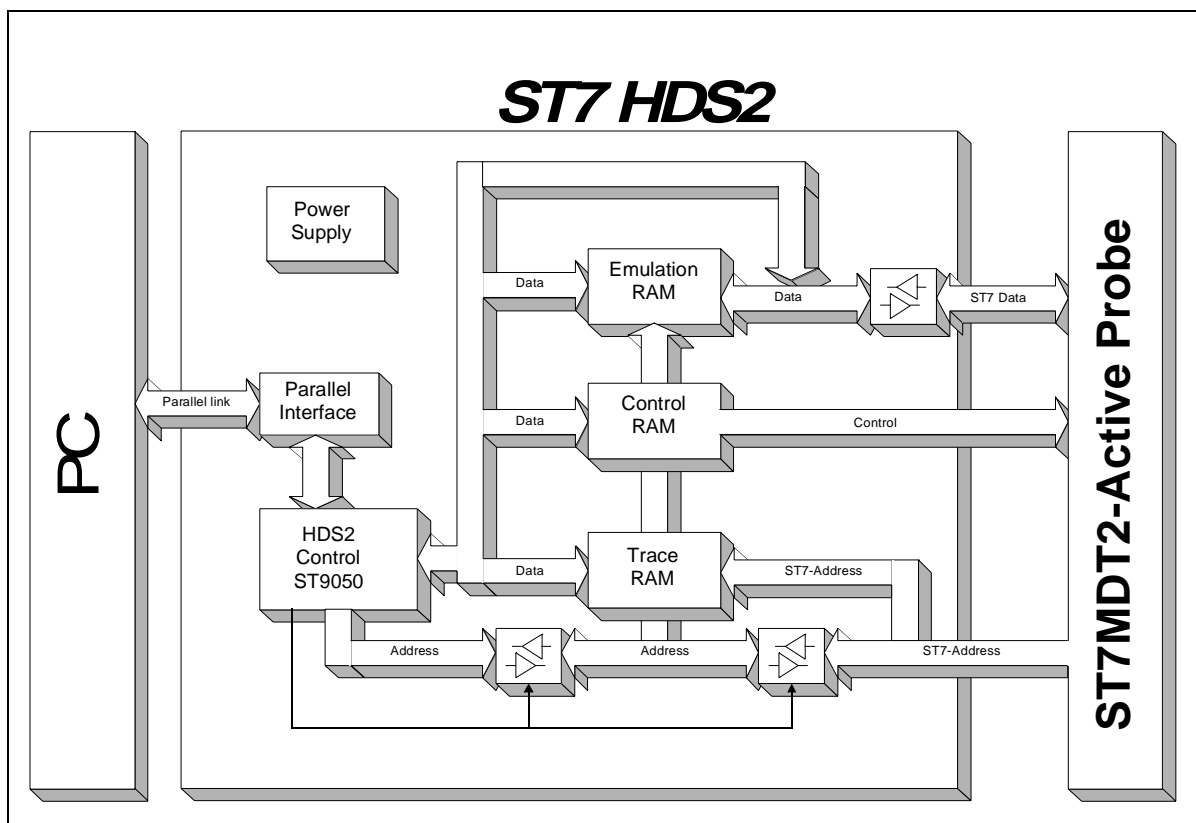


4.3.1 ST7 HDS2 Hardware

The hardware functions provided by this component are listed below:

- **HDS2 CPU:** Used to control the ST7XXX-HDS board and manage common HDS2 features such as the parallel link with the PC.
- **RAM memory:**
 - 64K bytes for ROM and RAM emulation.
 - 64K bytes as break points control and Mapping.
 - 1K 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.
- **ST7MDT2-Active Probe interface**—3 buses connect the ST7 HDS2 to the **ST7MDT2-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 ST7MDT2-Active Probe hardware-like breakpoint features.



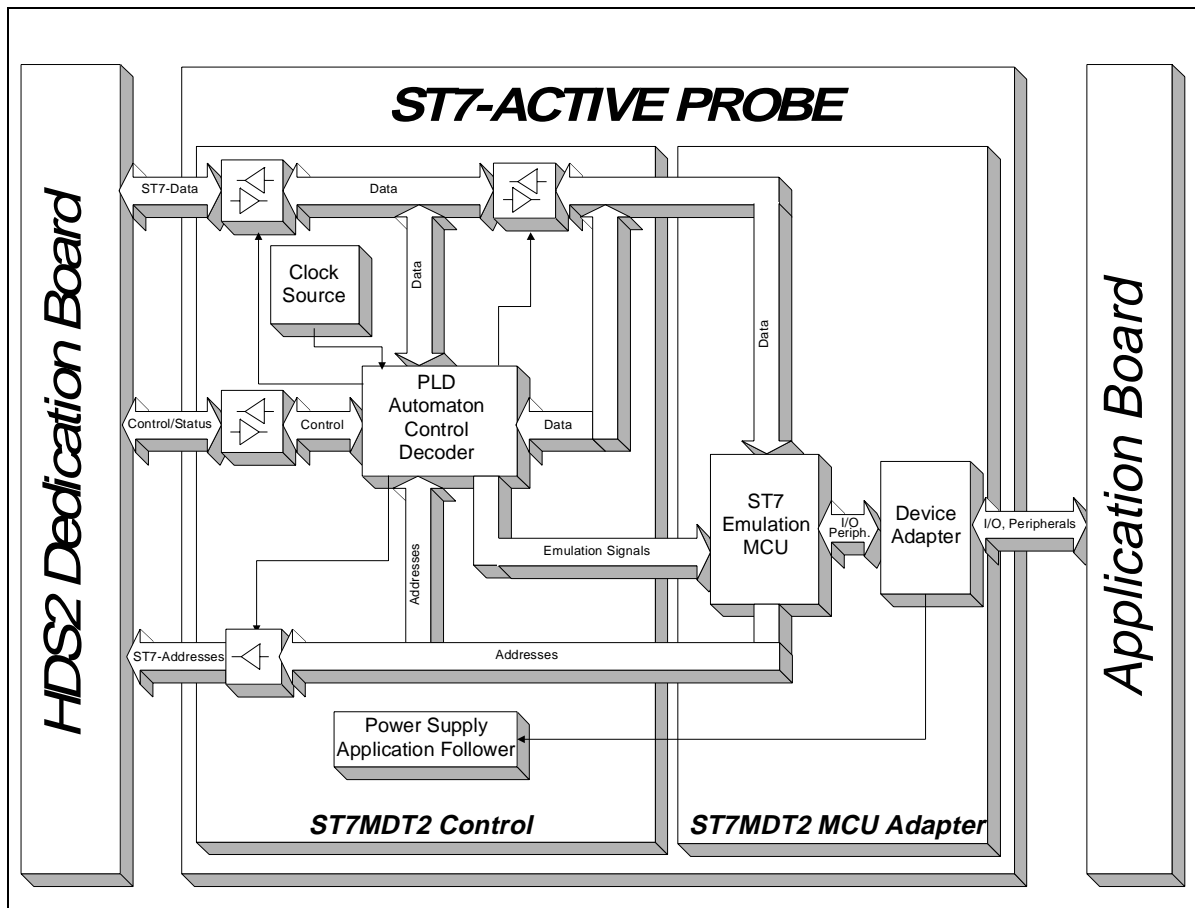
4.3.2 ST7MDT2-Active Probe Hardware

The hardware functions provided by the ST7MDT2-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

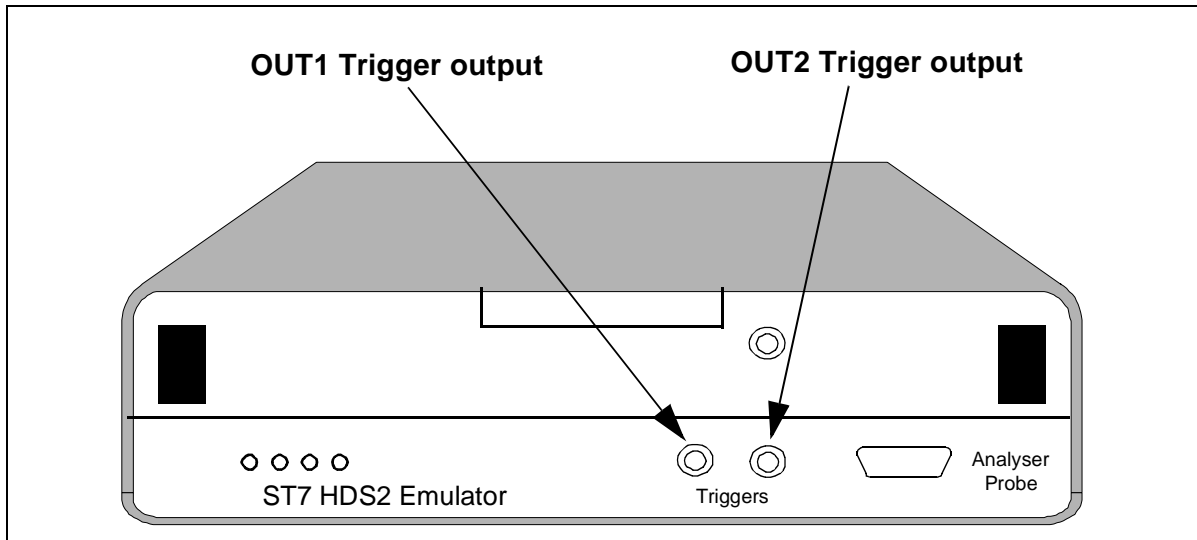
core and gives access to all on-chip peripherals.

- **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 3.5 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.



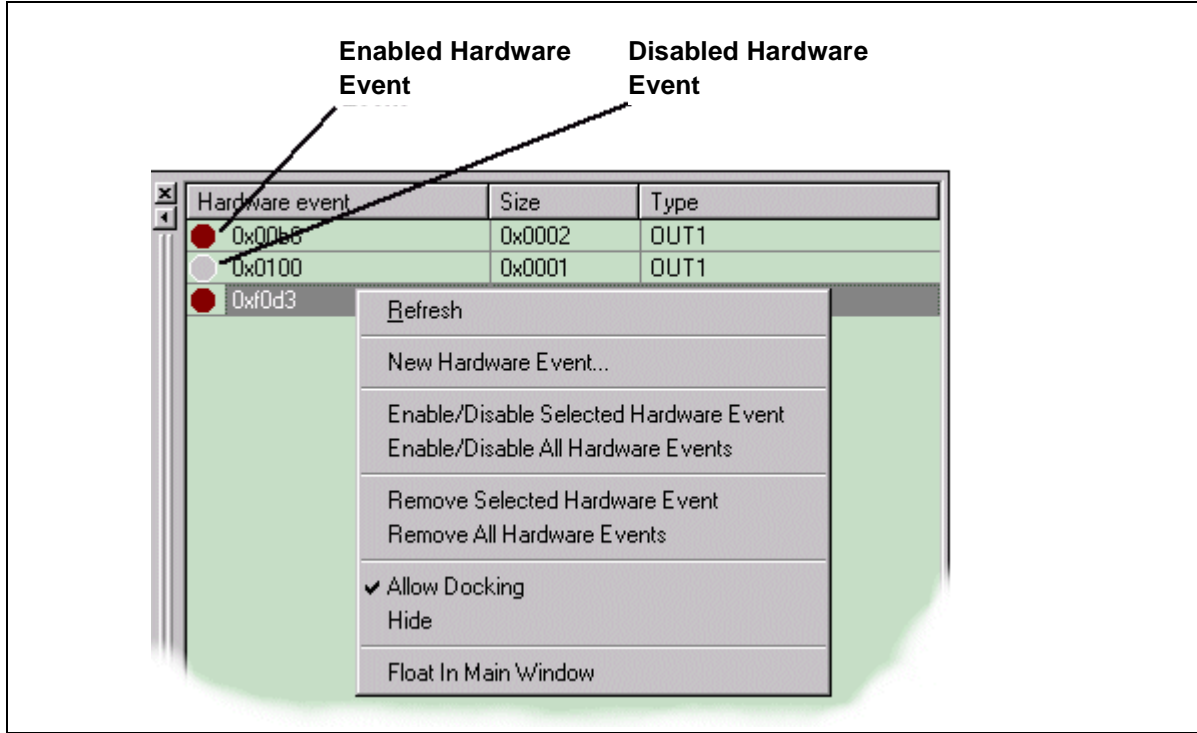
4.4 Output Triggers

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.

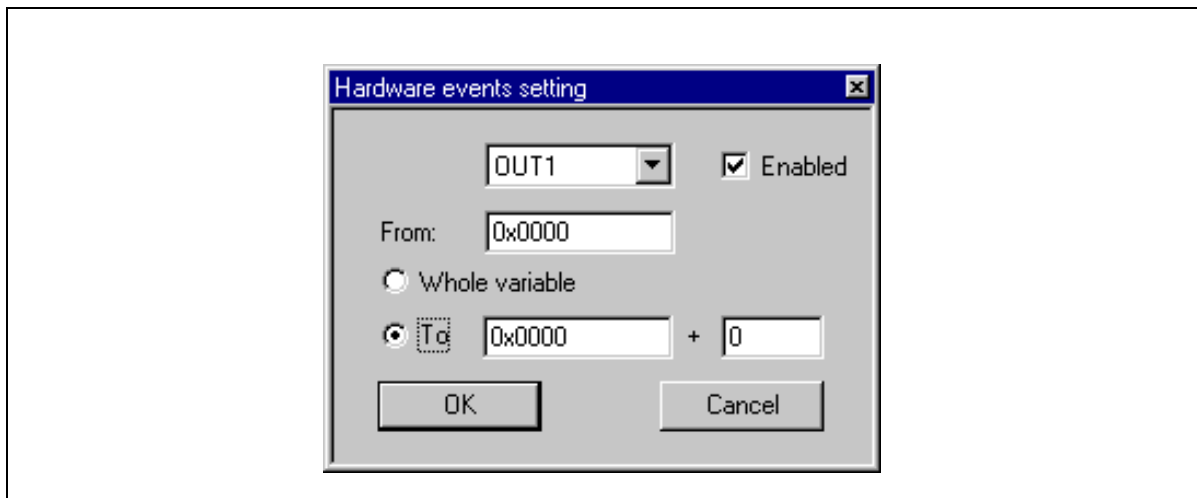


You can program the output signals to these triggers using ST7 Visual Debug:

- 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.

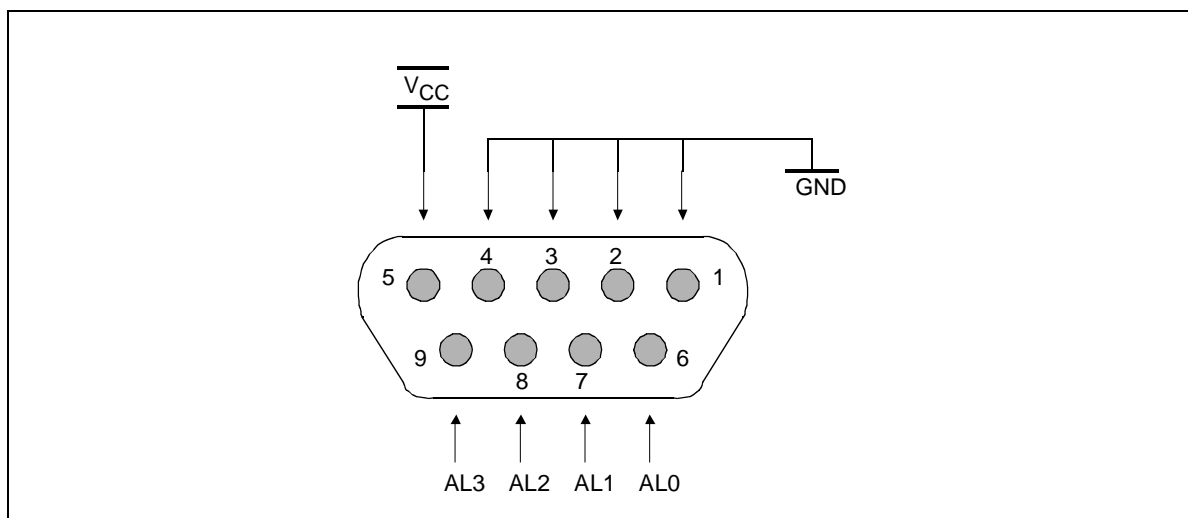


- 4 Choose the trigger output that you wish the signal to be sent to (i.e. OUT1 or OUT2) and check the **Enabled** box.
- 5 You may trigger output signals by setting an event on any of the following:
 - **a whole variable**—creating an *event for synchronization*, which enables you to preset the pulse 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.

4.5 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.



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 cable will also be delivered 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:

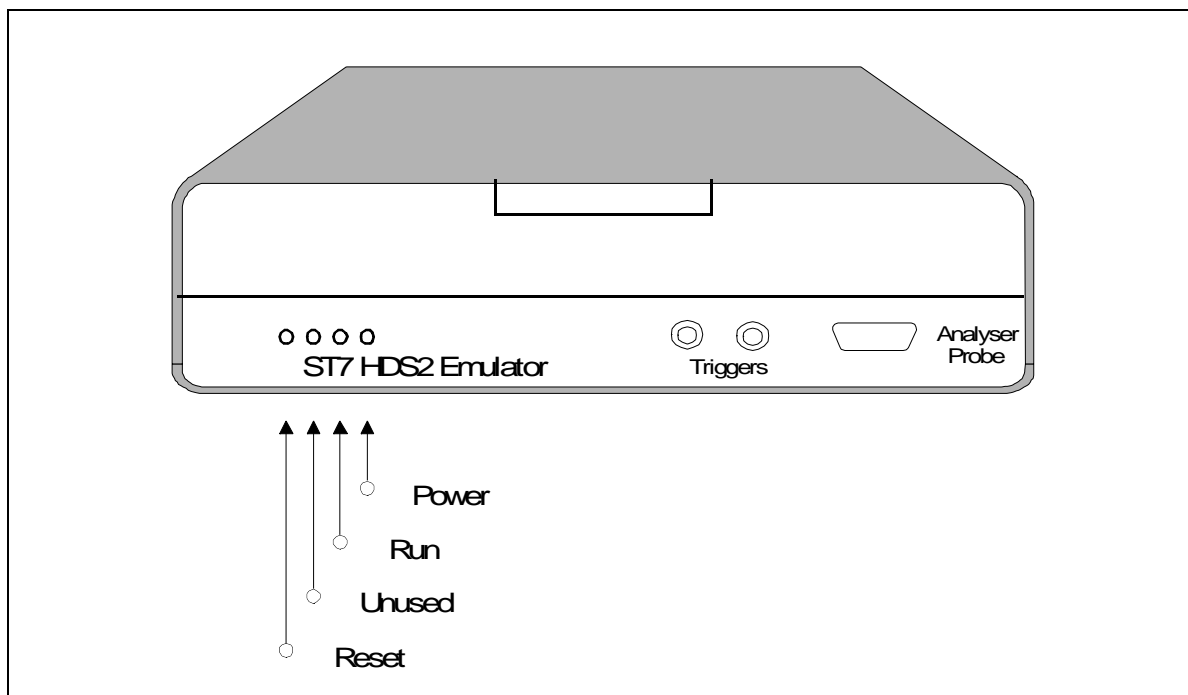
- AL0 is to be taken between the RED (signal) and BROWN (ground) wires.
- AL1 is to be taken between the YELLOW (signal) and ORANGE (ground) wires.
- AL2 is to be taken between the BLUE (signal) and GREEN (ground) wires.
- AL3 is to be taken between the GREY (signal) and PURPLE (ground) wires.

4.6 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.



4.7 On-Chip Peripherals

You can configure certain on-chip peripherals in ST7 Visual Debug's **MCU Configuration** dialog box (refer to *Creating a workspace* on page 32) so that the emulator accurately emulates your target device.

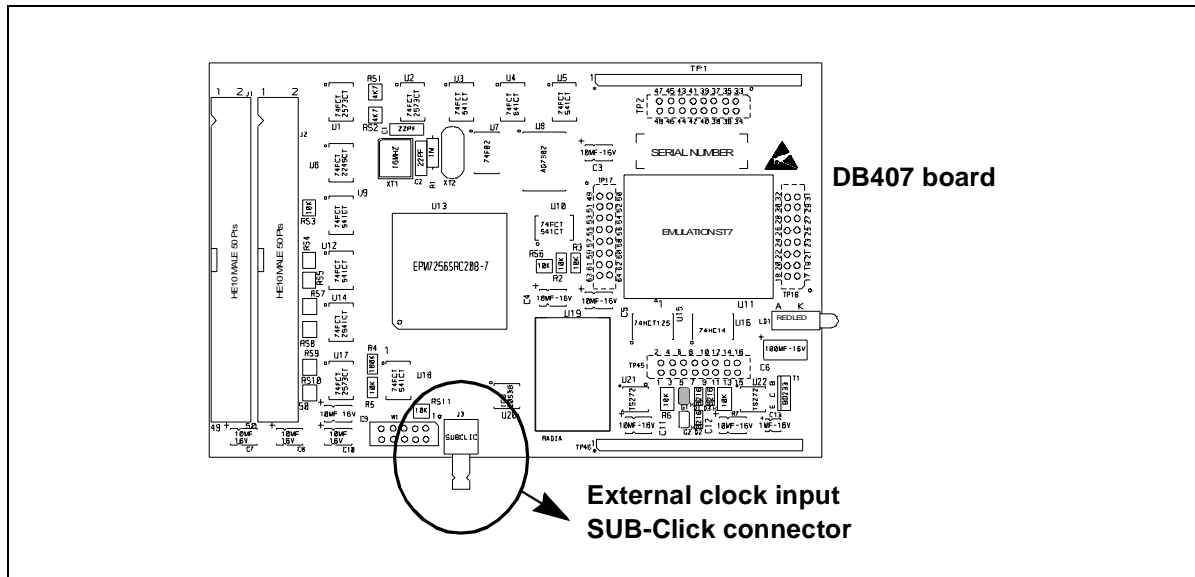
The on-chip peripheral options supported by the emulator are:

4.7.1 CLOCK

The emulator can work with six clock sources (Max. Frequency = 16 MHz, Min. Frequency = 2 MHz):

- **16 MHz** internal oscillator.
- **8 MHz** internal oscillator.
- **4 MHz** internal oscillator.
- **2 MHz** internal oscillator.
- **Application clock** or **Probe** (provided by your oscillator on emulation probe).
- A user-provided **External Clock** input SUB-Click located on the ST7MDT2-

Active Probe as shown below.



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 80. (This list is not exhaustive.)

The oscillator on the probe does not take into account the application circuitry but only the Probe circuitry. So, you must insert its quartz into the XT1 reserved location between the C1 and C2 capacitors.

Note: If you want to use a 'custom' clock oscillator located on your board, you must connect the same oscillator on the probe. After this, you must choose the **PROBE** clock as clock source.

Refer to *Emulation Functional Limitations and Discrepancies* on page 59 for information on the application clock available on your emulator.

4.7.2 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.

4.7.3 WGD+HALT

Allows you to determine if a reset is generated when entering HALT mode while the watchdog is active. There are two options: **Reset**, where a reset is generated when entering Halt mode if the watchdog is active, and **No Reset**, where no reset is generated when entering Halt mode.

4.8 Emulation Functional Limitations and Discrepancies

The following is a list of functional limitations and discrepancies between certain features of the ST7MDT2-EMU2B emulator and its actual target devices:

4.8.1 Power Supply

The application supply follower allows this emulator to run with an application V_{DD} ranging between **3 V** to **5.5 V**. If the application isn't powered, or the $V_{DD} < 3$ V, the power supply is maintained at 3 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.

4.8.2 Low Voltage Detector Management

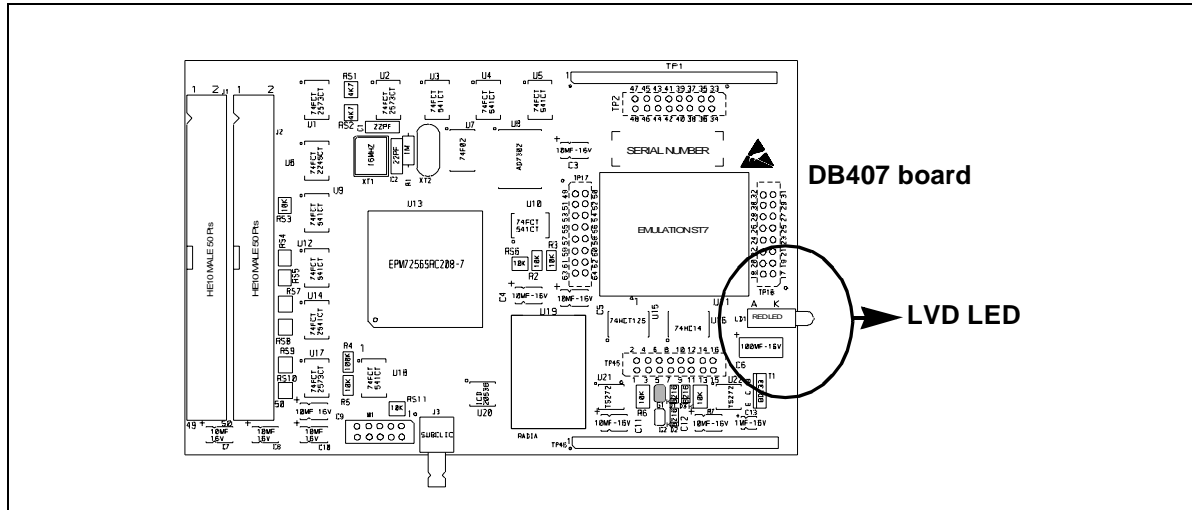
The Low Voltage Detector (LVD) of the emulator is slightly different than the one embedded in the emulated devices. Since the emulation device is always supplied by a voltage ranging between 3 V to 5.5 V, the LVD cell detects a fall in the **application voltage** and resets the emulation microcontroller and the application.

To enable this functionality, you have to select the **LVD.STATUS** field in the **Micro Configuration** window and toggle the value from **OFF** to **ON**. You can notice that the field **LVD.VALUE** is able to be set for ST72334 devices and derivatives (3 LVD values), and fixed to 4.05 V for ST72511 and ST72311 derivative. A LVD reset will be generated each time the following three conditions are met:

- The LVD.STATUS is ON
- The program is running
- The application voltage has fallen under the LVD.VALUE and the application is considered as supplied.

The emulator will consider that the application is supplied only if its voltage is superior to about 2.5V. A non-supplied application or a non connected application will not trigger a permanent reset.

To indicate that your application voltage is under the LVD.VALUE when LVD is ON, a LED located on the board as indicated on the figure will light up.



4.8.3 CROSS Module discrepancy

The Clock, Reset, Option and Supply Smart module is not exactly the same as in the emulated devices.

For ST72334, the clock security system is not emulated. This means in particular that On Clock Reset and supply register CSSIE and CSSD bits are set to 0 and are not able to be set. Only LVD reset flag and Watchdog reset flag are available.

4.8.4 Non-Maskable Interrupt Management

A non maskable interrupt pin is present on this MDT2 family ST72511 and derivative devices. To be considered by the emulator, your application has to be supplied. The emulator will consider that the application is supplied only if its voltage is superior to about 2.5 V.

4.8.5 Nested / Concurrent Interrupts Buttons

Two modes of interrupt management are possible on this MDT2 family devices: Nested or Concurrent. **Nested** is dedicated to ST72511R, ST72311R, ST72512R and ST72532R devices. **Concurrent** mode is dedicated to ST72334J/N, ST72314J/N and ST72124J devices. See the user datasheets for a detailed explanation of these two modes. As a result the CCR register does not have the same bits meaning. In the STVD7 debugger you can choose the aspect of the CCR register by selecting **View>ST7 Registers** from the main menu. A window is

displayed where you can toggle between the two displaying modes, Nested IT or Concurrent IT.

4.8.6 **Clock Probe**

This clock source is not available on the ST7MDT2-EMU2B emulator.

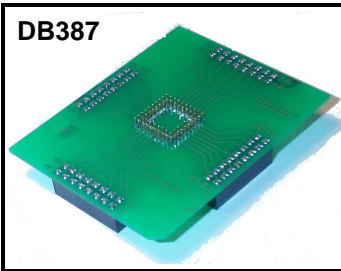
d

©

APPENDIX A: TROUBLESHOOTING

A.1 Identifying the Problem

IF THE FOLLOWING OCCURS:	THEN...
<p>Error Message (when starting the STVD7 for HDS Emulator): “No message received from emulator.”</p>	<p>Ensure that:</p> <ul style="list-style-type: none"> • 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. <p>If none of the above items has been overlooked, this may mean that your parallel port connection needs to be reconfigured.</p> <p>Please refer to <i>Changing the Parallel Port Setup on Your PC</i> on page 64.</p>
<p>Error Messages (when starting the STVD7 for HDS Emulator): “Communication error with EMULATOR board.”</p> <p>or</p> <p>“SYSTEM ERROR DETECTED by EMULATOR BOARD: RESET CPU.”</p>	<p>Ensure that:</p> <ul style="list-style-type: none"> • The flat cables linking the ST7MDT2-Active Probe and the emulator box are properly connected. • The selected configuration file matches the connected ST7MDT2-Active Probe configuration. <p>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 drop-down 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.</p>
<p>You are using the TQFP44 MCU package.</p>	<p>TQFP44 probes are not provided with this emulator kit. If you are using the TQFP MCU package, you can order a compatible probe from the STMicroelectronics Microcontroller Tools Sales Support (see <i>Product Support</i> on page 79).</p> <p>The order number is: ST7MDT2-PB/TQ44</p>

IF THE FOLLOWING OCCURS:	THEN...
<p>QFP64/TQFP64 Incompatibility</p> <ul style="list-style-type: none"> You have already designed your application board and it uses true TQFP64 footprint, so cannot solder Yamaichi QFP64 socket into place and connect emulator probe. You have not yet designed your application board, but wish to use a TQFP64 or TQFP44 compatible footprint. 	<p>Recovery solution in case of problems with TQFP64 footprints</p> <p>In the event you have already designed your application board before receiving this emulator kit and your footprint is a real TQFP64, you will not be able to solder the Yamaichi socket. You can, however, debug your application by using an Emulation Technology TQFP64 adapter (ref.: EPP-064-QF29D-SM). We can provide an adapter board (ref.: DB387, see figure above) for this component, which connects to the ST7MDT2-Active Probe. Ask your dealer or ST sales representative for information about availability (see <i>Product Support</i> on page 79).</p>  <p>Resolve QFP/TQFP incompatibilities by designing your application board using a hybrid footprint. See <i>QFP64/TQFP64 & QFP44/TQFP44 Footprint Issues</i> on page 67 for more details.</p>

A.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 ST7MDT2-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.

A.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 (*Product Support* on page 79).

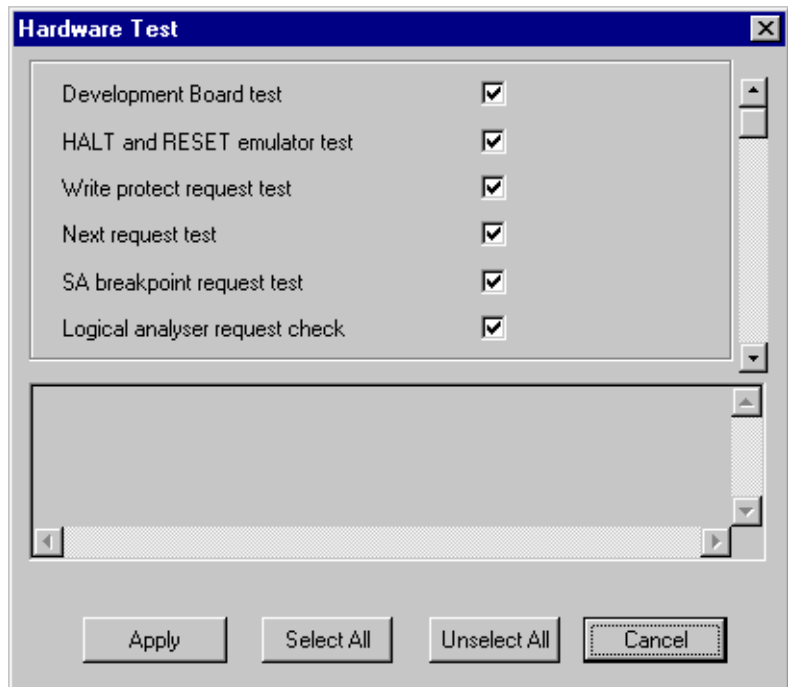
You may open the **Hardware Test** dialog box by:

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

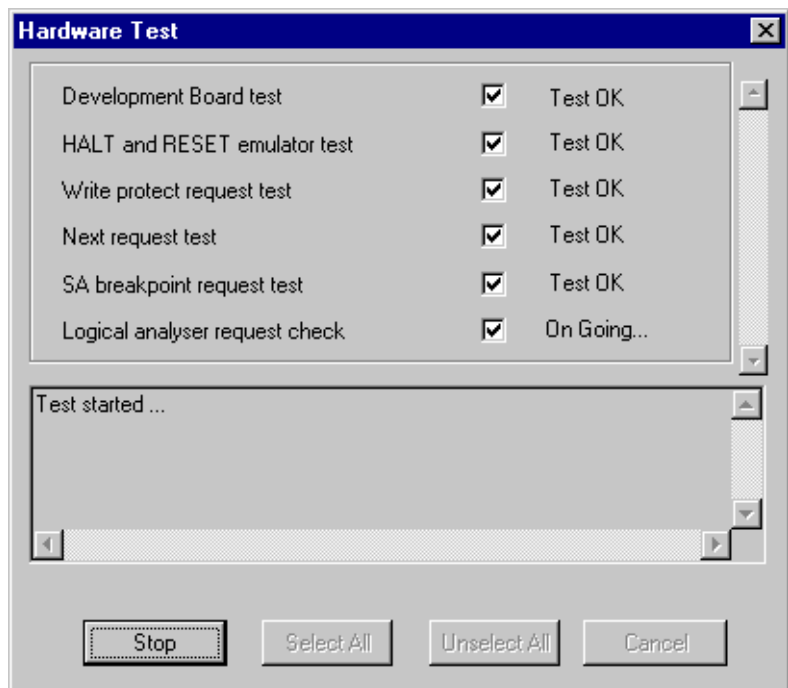
Caution: *Be cautious in performing a Hardware Test on the emulator while an application is open. The opened application WILL BE corrupted by the hardware testing process. If you find that your application has been corrupted, simply close the application, and reopen it.*

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:



A.4 QFP64/TQFP64 & QFP44/TQFP44 Footprint Issues

Some of the emulated devices have a TQFP footprint and require you to solder a Yamaichi socket onto your application board. However, be aware that you may encounter problems owing to the fact that the Yamaichi sockets used to connect the TQFP64 device adapter require footprints that are not compatible with TQFP copper traces.

For this reason, when designing your board, plan to have a double trace compatible with both Yamaichi sockets and TQFP copper traces. Specifications for compatible footprints are provided below. If you use compatible footprints, a single printed circuit board design will serve for both the development stage and the final product. When you are finished the development stage, you can simply replace the development Yamaichi socket by the programmable target device in an actual TQFP64 or TQFP44 package.

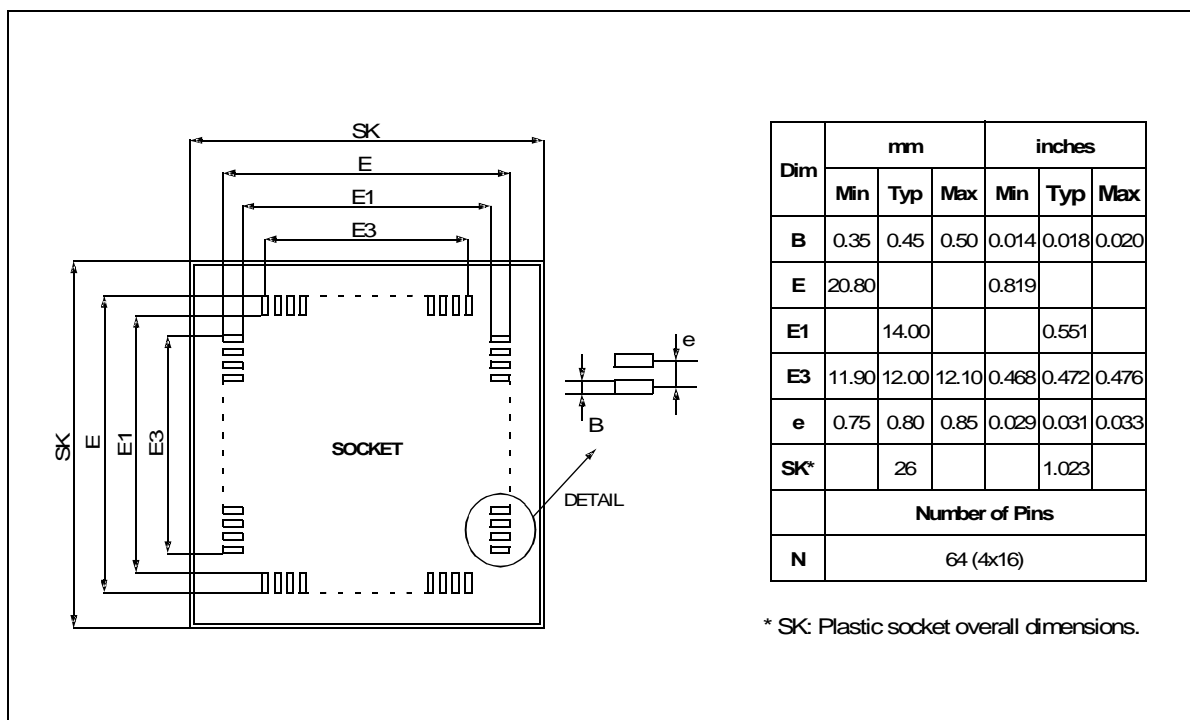


Figure 12: TQFP64 Device and Emulation Probe Compatible Footprint

Note: Remember that TQFP44 probes are not provided in this emulator kit. If you want to use one, you must order it from STMicroelectronics Microcontroller Development Tools Sales Support (refer to Contact List on page 79) using the following order number: ST7MDT2-PB/TQ44.

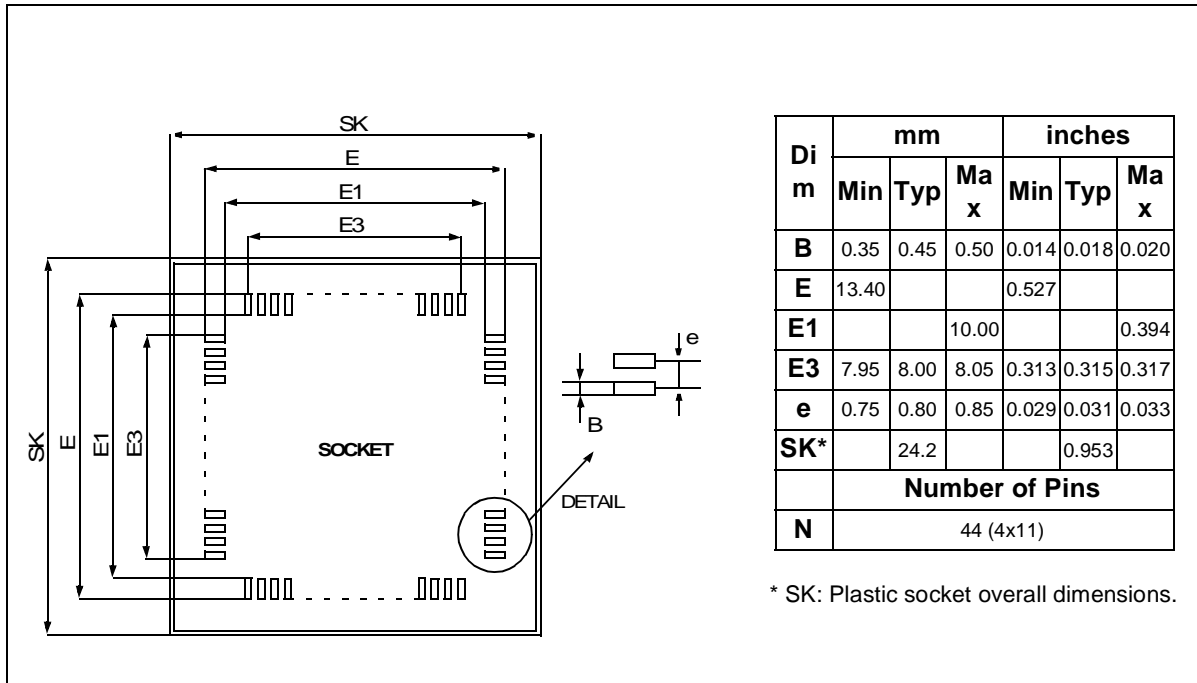


Figure 13: TQFP44 Device and Emulation Probe Compatible Footprint



APPENDIX B: HARDWARE SCHEMATICS

B.1 Component layouts

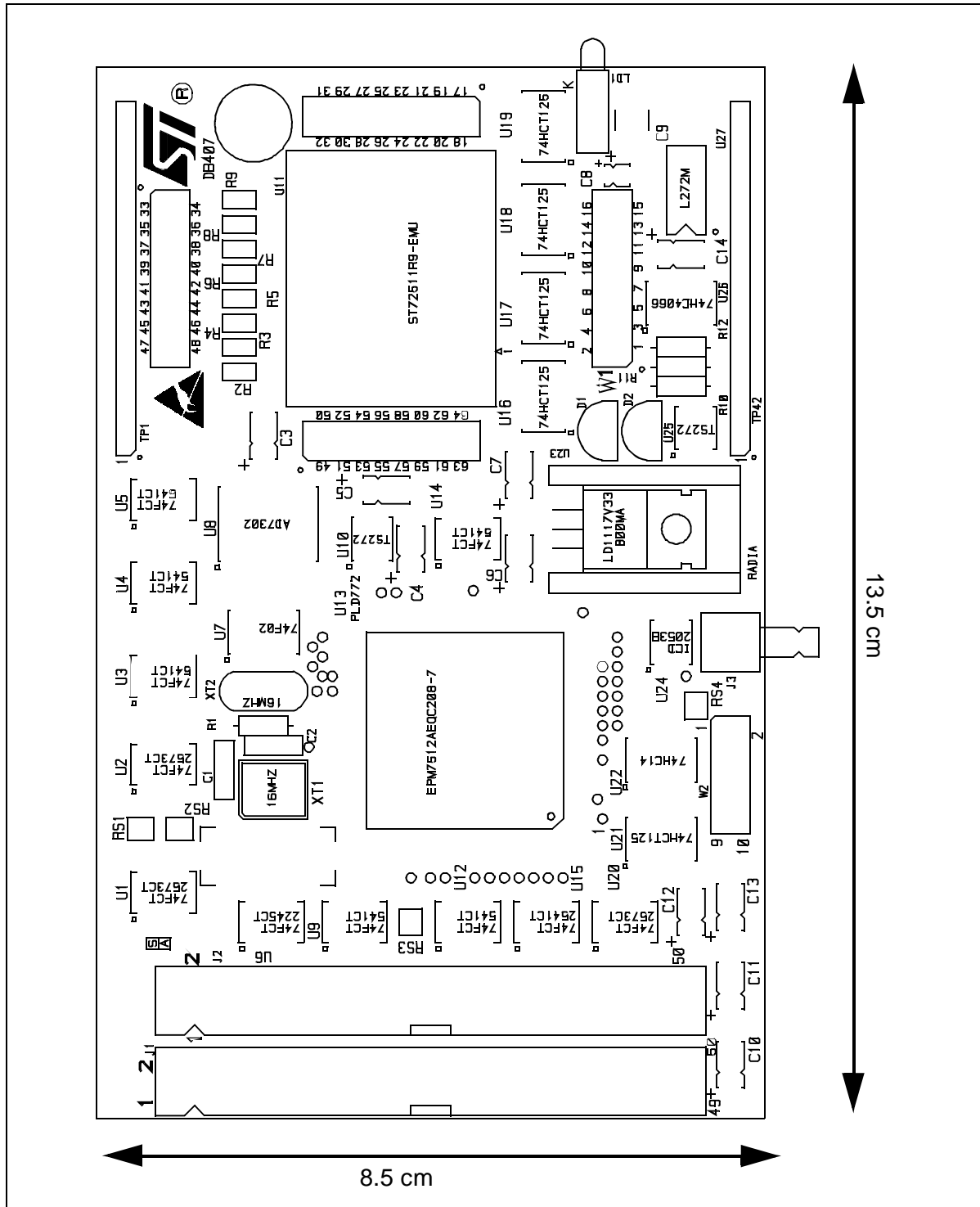


Figure 14: ST7MDT2-Active Probe (ref.: DB407) Components Layout

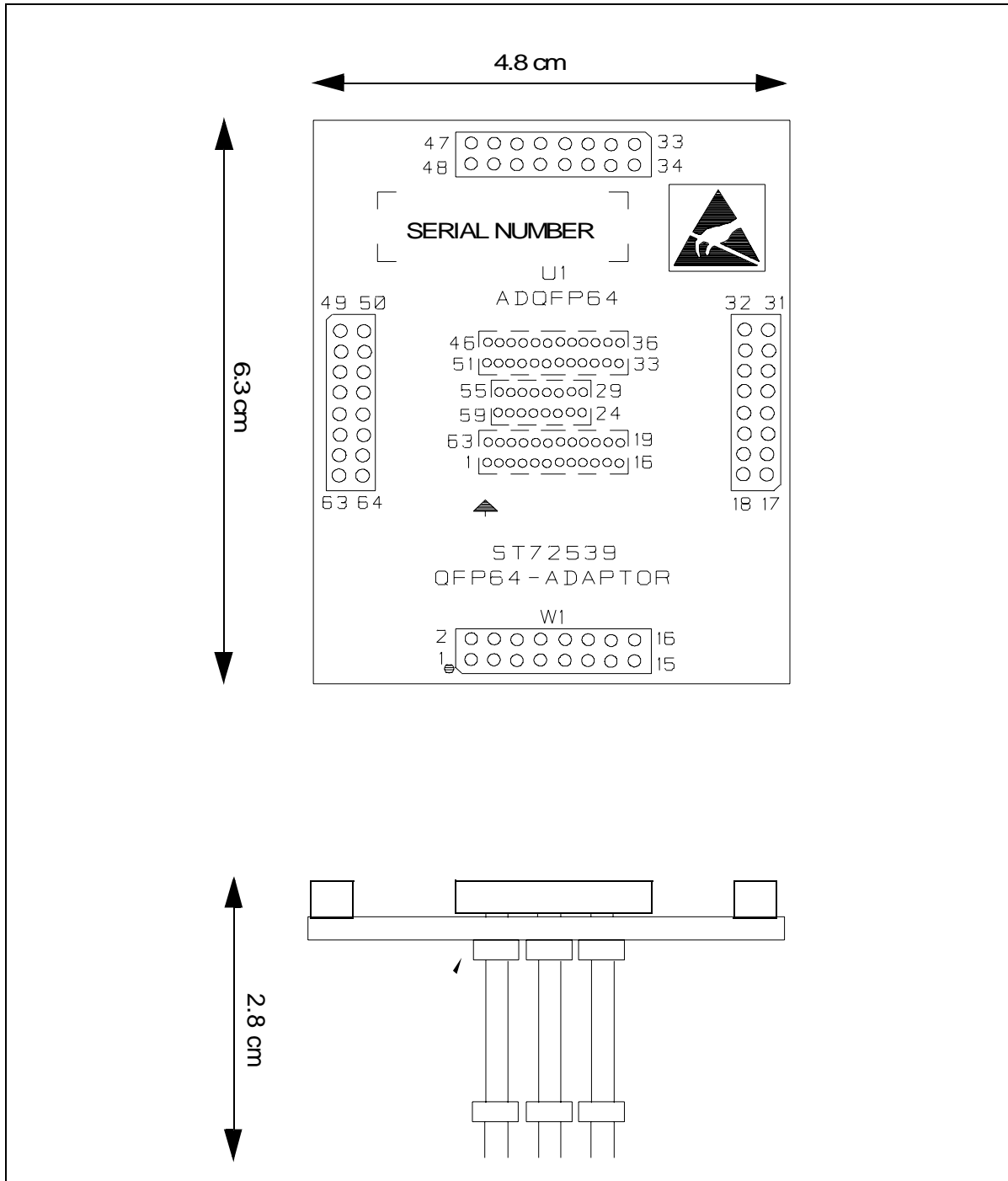


Figure 15: QFP64 Pin Socket Adapter (ref.: DB389) Components Layout

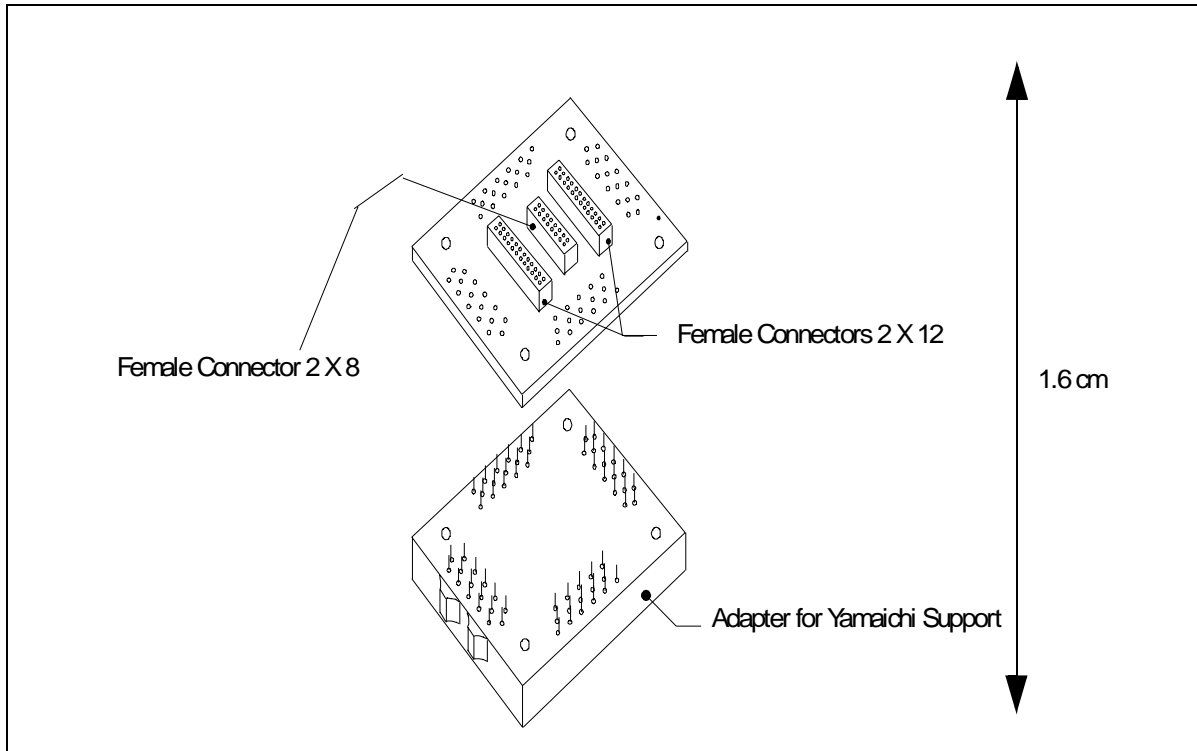


Figure 16: QFP64 Probe Adapter (ref.: DB200) Components Layout

Q

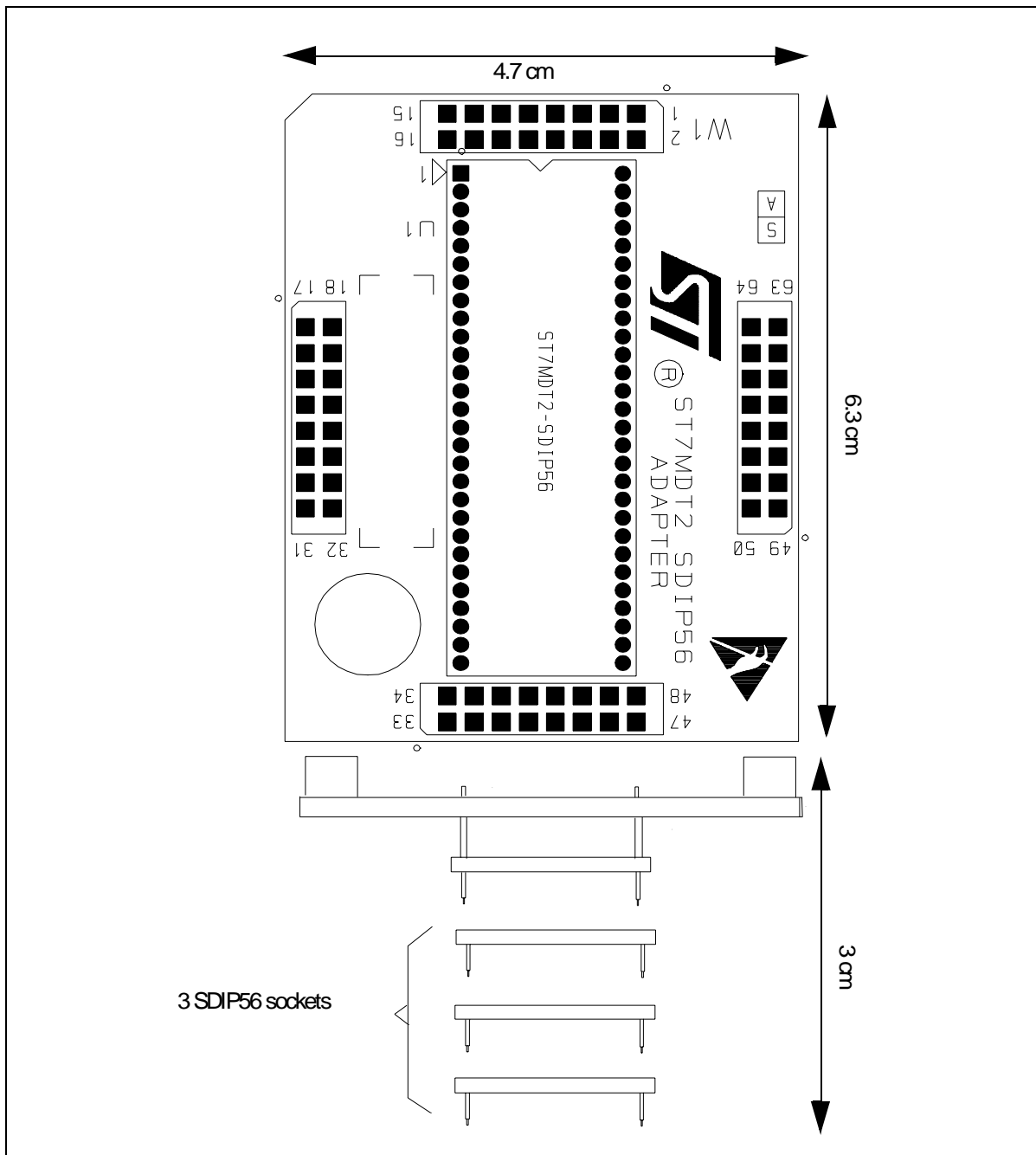


Figure 17: SDIP56 Pin Socket Device Adapter (ref.: DB408) Components Layout

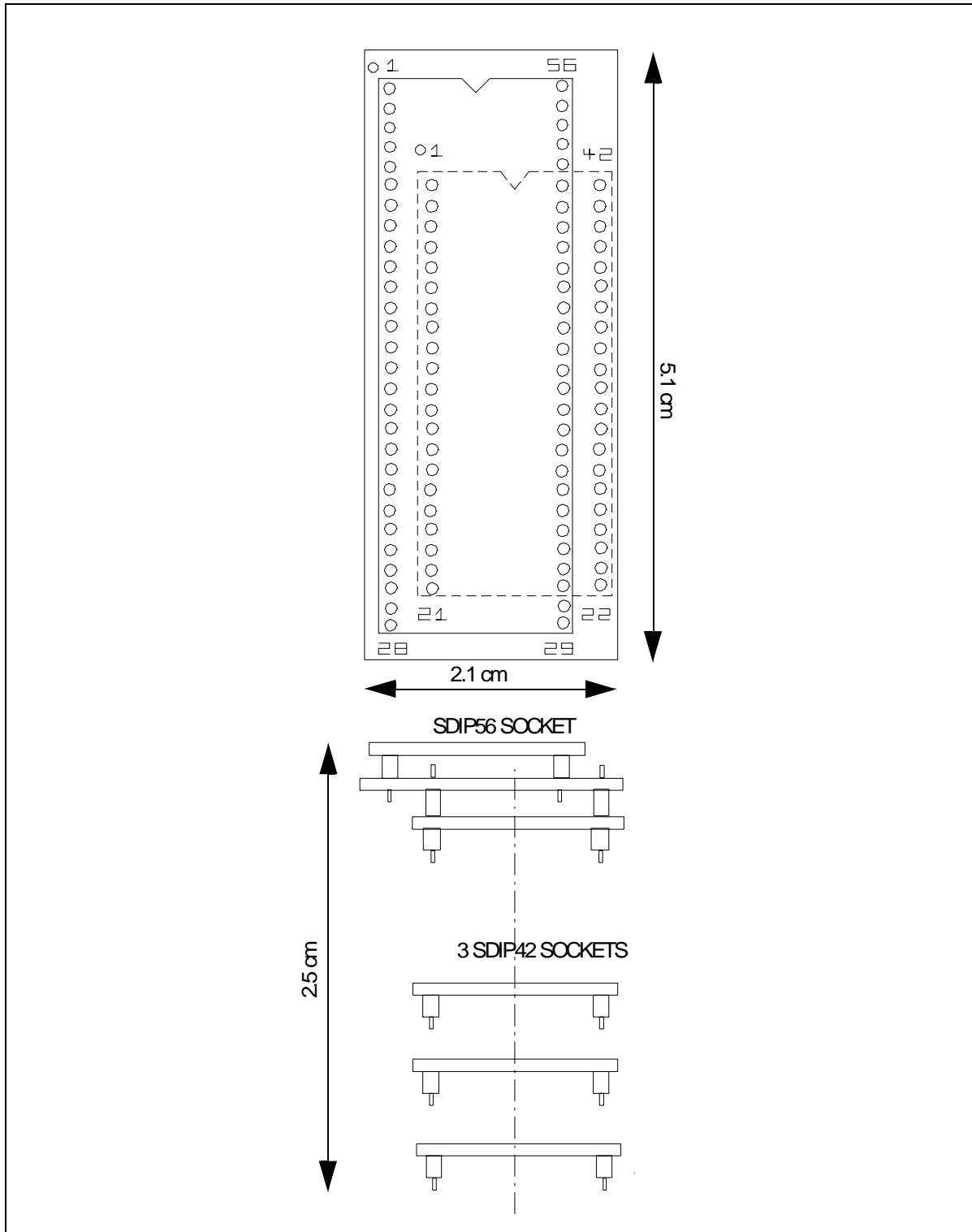


Figure 18: SDIP56 to SDIP42 Device Adapter (ref.: DB326) Components Layout

B.2 Device adapter pin-matching diagrams

On the top face of the ST7MDT2-Active Probe (ref.: DB407), there are 64 pins to which you can connect to a multimeter or oscilloscope to monitor signal values. However, if your target device package has less than 64 pins—as is the case for the TQFP44, SDIP56 and SDIP42 packages, you need to know how the device adapter pins are mapped onto the 64 pins of the ST7MDT2-Active Probe.

For this reason, we have included pin-matching diagrams for each of the three aforementioned package types. To use them, follow these instructions:

- Photocopy the corresponding diagram onto a stiff sheet of paper or cardboard.
- Cut along the dashed edges. Don't forget to cut along the inner rectangles.
- Punch out the rectangular boxes.
- Place the diagram over the 64 pins on the ST7MDT2-Active Probe (ref.: DB407) board. The two W1 pins must correspond.

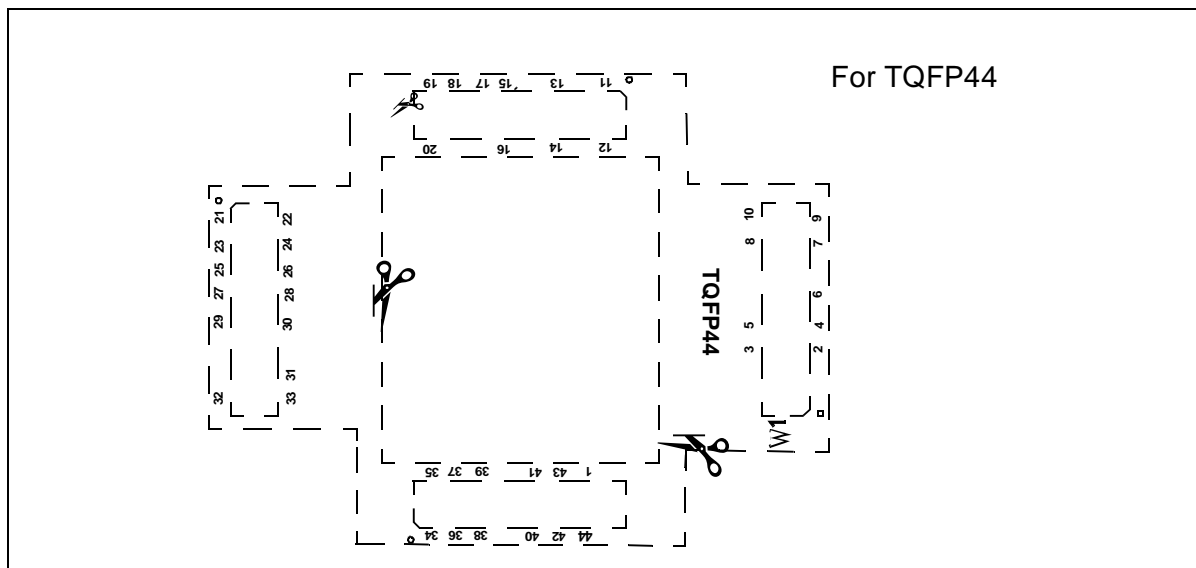


Figure 19: Pin-matching diagram for TQFP44 package

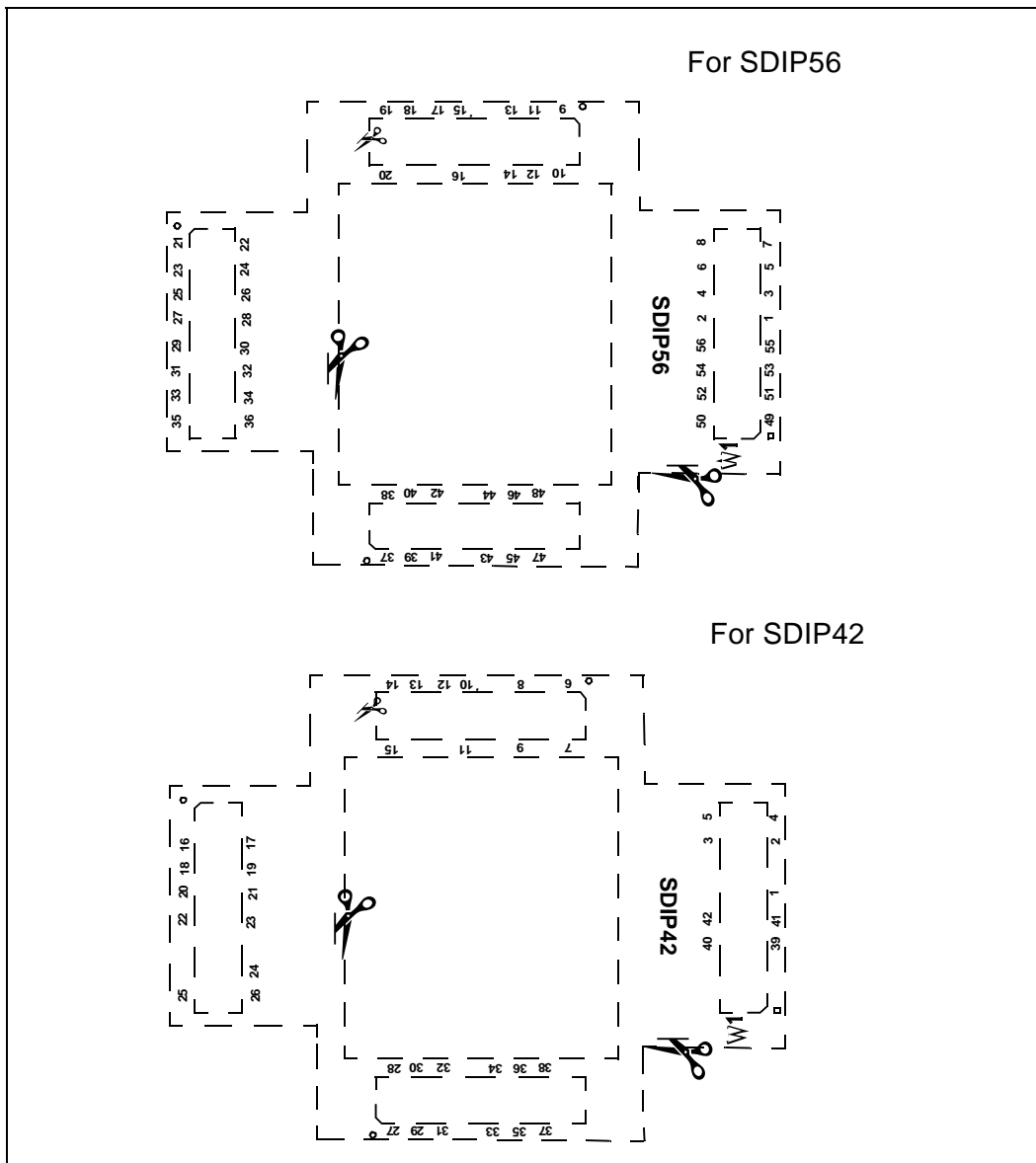


Figure 20: Pin-matching diagrams for SDIP56 and SDIP42 packages

0

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.

ST7MDT2-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.

PC (Program Counter)

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.

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

European manufacturers:

Manufacturer: Radiall
Dealer: Radialex
Phone: (+33) - 4 - 72 - 35 - 31 - 72.

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

- In SMB range: Number 114665.

Adaptable Female connectors to this connector are:

- SMB upright range
Number114005 for cable 2,6.
Number114003 for cable 4,2.
Number114009 for cable 3,8.
- SMB kneed range
Number114165 for cable 2,6.
Number114163 for cable 4,2.
- SMB to BNC range
Number191214. Adapter SMB female / BNC male.
Number191215. Adapter SMB female/ BNC female.

USA manufacturers:

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

Adaptable female connectors parts 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.

0

Index

A

Active Probe	
architecture	51
hardware	51
analyser probe signals	55

C

clock	
selecting frequency	57
selecting source	57
Clock Probe	61
configuration	
analyser probe input signals	55
output triggers	53
connections	
emulator power supply	19
emulator to PC	13
probe to emulator	14
SDIP42 package	18
SDIP56 package	18

D

Debuggers	20
Device pins	22
documentation	9

E

ECP	
definition of	77
EMC compliance	15
emulator kit	
configuration of	7
delivery checklist	11
functional limitations/discrepancies	59
installing software for	25
main functions of	6
operation of	7
software and documentation for	8

F

ferrites	
attaching to cables	15

H

hardware	
installation	12
hardware test	65
HDS2 emulators	
main features	49

I

input signals	55
installation	
hardware	12
STVD7	25

L

LEDs	56
load	
binary files	36
LVD.STATUS	59
LVD.VALUE	59

M

MB176	11
MCU	
emulated	5
on-chip peripherals	57
MCU configuration	42
MCU memory	
configuring	44
types	44

N

Nested / Concurrent Interrupts Buttons	60
Non-Maskable Interrupt Management	60

O

on-chip peripherals	57
watchdog	58
watchdog in Halt mode	59
output triggers	53

Index

P

parallel port	
troubleshooting connection problems	64
passive probe	
definition of.....	78
PC	
system requirements.....	11
peripherals	
configuring target	43
project settings	
modifying.....	37

Q

QFP44/TQFP44	67
QFP64/TQFP64	67

R

RAM	
minimum	11
ROM size	44

S

SDIP42.....	18
SDIP56.....	18
software	
updates	80
ST7MDT2	
devices.....	15
packages.....	15
ST7MDT2 EMU2B	
architecture	49
specific features of	49
ST7MDT2-Active Probe	
main features	49
STVD7	

about.....	27
build context	41
contexts	41
creating a workspace.....	32
debug mode.....	41
installing.....	25
main features.....	27
MCU configuration.....	42
opening binary files.....	36
opening workspaces	34
supported application files	29
supported toolchains	29
switching between contexts.....	42
toolchain paths	26
workspaces.....	28

support	
contact numbers for	79
for development kit	79
information required.....	79
web address	9

T

target device	
definition of	78
supported.....	5
TQFP64	20, 64
troubleshooting	63
connection error.....	63

U

user application board	
definition of	78

W

workspaces	
creating new	32
saving	39

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.

©2000 STMicroelectronics - All Rights Reserved.

Purchase of I²C Components by STMicroelectronics conveys a license under the Philips I²C Patent. Rights to use these components in an I²C system is granted provided that the system conforms to the I²C 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.