ST7MDT2-DVP2 Development Kit User Manual

Release 1.2

July 2001

Ref: DOC-ST7MDT2-DVP2

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, this development board is 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 67 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 Development board functional configurations 6
1.2 Software and documentation for the development kit 8
1.3 About this manual.... 8
1.4 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 Connecting the TQFP64 passive probe to your application board 14

Chapter 3: STVD7 15

3.1 Installing STVD7 15
3.2 Launching STVD7 16
3.3 About STVD7 debugging features 17
3.4 Workspaces 18
3.5 Toolchains and application files 19
3.6 Creating a workspace 22
3.7 Opening an existing workspace 24
3.8 Opening binary files 26
3.9 Opening lone programmable files (*.s19 or *.hex) 27
3.10 Changing your project settings 28
3.11 Saving workspaces 30
3.12 Debug context and Build context 32
3.13 Configuring the MCU 33
3.14 Start debugging! 38

Chapter 4: Programming ST7 Devices 39

4.1 Device programmer features 40
4.2 Programming methods 41
4.3 Device installation 42
4.4 Starting the Windows Epromer 43
4.5 Configuring the Epromer 44

Chapter 5: Hardware Features 47

5.1 ST7MDT2-DVP2 development board layout 47

5.2 ST7MDT2-DVP2 emulation architecture 48
5.3 Link to PC 48
5.4 Power supply 49
5.5 Jumper and solder point descriptions 50
5.6 CAN features 51
5.7 Pin descriptions and package footprints 53
5.8 Trigger/trace settings 60
5.9 Hardware events 62
5.10 On-chip peripherals 62
5.11 Emulation functional limitations and discrepancies 64

Appendix A: EMC Conformity and Safety Requirements 67

Appendix B: Troubleshooting 69

B.1 Identifying the problem 69
B.2 Changing the parallel port setup on your PC 70
5.12 Running the hardware test 70

Appendix C: Glossary 73

Product Support 77

Getting prepared before you call..... 77
Contact list 77
Software updates..... 78
Hardware spare parts 78

Index 79

1 INTRODUCTION

Thanks for choosing the ST7MDT2-DVP2 development kit! The ST7 DVP2 family of development kits offer the following new features:

- Delivered with the debugger software package — ST7 Visual Debug!
- Trace buffer recording, viewing and output.
- In Situ Programming (ISP) ability (for MCUs that support this feature).

This manual describes how to start and use the ST7MDT2-DVP2 development kit for the ST72334 MDT2 CAN-less family and the ST7511R9 MDT2 CAN family of MCUs, allowing you to get acquainted with the ST7 microcontroller world and become familiar with the methods for developing and debugging ST7-driven applications.

Note: *If you come across any terms or abbreviations you do not understand, you can check their meaning in the Glossary on page 73.*

This manual also provides a guidance for programming a selection of Flash, Eprom and OTP (One Time Programmable) ST7 microcontrollers.

The ST7MDT2-DVP2 development kit contains all the necessary resources that will help you:

- design, develop and debug ST7 application software running in a real environment,
- program selected ST7 devices in a variety of modes (refer to *Table 3* on page 39).

First off, check that the ST7 MCU that you have picked for your application is in the list of devices supported by this version of the ST7MDT2-DVP2:

Supported Devices
ST72124J2/J4
ST72314J2/J4 ST72314N2/N4
ST72334J2/J4 ST72334N2/N4
ST72532R4
ST72311R6/R7/R9
ST72512R4
ST72511R6/R7/R9

The development kit can be used as a tool to emulate applications on the target MCU, or as a chip programming tool as summarized in the following sections.

1.1 Development board functional configurations

Figure 1 shows the development board of the ST7MDT2-DVP2 development kit in an ST7 MCU Emulator configuration.

Figure 2 shows the development board of the ST7MDT2-DVP2 development kit in an ST7 MCU Programming Board configuration.

Figure 3 shows how you can set up the Development Board to perform in situ programming of devices on an application board.

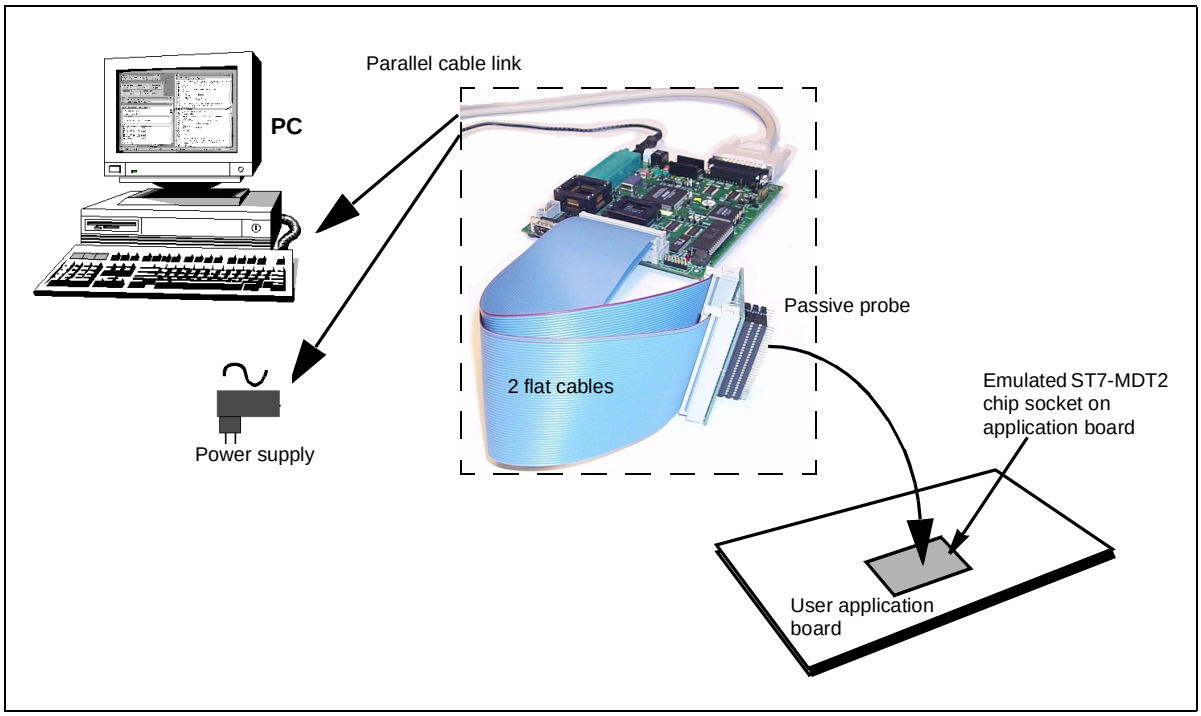


Figure 1: Using the development board as an ST7 MCU emulator

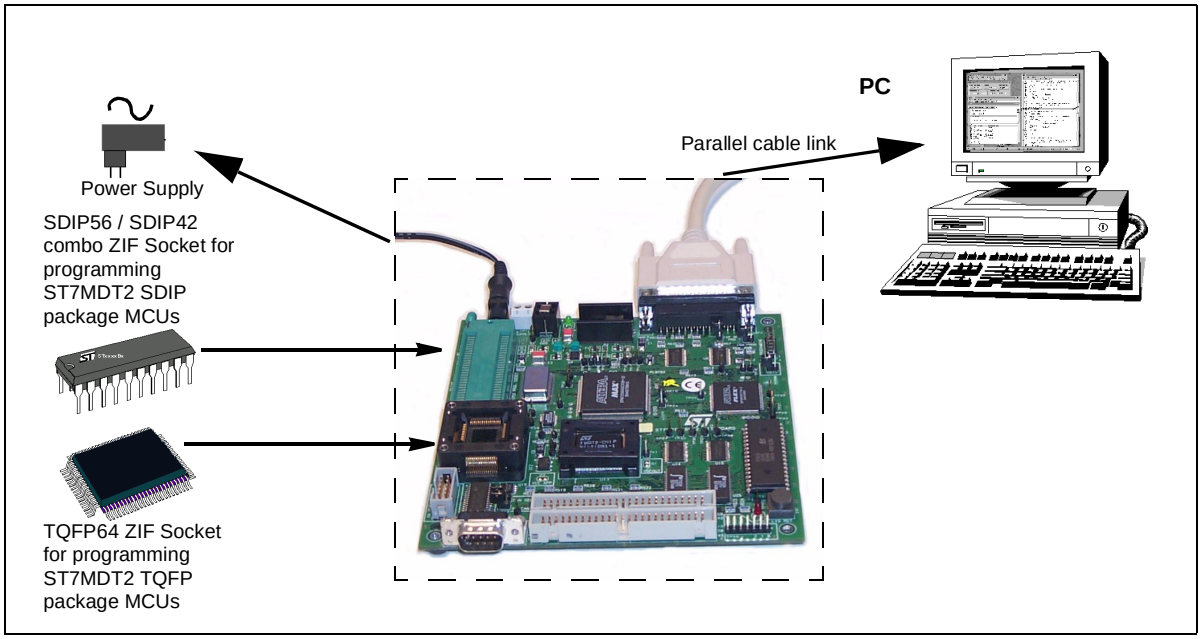


Figure 2: Using the development board as an ST7 MCU programming board

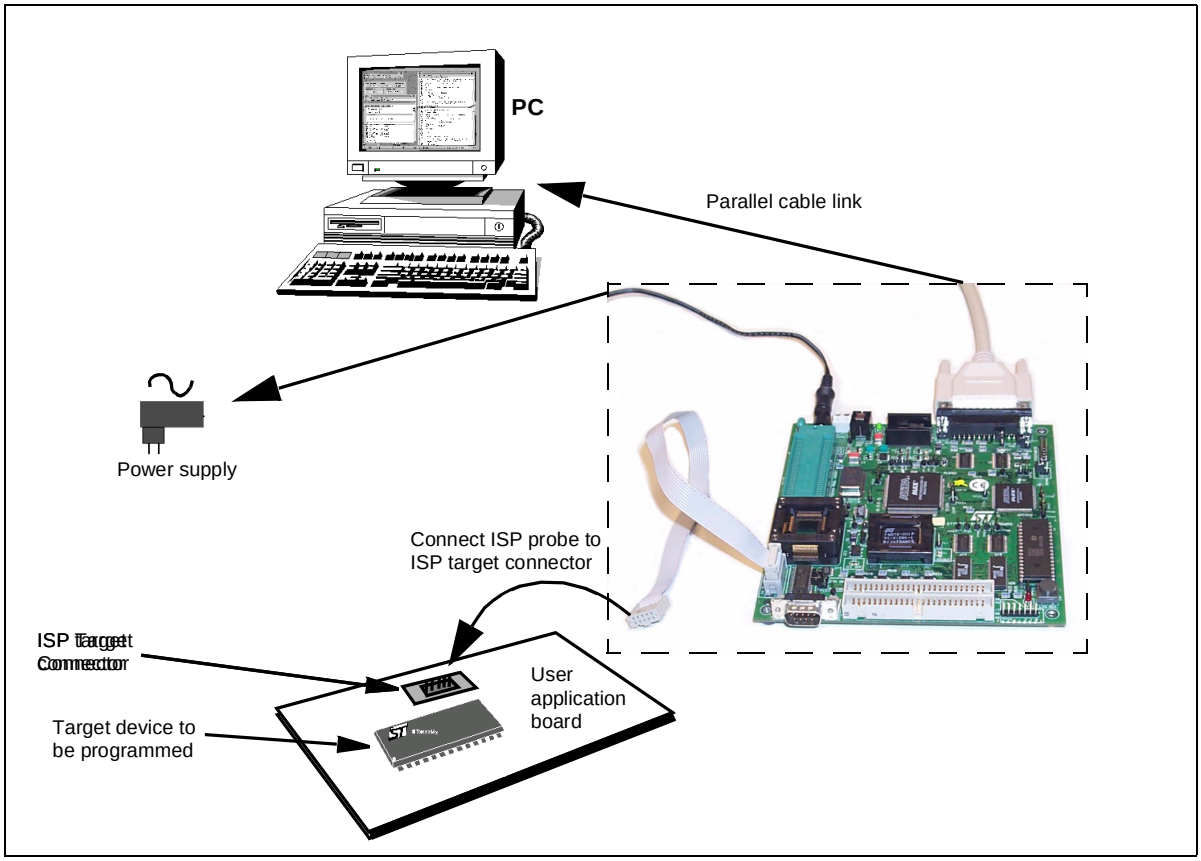


Figure 3: Using the development board for In Situ Programming (ISP)

1.2 Software and documentation for the development kit

The “MCU on CD” CD-ROM contains:

- ST7 Tools, comprising the following software:
  - The source-level graphic debugger, STVD7, that operates with ST7 development kits and ST7-HDS2 Emulators 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 documents in PDF format, such as:
  - Datasheets for the ST7 MCU family
  - *ST7 Programming Manual*
  - *ST7 Assembler-Linker User Manual*
  - *STVD7 User Manual*

1.3 About this manual....

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

How to start debugging your application using your development kit and STVD7 is described in *Chapter 3: STVD7* on page 15.

How to program devices with the development kit is described in *Chapter 4: Programming ST7 Devices* on page 39.

The development kit’s hardware features are described in *Chapter 5: Hardware Features* on page 47.

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 check boxes 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>***....”.

1.4 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 77. We’ll be glad to help you!

2 GETTING STARTED

2.1 Your system requirements

The ST7MDT2-DVP2 development kit (both hardware and software components) has been designed to work with PCs having the following configurations:

- One of the following operating systems: Microsoft® Windows® 95, 98, 2000 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 ST7MDT2-DVP2 development kit contains:

1 One development board (Ref.: MB289).

2 One parallel cable for PC connection and two EMC ferrites.

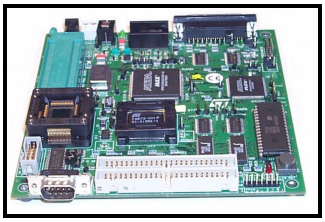
3 One 5 V external DC power supply with female connector cable (two possibilities exist).

4 SDIP56 passive probe connector (Ref.: DB347) and two 50-pin flat cables.

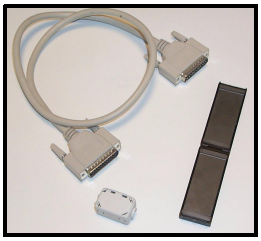
5 One TQFP64 passive probe connector (Ref.: DB404) and one SDIP56 to SDIP42 (ref.: DB326) device adapter.

6 One ISP 10-pin flat cable for in situ programming.

(1)

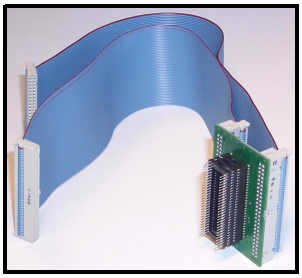


(2)

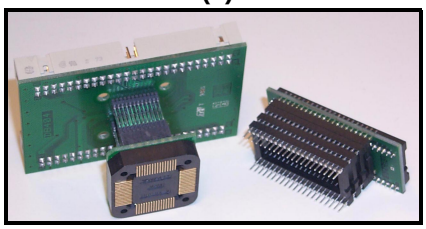


(3)

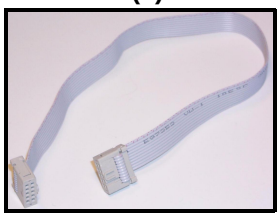
(4)



(5)



(6)



7 One Yamaichi QFP64 socket, with its cover, screws and washers. (Not shown.)

8 One *MCU-on-CD* CD-ROM.(Not shown.)

9 This manual.(Not shown.)

Note: A TQFP44 passive probe is available for the ST7MDT2-DVP2 but must be ordered separately. Contact your nearest STMicroelectronics sales representative (see page 77).

2.3 Installing the hardware

To install the hardware, follow these steps:

1 Shut down and power-off the PC that is to be connected to the development board.

2 Connect one end of the supplied parallel cable to the parallel connector (P2) on the development board. Connect the other end of the parallel cable to the LPT1 or LPT2 parallel port on your PC.

Note: The supplied parallel cable has been tested in order to operate properly on most PCs. Do not use any other cable, especially if it is longer than the one provided in the kit—the board may not operate properly.
The cable should be connected directly to the DB-25 female connector of the PC parallel port. This connector is similar to the one installed on the board. Do not insert any additional cables or switchboxes between the PC and the board: a malfunctioning of the board may result.
If a dongle is mounted on the PC parallel port, it should not interfere with the programming board. Should you notice that the board is dysfunctional, remove the dongle and restart the installation procedure.

3 Connect the two 50-pin passive probe cables to J1 and J2 on the development board.

4 To the other ends of the 50-pin cables, connect the passive probe and/or device adapter corresponding to the MCU package you wish to emulate:

- SDIP56 passive probe alone for SDIP56 MCU packages.
- TQFP64 passive probe alone for TQFP64 MCU packages.
- SDIP56 passive probe with the SDIP42 device adapter for SDiP42 MCU packages.

Caution: *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 Section 5.7.2: QFP64/TQFP64 footprint discrepancy issues on page 59).*

5 **EMC-Compliant Probes (optional)**: In order to work under an EMC-compliant environment, you will have to clip one EMC-ferrite on both 50-wire flat cables linking the application to the development kit board. Place this ferrite as close to the development kit board as possible. You also need to clip one EMC ferrite on the power supply wire coming from the power supply box. See *Figure 4* on page 13.

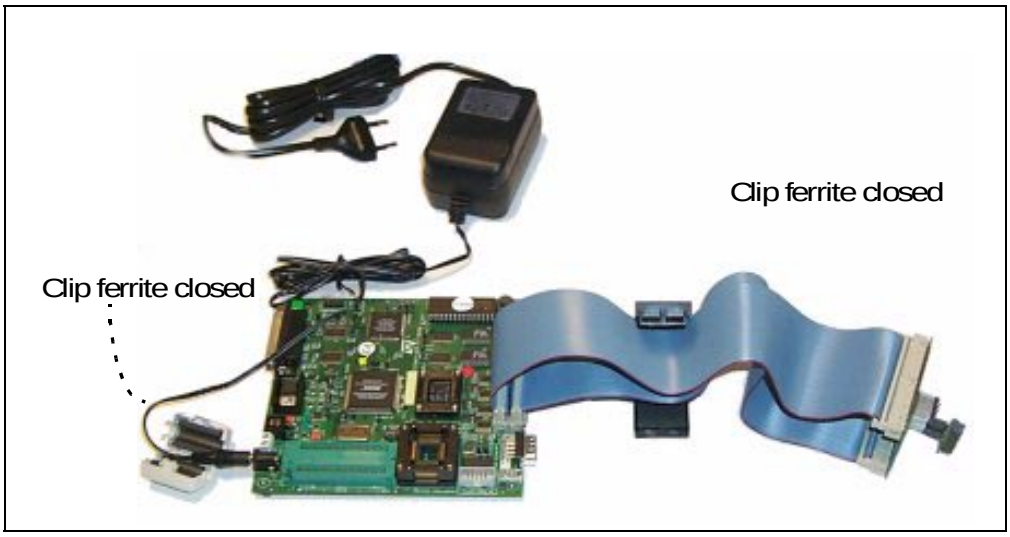


Figure 4: Connecting the ferrites

6 Connect the passive probe to the appropriate socket on your application board.

For the **TQFP64 passive probe**, a Yamaichi QFP64 socket has been included. Follow the instructions in *Section 2.4* on page 14 to connect the TQFP64 passive probe to your application board.

7 Plug the DC power supply provided with the development kit into a power outlet. Connect the power cable to the development board. The green Power LED will light up.

Note: *The development board can also be fed via the JP1 connector by an external stabilized power supply (5 V ± 0.25 V, 1 A) not provided with the Kit.*
If the board is fed via the JP1 two-point connector make sure that the right feeders lead to the right polarities.

8 Power on the PC and proceed with the installation of the software as described on *Section 3.1* on page 15).

Caution: *Do not use the jumper connections TP17, TP4 and TP5 — they are for factory testing only and modifications to them could cause your development board to malfunction. Refer to Section 5.5 on page 50 for a description of all jumpers and solder connections on your development board.*

2.4 Connecting the TQFP64 passive probe to your application board

A Yamaichi QFP64 socket and its cover are provided in the package (Ref.: DB200). Before going through the procedure, make sure that you were properly delivered the socket, its cover and its screws and washers.

To connect the TQFP64 passive probe to your application board, proceed as follows (see following figure for flow order):

1. Solder the Yamaichi QFP64 socket base onto your application board (see *Section 5.7.2* on page 59 for footprint information). Do NOT screw the socket cover onto its base.
2. Place the end of the TQFP64 passive probe onto the Yamaichi socket base, taking care to align pin 1 of your application board with pin 1 of the passive probe. Pin 1 is indicated by a chamfer on the passive probe and by a little arrow or chamfer on the Yamaichi socket.
3. Once placed, you can screw the TQFP64 passive probe onto the Yamaichi socket using the four threaded holes located on the upper surface of the passive probe.

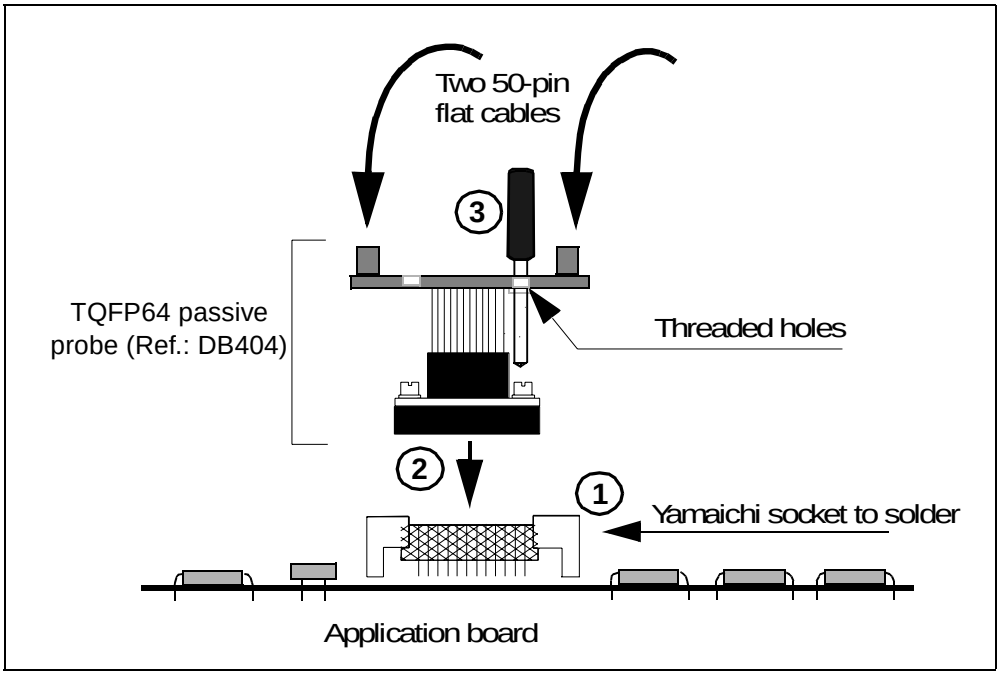


Figure 5: Connecting the TQFP64 probe to your application board

If you require supplementary sockets, the commercial references of Yamaichi QFP64 sockets are given below:

QFP64/TQFP64 Yamaichi socket WITH positioning pins	IC149-064-108-S5
QFP64/TQFP64 Yamaichi socket WITHOUT positioning pins	IC149-064-008-S5

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 development kit 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 and program your application with your DVP, you must install STVD7 for DVP and the Windows Epromer.**

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.

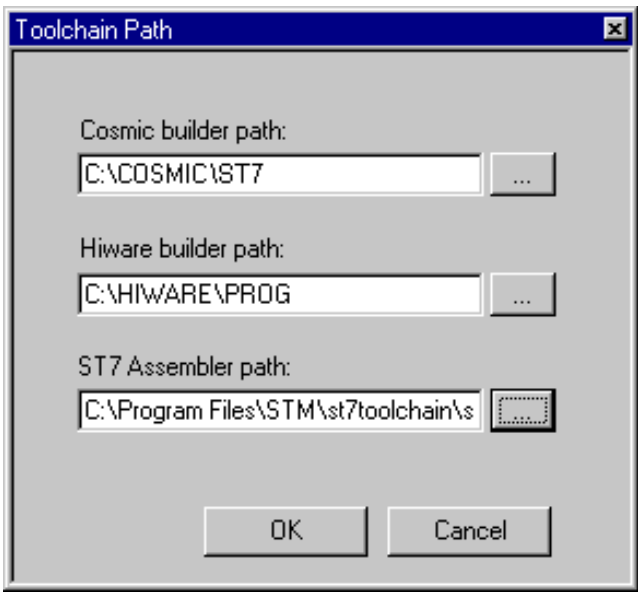
The installation is now complete. You will be prompted to reboot your computer. You should do so before launching STVD7.

3.2 Launching STVD7

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

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 **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 256 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.
- **Hardware Events** can be used to control the trace recording or the sending of signals to the trigger outputs.
- 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 22. 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 20).

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 19).
- 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 32).
- 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 21).
- **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/dvp/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 21 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`

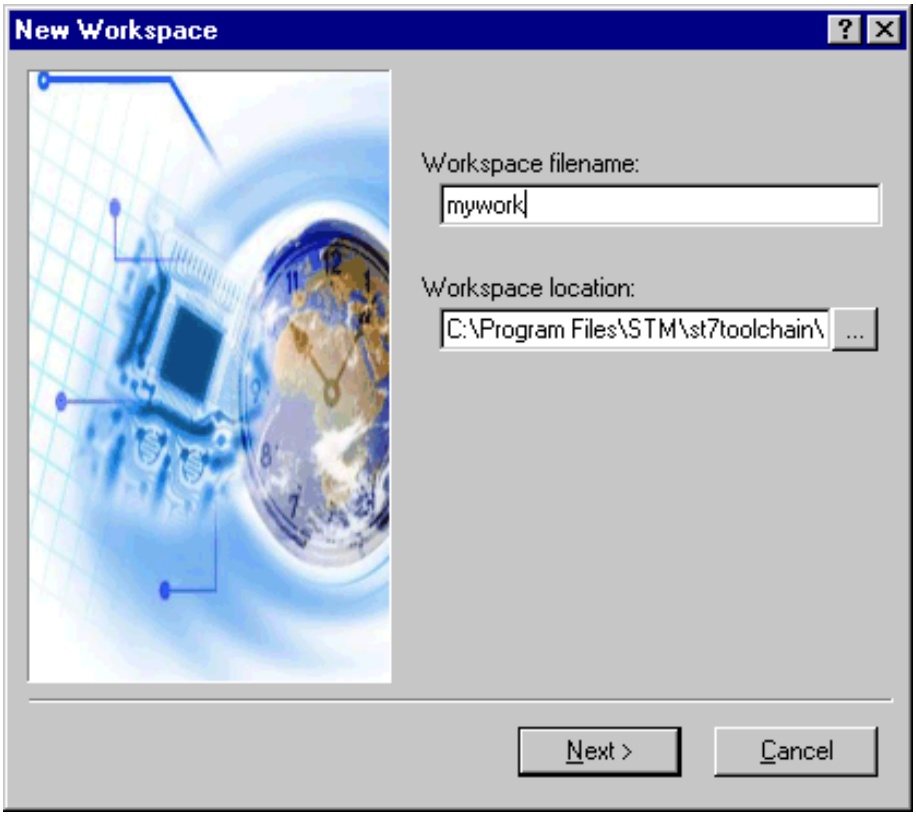
Resulting executable file:	`.s19 or .hex`	`.abs, .elf`	`.elf`
Necessary Intermediate Files:	`.map, .lst`	`.o, .dbg`	<NONE>

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.

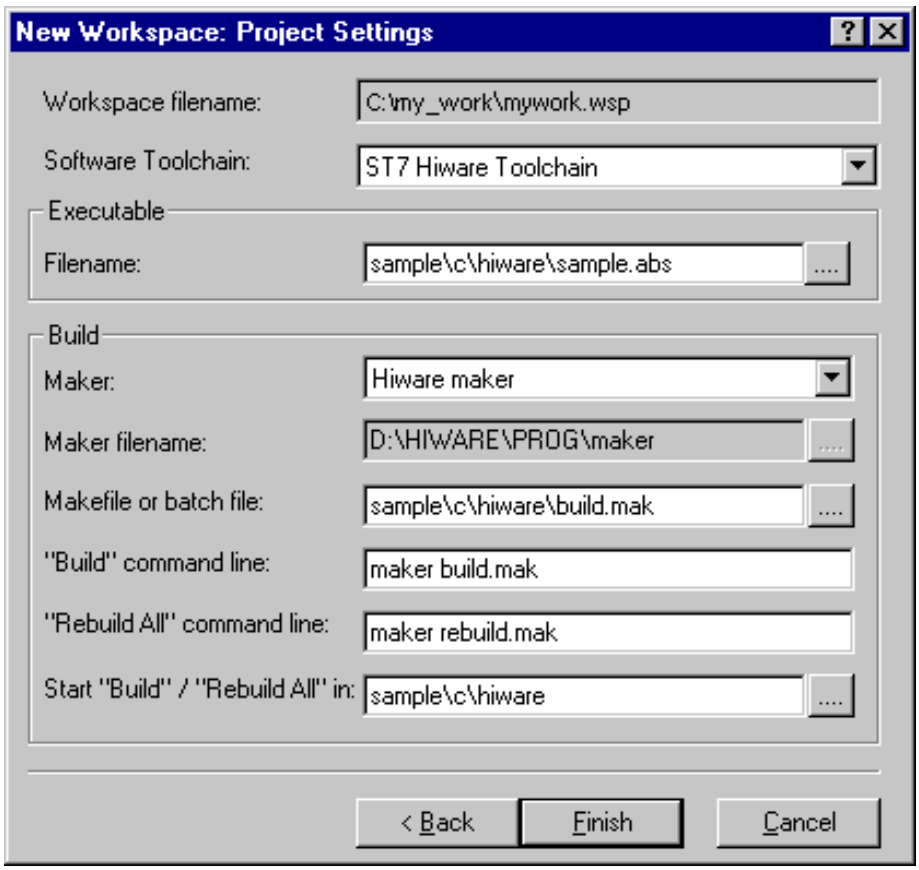
Note: *It is always preferable to have access to all of the files generated by the development toolchain. However, you can load* `*.s19` *or* `*.hex` *binary files directly and have limited debugging capabilities (refer to Section 3.8 on page 26).*

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 21)—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 20). 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.

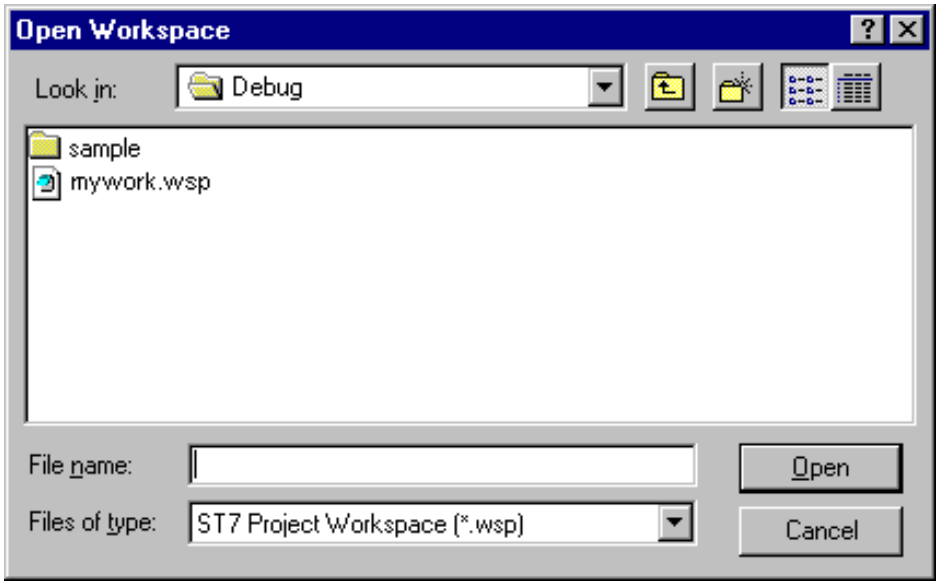
When you create a new workspace, the first time you switch to Debug context (see *Section 3.12* for an explanation of STVD7 contexts), the MCU Configuration window will automatically open to prompt you to choose you target MCU and confirm or modify its option and memory configuration (see *Section 3.13* on page 33).

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.

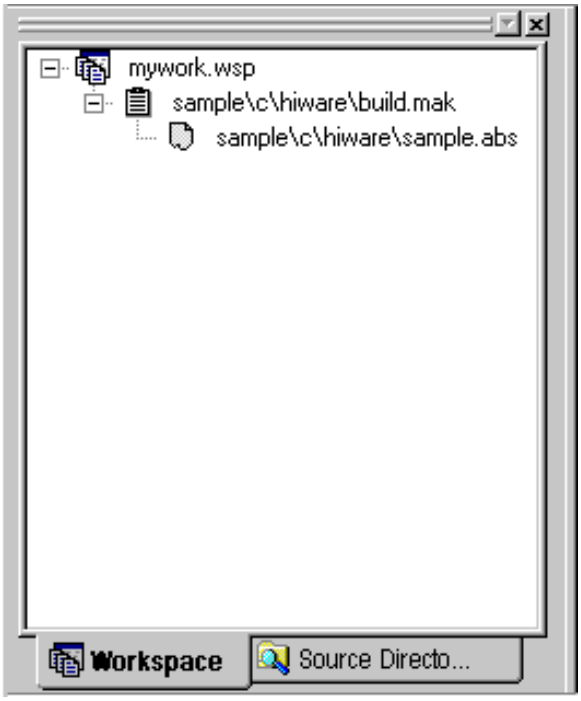
Note: There are a number of sample workspaces provided with STVD7 that you can open to get familiar with STVD7. These samples are listed in Table 1 on page 20.

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.



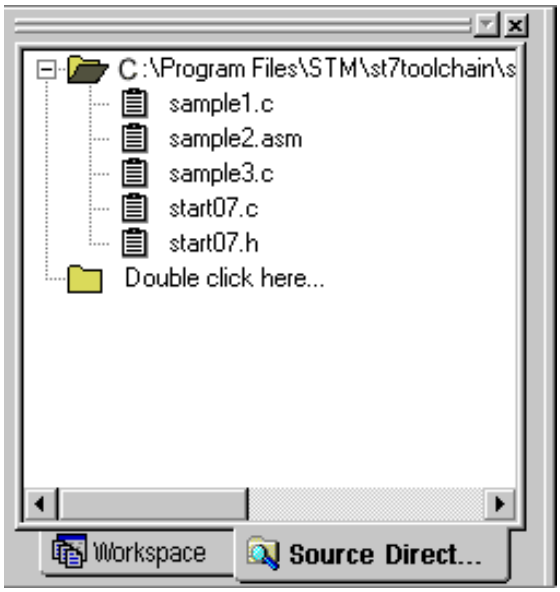
2 The Workspace window opens.

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.



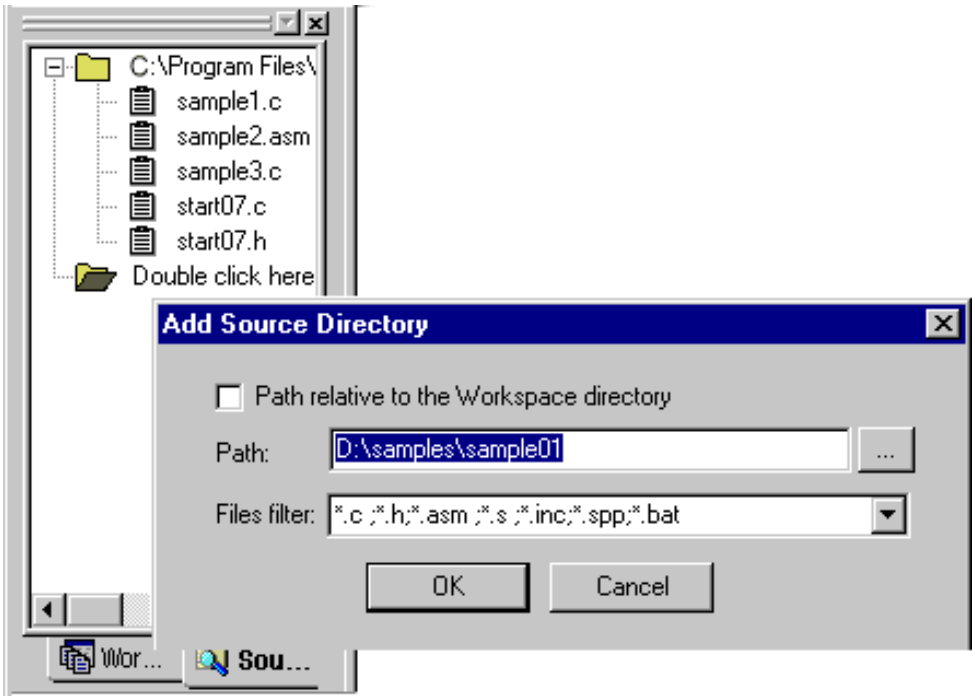
Note: Although the name of the executable file is shown in the Workspace window, it has not yet been loaded into the emulation memory—see page 26.

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

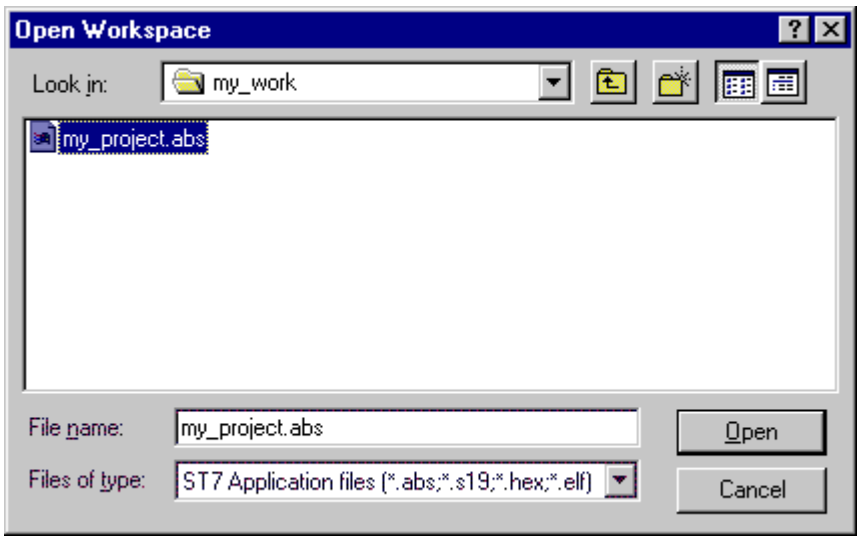
If you do not have access to the source or intermediate files generated by a toolchain, you may also load **`*.abs`**, **`*.s19`**, **`*.hex`** or **`*.elf`** files on their own using the ***Open Workspace*** command.

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 27.*

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**.

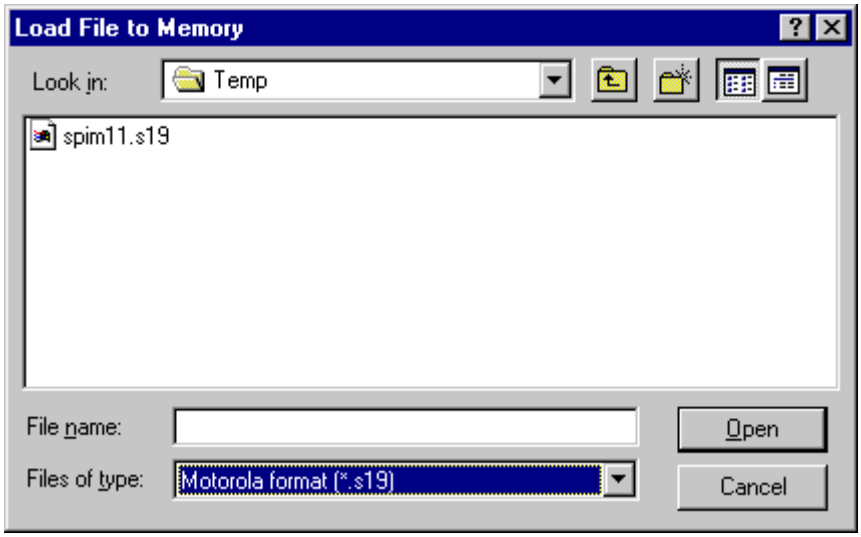
1 Launch STVD7 and select ***Debug>Start Debugging*** from the main menu or click on .

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.

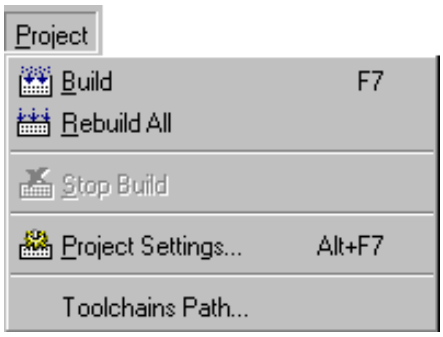


6 Select your programmable file (`*.hex` or `*.s19`) and click ***Open***.

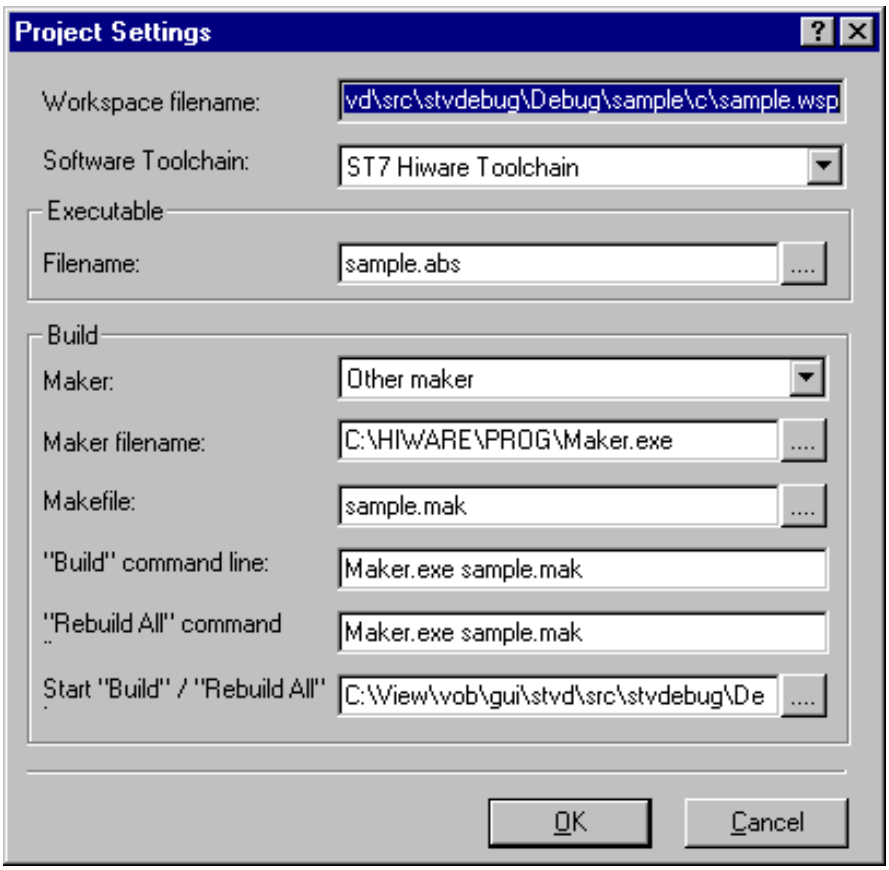
The binary code in the `.s19` or `.hex` file will be loaded into STVD7 and you will be able to access the Disassembly window and the Memory window. A workspace file (of the same name as the programmable file, but with an extension .wsp) will be created automatically.

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.

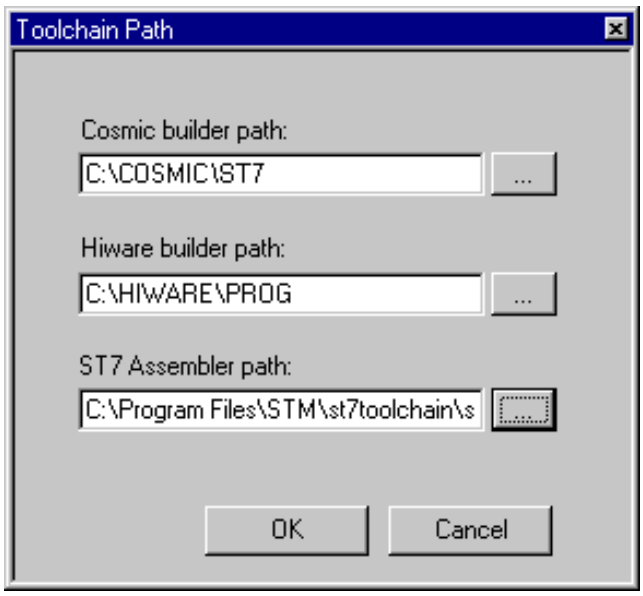


From the main menu, select ***Project>Project Settings***.



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 6 on page 31*), 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.

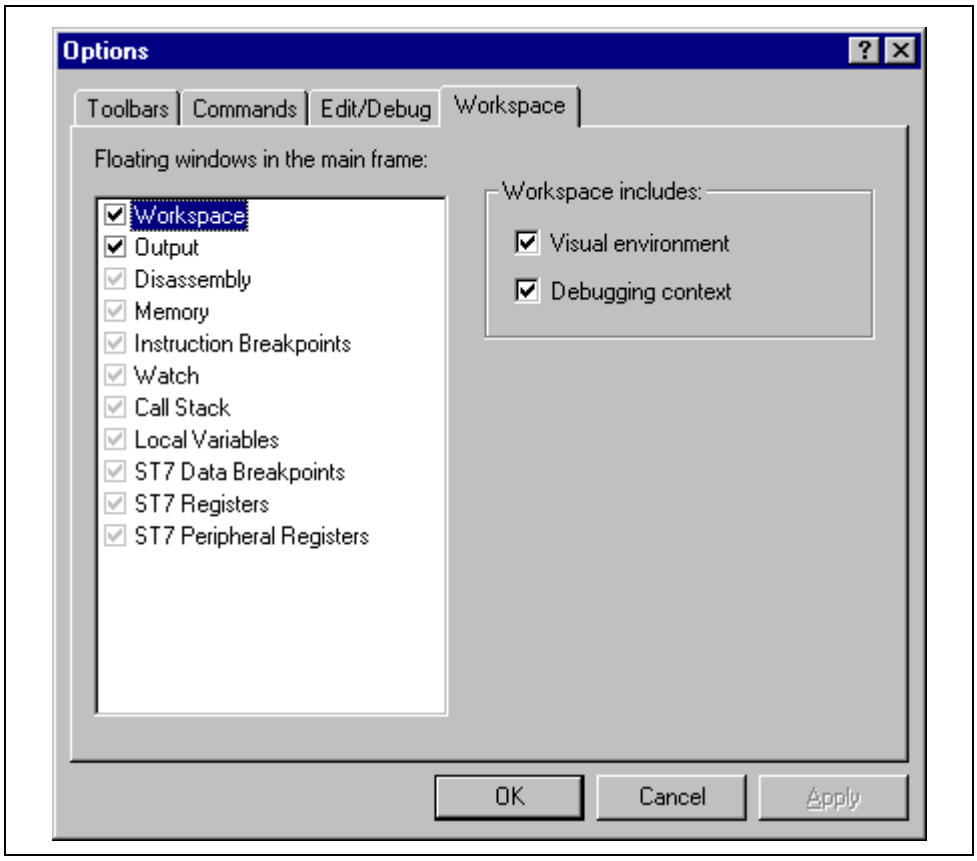


Figure 6: Options window

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 33) 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 a development kit 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 7*.

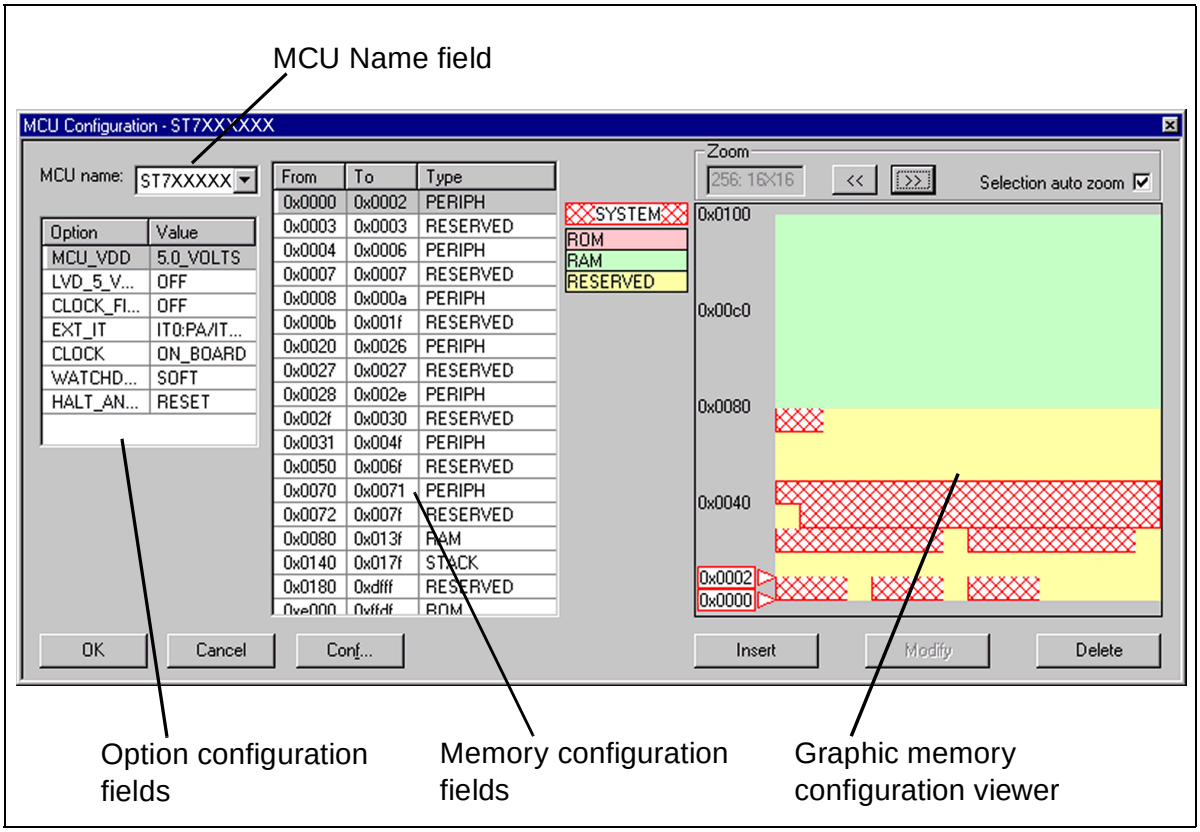


Figure 7: MCU Configuration window

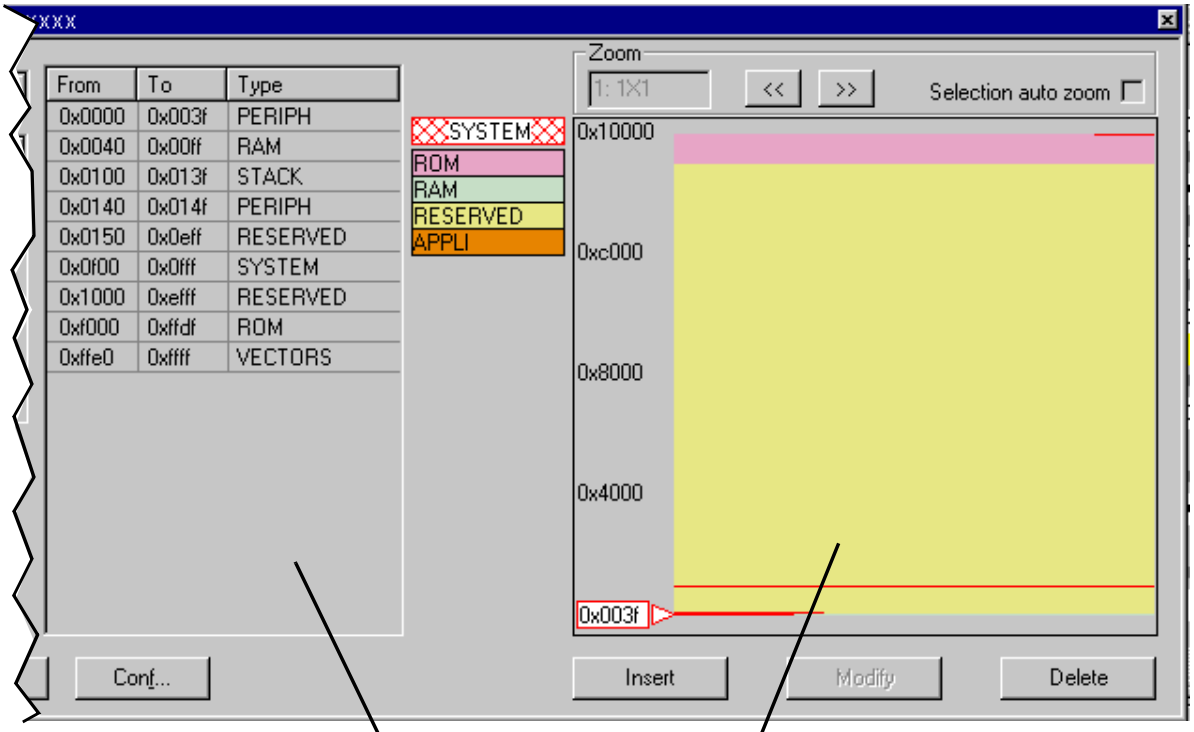
Note: The options shown in the above example may not be available for your particular target MCU.

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 development kit 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 36* 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 development kit 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 development kit. Properties are linked to the user hardware. This memory type can be modified.

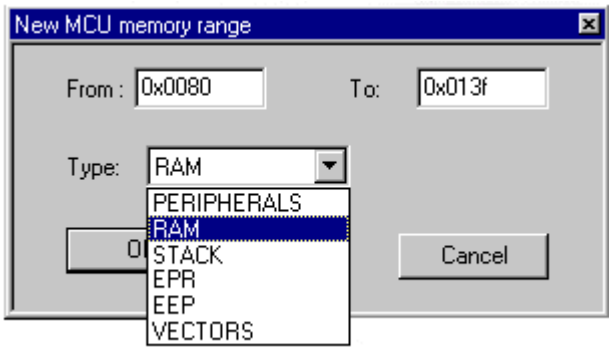
For most target MCUs, you may modify the following types of memory zone: **RAM**, **ROM**, **Reserved** and **Application**. This feature would enable you, for instance, to temporarily decrease the RAM zone, increase the size of the ROM (to exceed what is available on the real microcontroller) during 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 applicationusing the development kit. 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 PROGRAMMING ST7 DEVICES

Once bug-free and ready for operation, your application program needs to be transferred into an ST7 MCU program space.

With the ST7MDT2-DVP2 development kit, you may program the MCUs shown in *Table 3*.

Table 3: Programmable devices

Supported Device	Programmable Memory Type[1]	Programming method(s)[2]
ST 72124J2/J4	EEPROM and OTPROM	ZIF Sockets[3] and ISP
ST 72314J2/J4 ST 72314N2/N4	EEPROM and OTPROM	ZIF Sockets and ISP
ST 72334J2/J4 ST 72334N2/N4	EEPROM and OTPROM	ZIF Sockets and ISP
ST72532R4	EEPROM and OTPROM	ZIF Sockets
ST 72311R6/R7/R9	EEPROM and OTPROM	ZIF Sockets
ST 72512R4	EEPROM and OTPROM	ZIF Sockets
ST 72511R6/R7/R9	EEPROM and OTPROM	ZIF Sockets

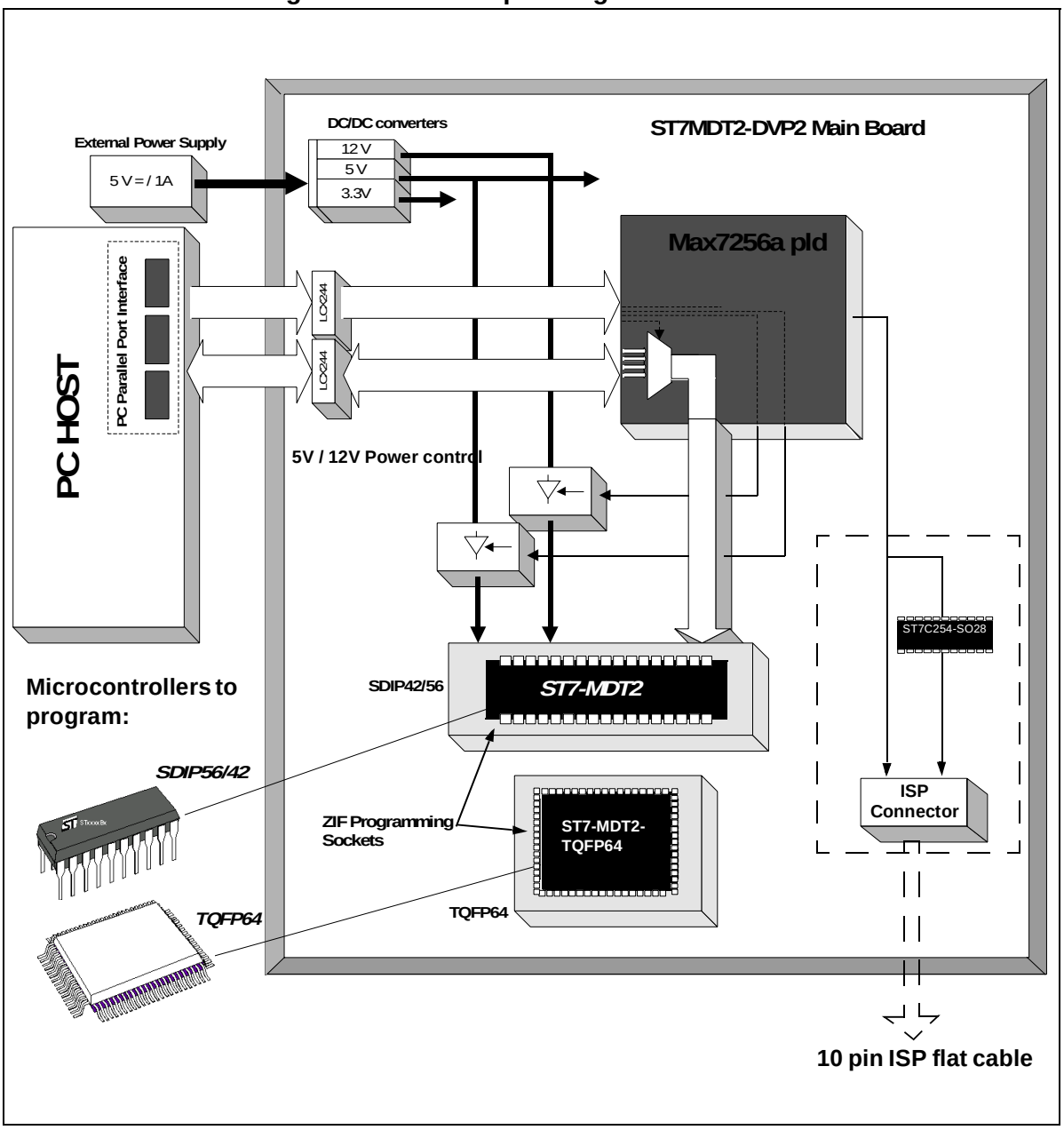
1) For more information about the programmable memory for each target device, refer to the target device's datasheet.

2) For descriptions of programming methods, see *Section 4.2* on page 41.

3) SDIP56, SDIP42 and TQFP64 packages are programmable using the ZIF sockets provided with the development kit. **TQFP44 packages are not supported by the development kit.**

4.1 Device programmer features

Figure 8: General eproming architecture



4.2 Programming methods

4.2.1 ZIF sockets

The ST7MDT2-DVP2 development board is provided with a SDIP56/42 **Zero Insertion Force (ZIF)** socket which allows the programming of SDIP56 and SDIP42 packaged MCUs.

A TQFP64 ZIF socket is also provided on the development board, for the programming of TQFP64 packaged MCUs.

4.2.2 In Situ Programming

In addition to classic MCU programming using ZIF sockets, the ST7MDT2-DVP2 development kit is provided with an In Situ Programming (ISP) functionality. This allows the user to program a target MCU mounted on an application board.

Note: Only Group 1 target devices (see page 39) support the ISP functionality. In order to take advantage of the ISP functionality, your application board must be designed to allow for in situ programming. Only target MCUs provided with a power supply of 5 V support the ISP functionality.

A provided 10-pin flat cable can be connected to the ISP connector (Ref: W2) at one end, and to an ISP target connector which is contained on the application board. Refer to the target device's datasheet for details on adding an ISP connector to your application board. You will find the ISP connector pin number assignment below, as it is on your development kit board.

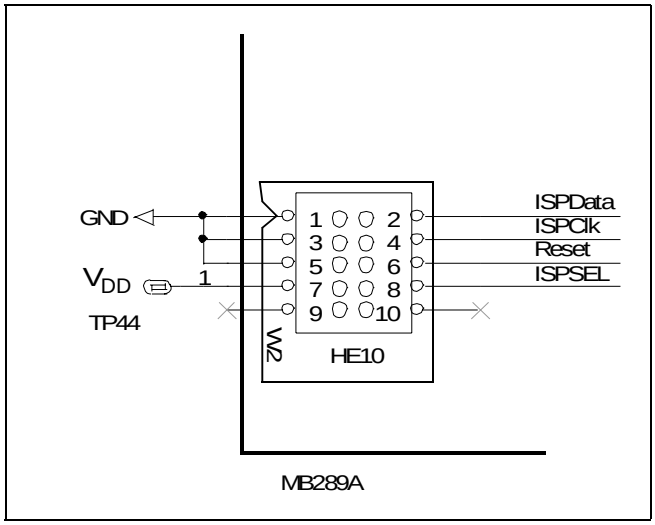


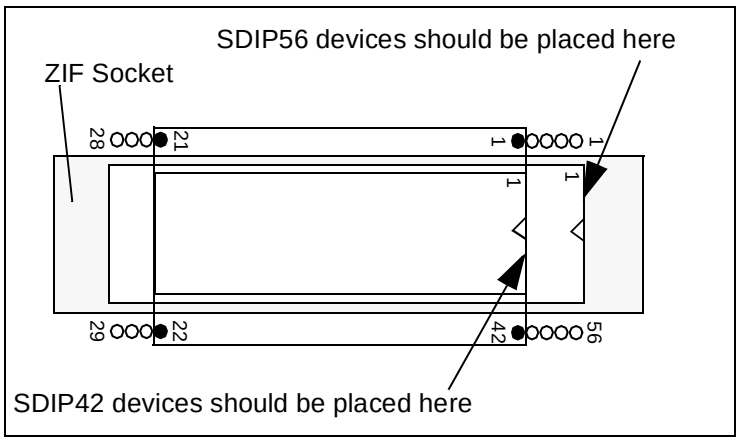
Figure 9: ISP connector

The Windows Epromer fully supports the ISP functionality. For more information concerning ISP consult the application notes (ref.: AN1179).

4.3 Device installation

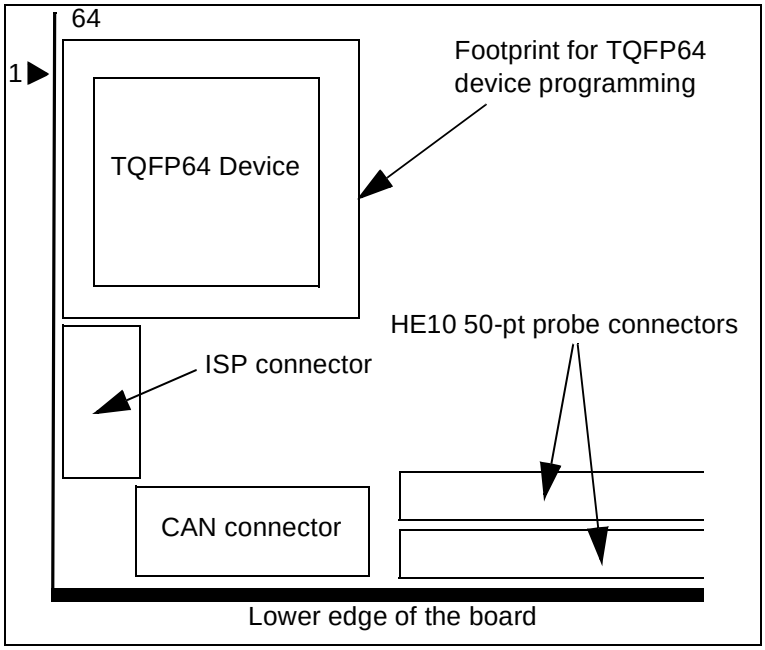
Caution: *Take care when placing the device into a socket so as not to damage the device or the board. Never insert or remove devices when powered. Devices are powered only during read or write operations.*

Place the SDIP56 device into the zero insertion force (ZIF) socket mounted on the board (location U6) with the erasure window on top and pin 1 matching the **hollow (unfilled) circle No.1** on the board.



You may also place SDIP42 devices on the same socket. In this case, pin 1 of the device should be aligned with the **solid filled circle No.1**, as shown in the diagram at right.

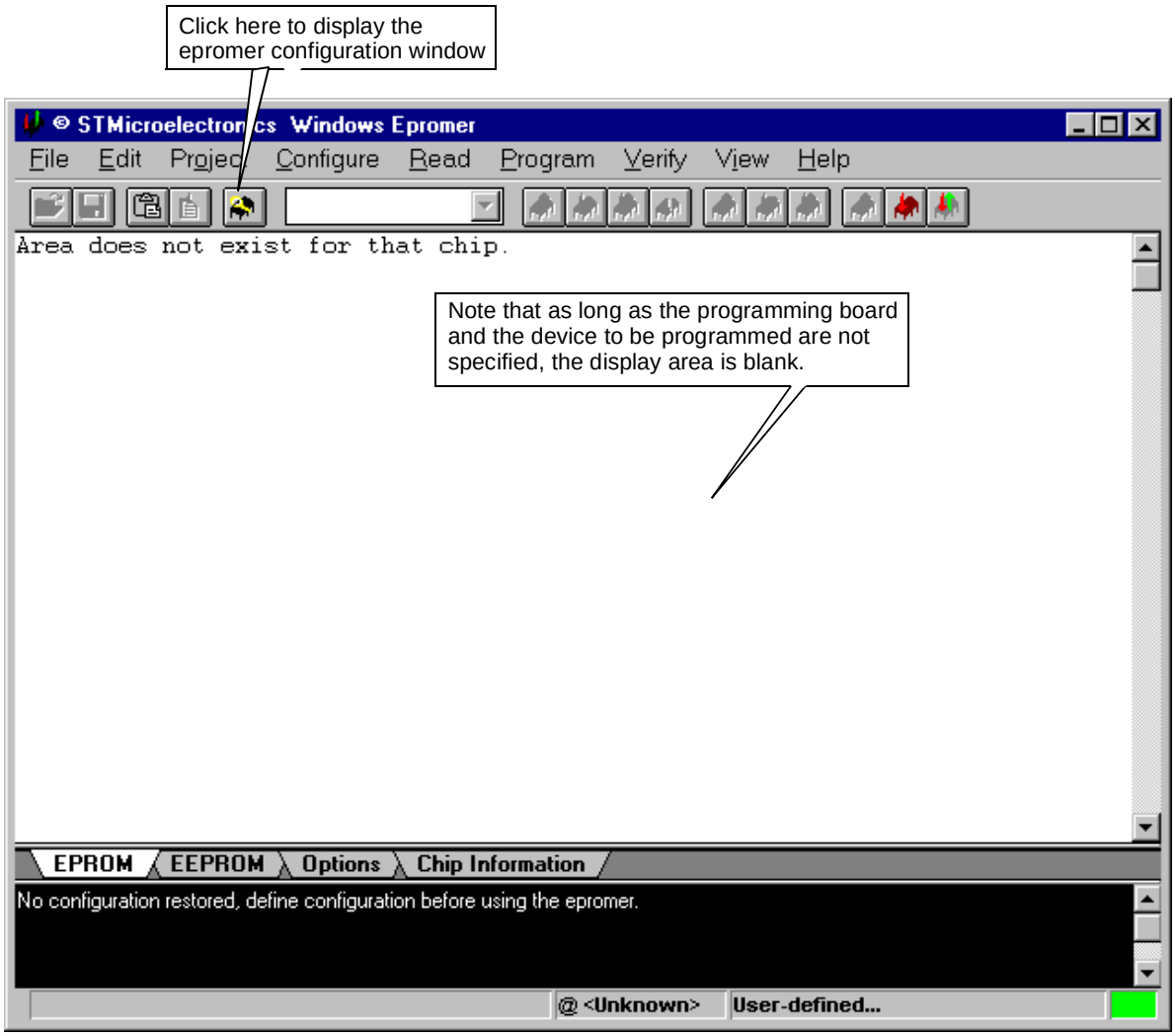
You may also program TQFP64 devices using the TQFP64 ZIF socket at location U11. Place the device as indicated in the figure at right.



4.4 Starting the Windows Epromer

1 To start the Windows Epromer (Winee), select ***Start>ST7 Tools>Windows Epromer***.

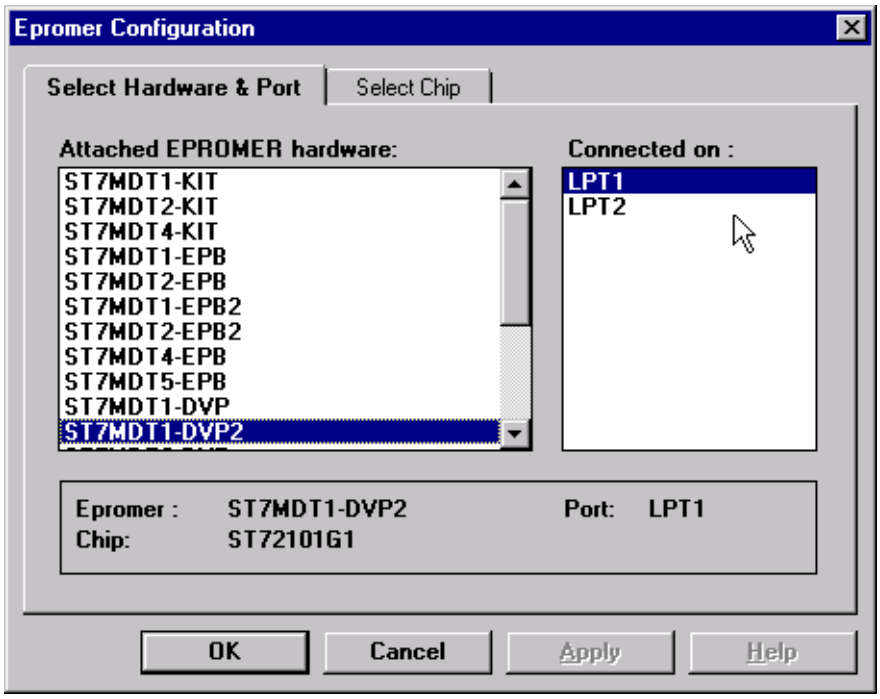
The Epromer main window appears:



4.5 Configuring the Epromer

Follow these steps:

1 In the main window tool bar, click the icon to open the epromer configuration dialog box:

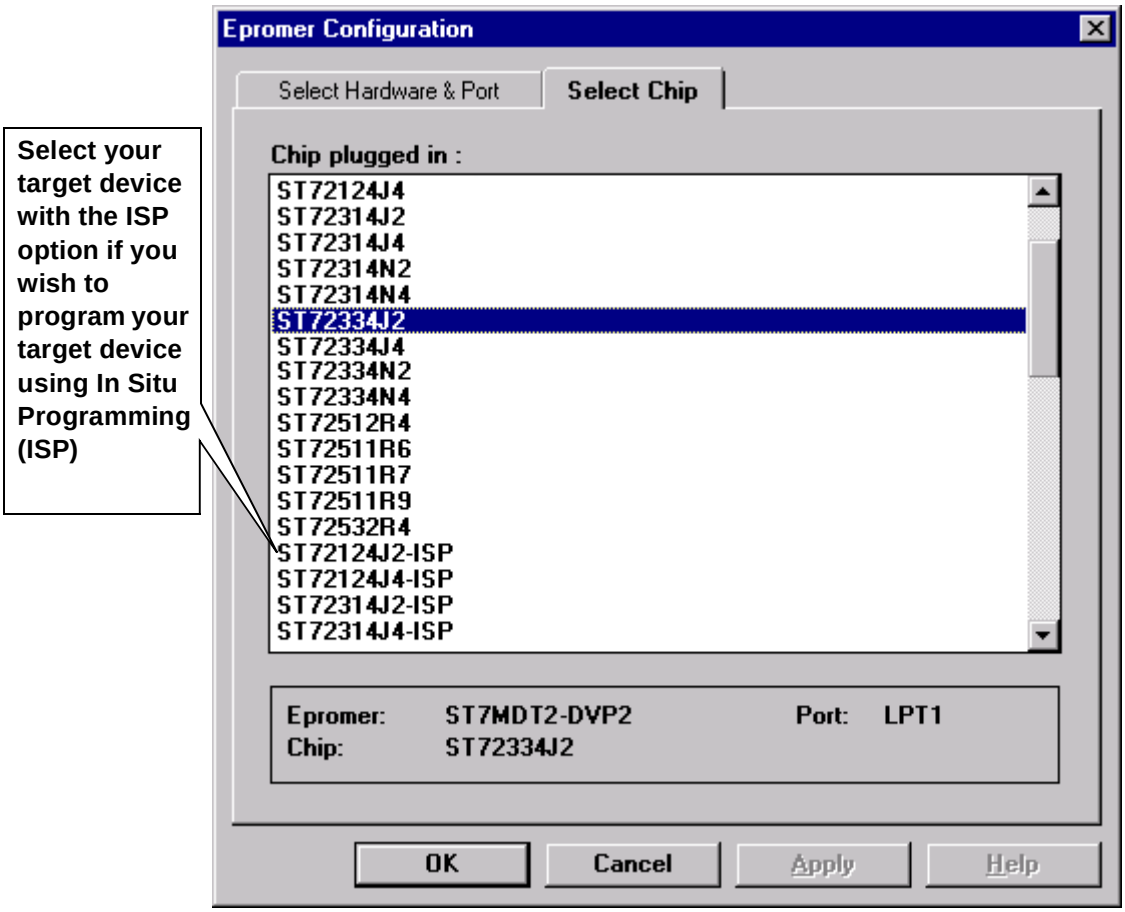


2 From the list, select the **ST7MDT2-DVP2** as the attached hardware.

3 Select the parallel port (LPT1 or LPT2) on your PC to which the development board is connected.

4 Click the ***Select Chip*** tab to display the list of the devices that can be programmed with this ST7MDT2-DVP2 development kit.

The list box shown below appears.



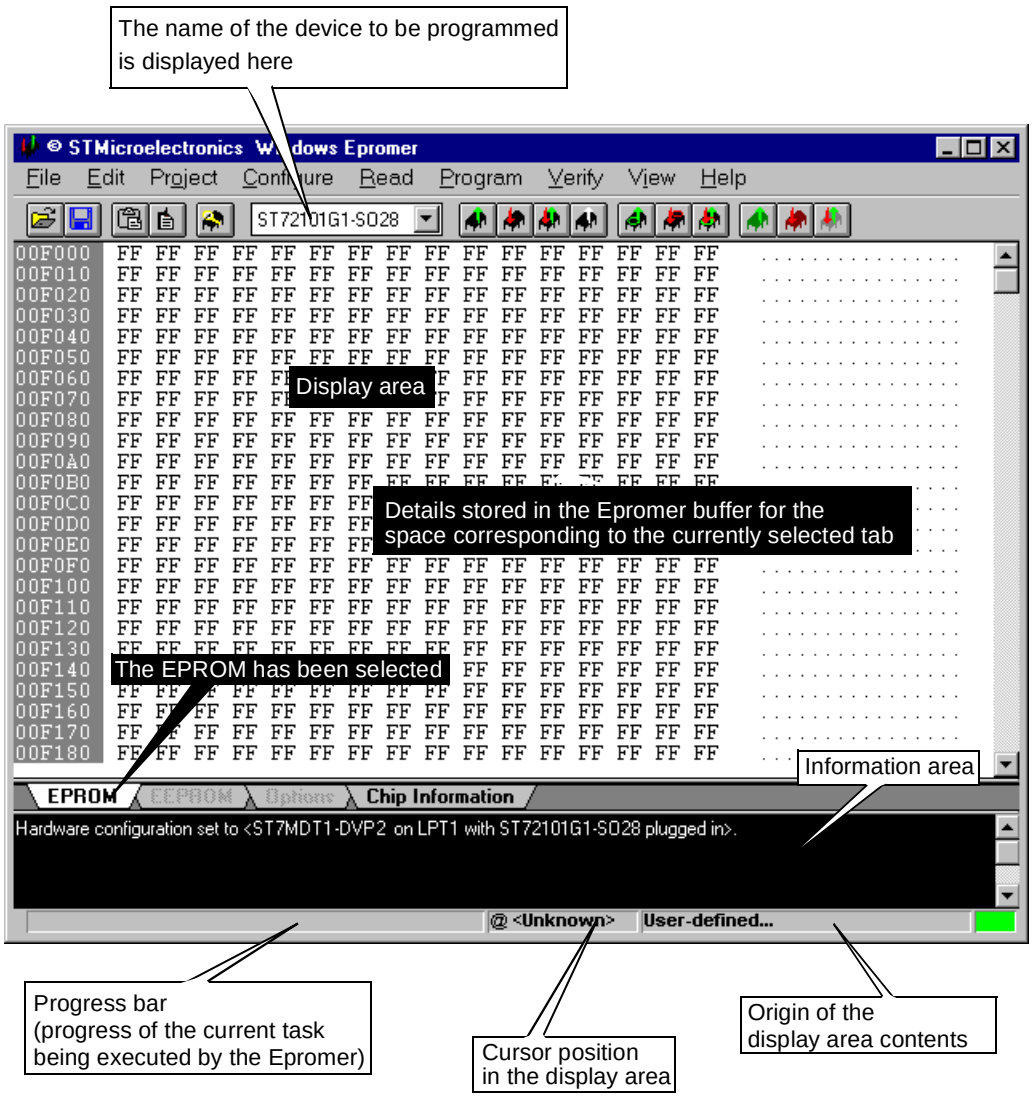
5 From the list shown in the above dialog box, select the device to be programmed.

Note: Only devices displayed with an ISP suffix can be programmed using the ISP functionality.

Click **OK** to confirm. The dialog box closes.

The memory mapping of the specified device now appears in the display area of the main window. It is made up of “FFs”, as one may expect, since programming has not taken place yet.

To view in turn the memory mapping of a selection of devices plugged in, open the configuration window again, then the ***Select Chip*** list box, and click **Apply**. The display area of the main window changes while the list box stays open, for you to choose another chip if necessary.



6 Start your programming session.

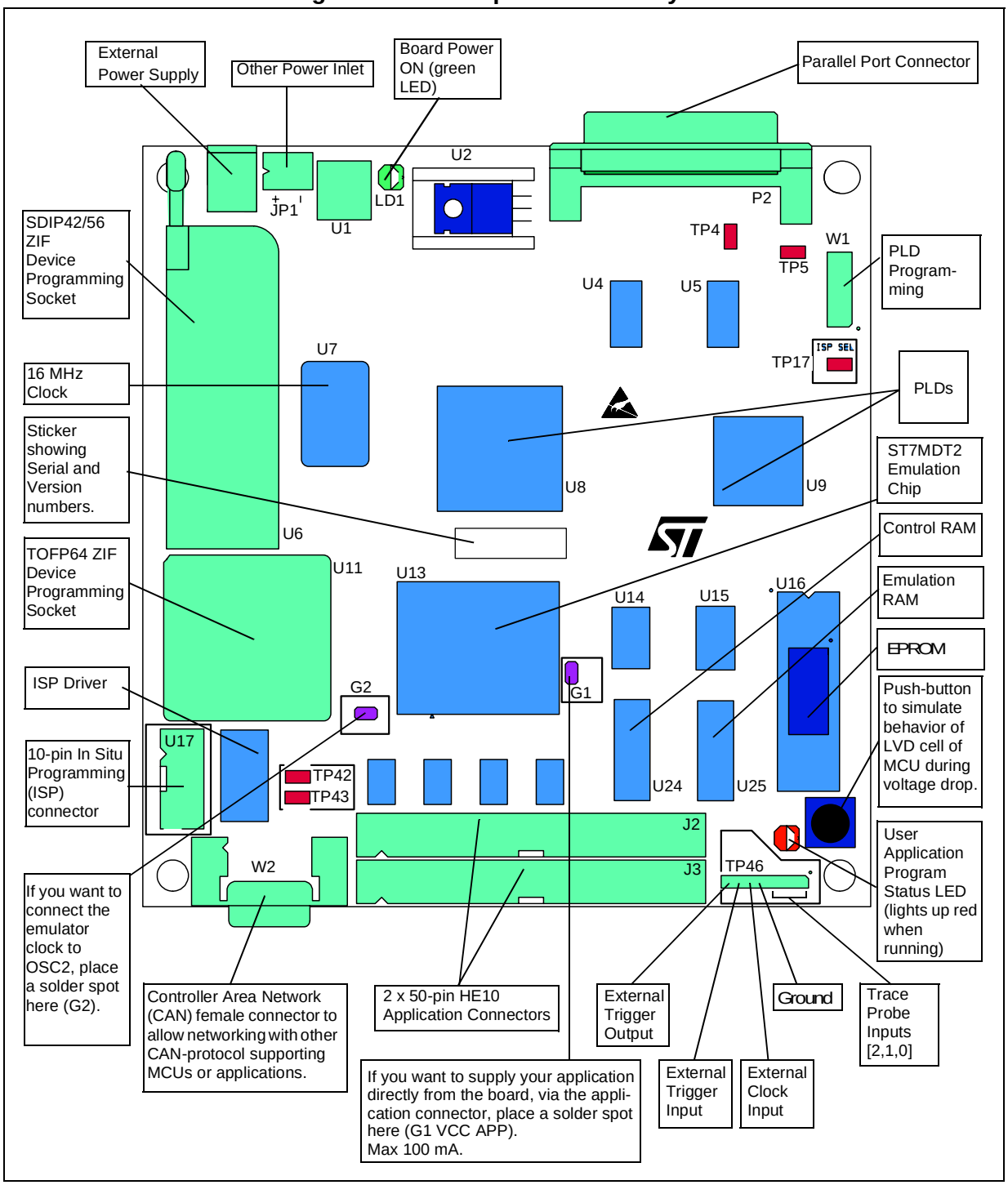
For more information on how to use the Windows Epromer, click the **Help** command in the main menu bar.

Note: *Refer to Section 4.2 on page 41 for details on which programming methods can be used with your target device.*

5 HARDWARE FEATURES

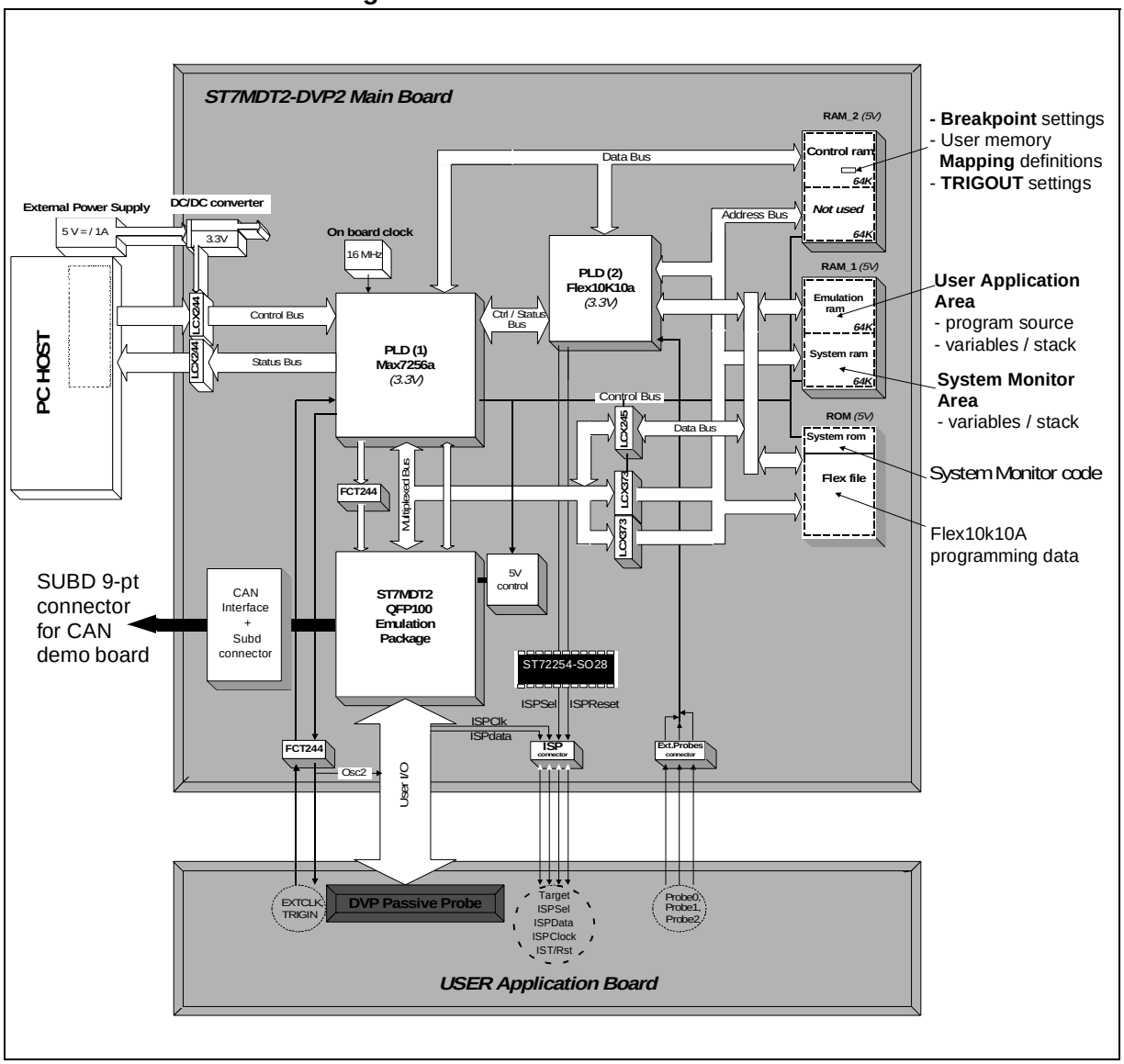
5.1 ST7MDT2-DVP2 development board layout

Figure 10: Development board layout



5.2 ST7MDT2-DVP2 emulation architecture

Figure 11: Emulation architecture



5.3 Link to PC

The ST7MDT2-DVP2 development board communicates with your PC via the P2 connector connected to the PC parallel port (LPT1 or LPT2).

*Note: The parallel port of your PC should have been configured (in the BIOS settings) with either the **Centronics**, **EPP**, **ECP** or **bidirectional** parallel port configurations.*

5.4 Power supply

A plug-in power supply pack is supplied with the ST7MDT2-DVP2 development kit to be connected to the P1 male jack connector. This power supply must be plugged into the appropriate AC source. Specific sales types indicate the corresponding mains AC voltage supported:.

Sales Type	AC Mains Voltage Supported
ST7MDT2-DVP2/EU	220 V
ST7MDT2-DVP2/UK	240 V
ST7MDT2-DVP2/US	110 V

Provided DC power specifications are as follows:

Voltage: 5 V

Current: 1 A

A complementary power supply inlet (ref.: JP1) is provided with the same specifications. When using this power supply, take care of the polarities marked nearby the two-point connector.

5.4.1 Supplying the application board

It is possible to supply your application board directly from the development board via the V_{DD} pin in the P3 probe connector, after a solder spot has been placed in G2 on the development board. This can be done only if your application consumption is <100 mA.

5.5 Jumper and solder point descriptions

The following table lists jumpers and solder points located on the development kit board, and whether they can be configured by the user, and if so, what they do.

Jumper name	User-configurable?	Description of utility
ISP-SEL (TP17)	NO	Use of this jumper could permanently damage the development board. This jumper has been removed from current versions of the ST7MDT2-DVP2.
ISP DRV PRG (TP4)	Under very specific circumstances.	On older ST7MDT2-DVP2 development kit boards, you may need to use this jumper to perform an ISP driver firmware patch as detailed in the application note AN1363/0401, entitled *Workaround to ISP Mode Limitation in ST7MDT1-DVP2 and ST7MDT2-DVP2*. DO NOT attempt to use this jumper for ISP mode programming! Use of this jumper is only supported following the detailed instructions given in AN1363/0401, available from ST's website at **http://mcu.st.com**. In currently released version of the development kit, this jumper has been removed, as the ISP mode limitation has been corrected.
JAM PROG (TP5)	NO	Use of this jumper could permanently damage the development board. This jumper has been removed from current versions of the ST7MDT2-DVP2.
CAN_SEL (TP42)	YES	Enable this jumper when you wish to connect a CAN demo board or any other CAN device to the DVP board. Refer to *Section 5.6: CAN features* on page 51.
5V_CAN (TP43)	YES	Enable this jumper **ONLY** if you wish to allow the CAN demo board's 5 V CAN supply to be available to the DVP board. Refer to *Section 5.6: CAN features* on page 51.
G1	YES	Soldering the two poles of this solder point together allows you to connect the clock to the OSC_OUT pin on the passive probe connector (see *Section 5.10: Clock* on page 63).
G2	YES	Soldering the two poles of this solder point together allows you to power your application board using the VDD pin in the passive probe connector (see *Section 5.4.1* on page 49)

5.6 CAN features

5.6.1 Overview

The ST7MDT2-DVP2 board comes with the necessary hardware—a line interface and terminator resistor—to allow you to connect it to a CAN network.

5.6.2 CAN peripheral features

The CAN cell is compliant with the version 2.0B passive of the BOSCH CAN specification. The user has three buffers (configurable as transmit or as receive) to handle CAN messages and two filters allowing to select the messages to be received. The baud rate can be configured up to 1 Mbit. For detailed information on the CAN peripheral, please refer to the CAN section of the data sheet of the corresponding ST725XX device.

5.6.3 CAN line interface

The board is fitted with a CAN line driver (L9615D) and terminated with a 120 Ohm resistor. This line driver is compliant with the high speed standard ISO 11898-2 and allows a baud rate up to 1 Mbit.

In order to easily connect the board to various third party board and tools, the pin assignment for the 9-pole Sub-D CAN connector is compliant with the CiA DS-102 - specification. This standard is widespread in automation and used by many CAN tools manufacturers.

Pin	Signal	Description
1	-	reserved
2	CAN_L	bus line dominant low
3	CAN_GND	CAN ground
4	-	reserved
5	CAN_SHLD	not available on DVP2
6	GND	optional ground
7	CAN_H	bus line dominant high
8	-	reserved
9	(CAN_V+)	optional CAN external supply

Table 4: CAN connector pin assignment

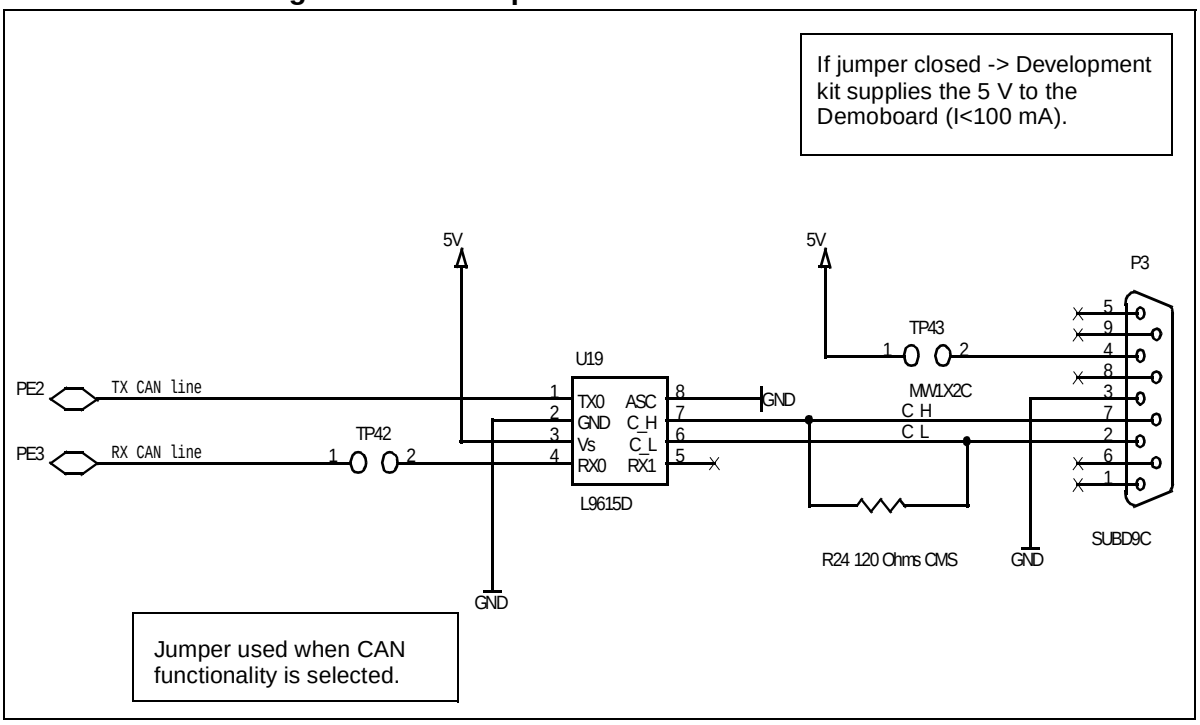
5.6.4 CAN jumper settings

The CAN connections on your ST7MDT2-DVP2 board are shown below in *Figure 12*.

If you wish to connect a CAN demo board to the development board, you must change the jumper settings on the development board (refer to *Figure 10* on page 47 to find the jumper pin locations).

- **Jumper CAN_SEL (ref.: TP42)**
  Enable this jumper when you wish to connect a CAN demo board or any other CAN device to the DVP board.
- **Jumper 5V_CAN (ref.: TP43)**
  Enable this jumper ONLY if you wish to allow the CAN demo board's 5 V CAN supply to be available to the DVP board.

Figure 12: Development kit CAN connections



5.6.5 Connecting a CAN demo board

You can quickly start developing your CAN application on the ST7MDT2-DVP2 using the CAN demonstration board to receive and transmit your application CAN messages. The following steps have to be performed:

1. Select one of the ST7 devices with CAN (ST72511, ST72512, ST72532).
2. Set the CAN_SEL jumper.

3. Connect the ST7MDT2-DVP2 CAN connector with the CAN connector of the demo board.
4. Set the 5 V CAN jumper if you want the ST7MDT2-DVP2 to supply the demo board with 5 V.

You can order the ST7 CAN demonstration board from your STMicroelectronics Microcontroller Development Tools Sales Support using the order code: ST7CAN-DEMO.

5.6.6 Connecting to a CAN network

Before connecting the ST7MDT2-DVP2 to another CAN node or network, please check the following points:

- Make sure that the CAN line drivers used by the other nodes in the network are compatible with the CAN high speed standard (ISO 11898-2).
- Verify that the pin assignment is compatible with the CiA DS102, see 1.3 table “CAN connector pin assignment”.
- Select one of the ST7 devices with CAN (ST72511, ST72512, ST72532).
- Configure the ST7 CAN cell baud rate according to the baud rate of the other nodes.
- Set the CAN_SEL jumper.
- Connect the ST7MDT2-DVP2 CAN connector to the CAN network.

5.7 Pin descriptions and package footprints

5.7.1 Pin descriptions

You may connect an application board to the ST7MDT2-DVP2 development board for evaluation or debugging in linked emulation mode. Signals are transmitted via a **passive probe** to be plugged in the application board at the location of the emulated device.

Tables 4 to 7 show the pin number assignment for all the passive probes’ connectors on the application board. The development kit board J1 and J2 connectors schematics which follow can help you locate the corresponding signals.

Table 5: SDIP42 passive probe pin assignment

Probe Pin No.	Pin Name/Description	Probe Pin No.	Pin Name/Description
1	PB4	22	PC6/SCK/ISPCLK
2	AIN0/PD0	23	PC7/$\overline{SS}$
3	AIN1/PD1	24	PA3
4	AIN2/PD2	25	VDD(1)
5	AIN3/PD3	26	VSS
6	AIN4/PD4	27	PA4
7	AIN5/PD5	28	PA5
8	VDD[1]	29	PA6
9	VSS	30	PA7
10	CLKOUT/PF0	31	ISPSEL
11	BEEP/PF1	32	$\overline{RESET}$
12	PF2	33	VSS
13	OCMP1_A/PF4	34	OSC2[2]
14	ICAP1_A/PF6	35	OSC1
15	EXTCLK_A/PF7	36	VDD(1)
16	PC0/OCMP2_B	37	PE0/TD0
17	PC1/OCMP1_B	38	PE1/RDI
18	PC2/ICAP2_B	39	PB0
19	PC3/ICAP1_B	40	PB1
20	ISPDATA/PC4/MISO	41	PB2
21	PC5/MOSI	42	PB3

1) The application board may be supplied by the development kit only if a solder spot is placed on G2. 5 Volt/100mA are supplied. Take care that your application runs under this specification.

2) The emulator is connected to OSC2 (OSCOUT) only when a solder spot is placed at G1 on the development board.

Table 6: SDIP56 passive probe pin assignment

Probe Pin No.	Pin Name/Description	Probe Pin No.	Pin Name/Description
1	PB4	29	PC6/SCK/ISPCLK
2	PB5	30	PC7/$\overline{SS}$
3	PB6	31	PA0
4	PB7	32	PA1
5	AIN0/PD0	33	PA2
6	AIN1/PD1	34	PA3
7	AIN2/PD2	35	VDD(1)
8	AIN3/PD3	36	VSS
9	AIN4/PD4	37	PA4
10	AIN5/PD5	38	PA5
11	AIN6/PD6	39	PA6
12	AIN7/PD7	40	PA7
13	VDD1	41	ISPSEL
14	VSS	42	$\overline{RESET}$
15	CLKOUT/PF0	43	VSS
16	BEEP/PF1	44	OSC22
17	PF2	45	OSC1
18	OCMP1_A/PF4	46	VDD(1)
19	ICAP1_A/PF6	47	PE0/TD0
20	EXTCLK_A/PF7	48	PE1/RDI
21	VDD(1)	49	PE4
22	VSS	50	PE5
23	PC0/OCMP2_B	51	PE6
24	PC1/OCMP1_B	52	PE7
25	PC2/ICAP2_B	53	PB0
26	PC3/ICAP1_B	54	PB1
27	ISPDATA/PC4/MISO	55	PB2
28	PC5/MOSI	56	PB3

1) The application board may be supplied by the development kit only if a solder spot is placed on G2. 5 Volt/100mA are supplied. Take care that your application runs under this specification.

2) The emulator is connected to OSC2 (OSCOUT) only when a solder spot is placed at G1 on the development board.

Table 7: TQFP64 passive probe pin assignment (CAN-less family)

Probe Pin No.	Pin Name/Description	Probe Pin No.	Pin Name/Description
1	PE4	33	VDD(1)
2	PE5	34	VSS
3	PE6	35	PC0/OCMP2_B
4	PE7	36	PC1/OCMP1_B
5	PB0	37	PC2/ICAP2_B
6	PB1	38	PC3/ICAP1_B
7	PB2	39	PC4/MISO/ISPDATA
8	PB3	40	PC5/MOSI
9	PB4	41	PC6/SCK/ISPCLK
10	PB5	42	PC7/$\overline{SS}$
11	PB6	43	PA0
12	PB7	44	PA1
13	AIN0/PD0	45	PA2
14	AIN1/PD1	46	PA3
15	AIN2/PD2	47	VDD(1)
16	AIN3/PD3	48	VSS
17	AIN4/PD4	49	PA4
18	AIN5/PD5	50	PA5
19	AIN6/PD6	51	PA6
20	AIN7/PD7	52	PA7
21	VDD[1]	53	ISPSEL
22	VSS	54	$\overline{RESET}$
23	VDD(1)	55	Not Used
24	VSS	56	Not Used
25	CLKOUT/PF0	57	VSS
26	BEEP/PF1	58	OSC2[2]
27	PF2	59	OSC1
28	Not Used	60	VDD(1)
29	OCMP1_A/PF4	61	PE0/TD0
30	Not Used	62	PE1/RDI
31	ICAP1_A/PF6	63	Not Used
32	EXTCLK_A/PF7	64	Not Used

1) The application board may be supplied by the development kit only if a solder spot is placed on G2. 5 V/100 mA are supplied. Take care that your application runs under this specification.

2) The emulator is connected to OSC2 (OSCOUT) only when a solder spot is placed at G1 on the development board.

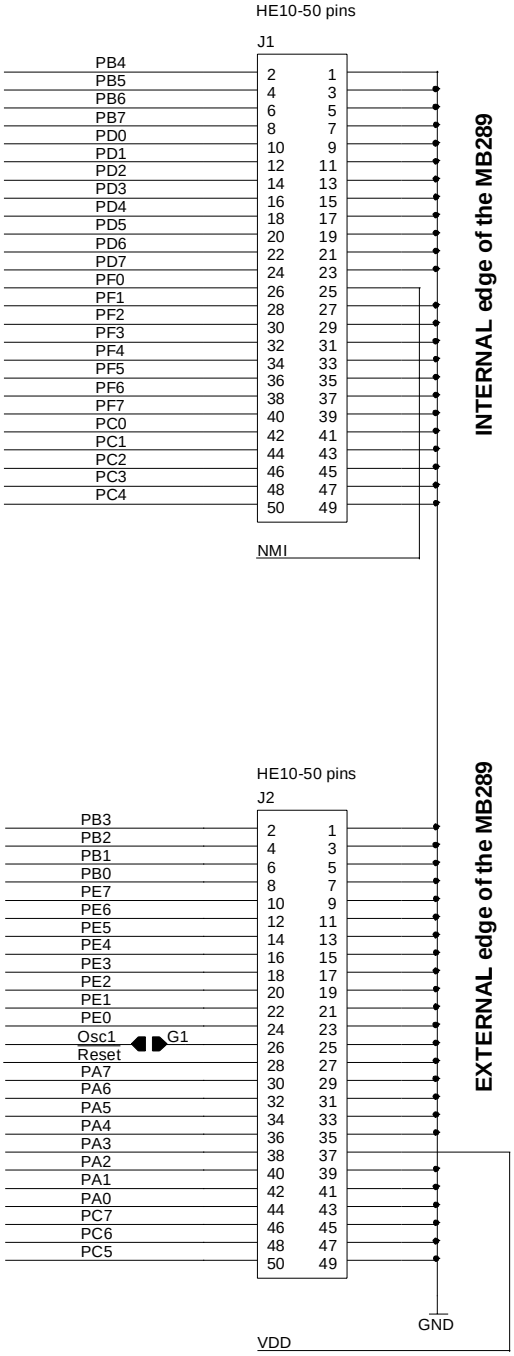
Table 8: TQFP64 passive probe pin assignment (CAN family)

Probe Pin No.	Pin Name/Description	Probe Pin No.	Pin Name/Description
1	PE4	33	VDD(1)
2	PE5	34	VSS
3	PE6	35	PC0/OCMP2_B
4	PE7	36	PC1/OCMP1_B
5	PWM3/PB0	37	PC2/ICAP2_B
6	PWM2/PB1	38	PC3/ICAP1_B
7	PWM1/PB2	39	PC4/MISO
8	PWM0/PB3	40	PC5/MOSI
9	ARTCLK/PB4	41	PC6/SCK
10	ARTIC1/PB5	42	PC7/$\overline{SS}$
11	ARTIC2/PB6	43	PA0
12	IDCELL/PB7	44	PA1
13	AIN0/PD0	45	PA2
14	AIN1/PD1	46	PA3
15	AIN2/PD2	47	VDD(1)
16	AIN3/PD3	48	VSS
17	AIN4/PD4	49	PA4
18	AIN5/PD5	50	PA5
19	AIN6/PD6	51	PA6
20	AIN7/PD7	52	PA7
21	VDD[1]	53	VPP/TEST
22	VSS	54	$\overline{RESET}$
23	VDD(1)	55	Not Used
24	VSS	56	NMI
25	PF0	57	VSS
26	PF1	58	OSC2[2]
27	PF2	59	OSC1
28	OCMP2_A/PF3	60	VDD(1)
29	OCMP1_A/PF4	61	PE0/TD0
30	ICAP2_A/PF5	62	PE1/RDI
31	ICAP1_A/PF6	63	PE2/CANTX
32	EXTCLK_A/PF7	64	PE3/CANRX

1) The application board may be supplied by the development kit only if a solder spot is placed on G2. 5 V/100 mA are supplied. Take care that your application runs under this specification.

2) The emulator is connected to OSC2 only when a solder spot is placed at G1 on the development board.

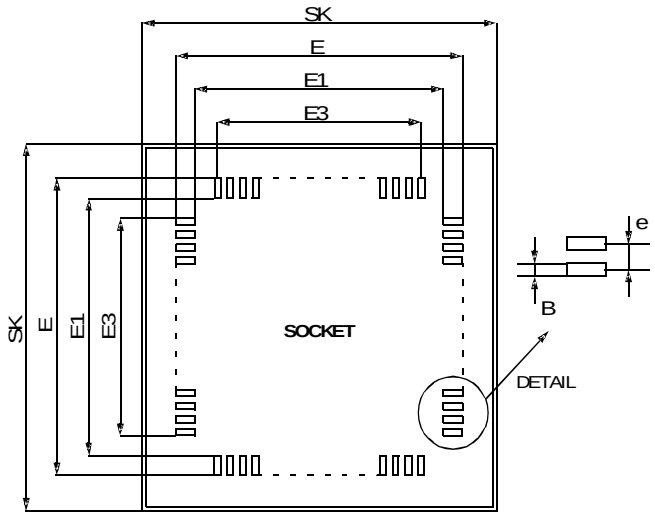
Figure 13: J1 and J2 connector pins functions



5.7.2 QFP64/TQFP64 footprint discrepancy issues

Some of the emulated devices have a TQFP64 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 QFP64 footprints that are not compatible with TQFP64 copper traces.

For this reason, when designing your board, plan to have a double trace compatible with both QFP64 Yamaichi sockets and TQFP64 copper traces. A specification for a compatible footprint is 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 package.



Dim	mm			inches		
	Min	Typ	Max	Min	Typ	Max
B	0.35	0.45	0.50	0.014	0.018	0.020
E	20.80			0.819		
E1		14.00			0.551	
E3	11.90	12.00	12.10	0.468	0.472	0.476
e	0.75	0.80	0.85	0.029	0.031	0.033
SK*		26			1.023	
	Number of Pins					
N	64 (4x16)					

* SK: Plastic socket overall dimensions.

Figure 14: TQFP64 device and emulation probe compatible footprint

5.7.3 Device mode for TQFP64 packages

Once your programmed has been debugged, you may want to program some devices and test them directly in your application without using the DVP. This mode is called ‘Device Mode’.

Because Yamaichi sockets are QFP64, you will not be able to insert a TQFP64 device and ensure a perfect contact.

However, you can order a special TQFP64 device adapter (ref.: DB379) to allow you to connect an actual TQFP64 device to the Yamaichi socket (i.e. without soldering the MCU directly to the footprint).

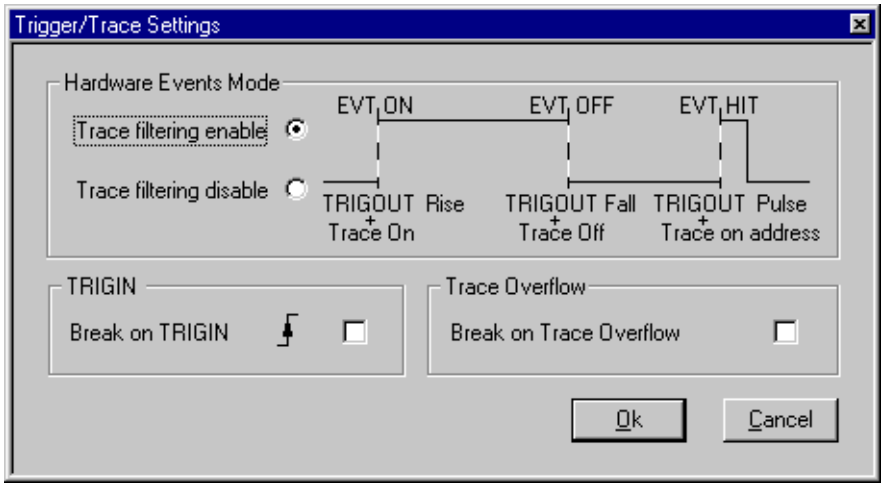
You can order this TQFP64 device adapter (ref.: DB379) from your nearest STMicroelectronics Microcontroller Development Tools Sales Support.

On the upper face of this adapter you will find a low insertion socket for your programmed device. A chamfer on the adapter marks the location of pin 1 and must be aligned with the pin 1 chamfer on the Yamaichi socket base.

On the upper face of this adapter you will also find 64 numbered pins enabling access to the device pin voltage. These pins are numbered as they appear on the device user datasheets.

5.8 Trigger/trace settings

The Trigger/Trace Settings dialog box (shown below), allows you choose a variety of options:



- Select between two hardware event modes: trace filtering enabled or trace filtering disabled. In the first mode, the recording of the trace buffer is controlled using the same hardware events as those used to control the signals from the external output trigger. See below for details on the external output trigger, TRIGOUT.
- Choose to enable the ***Break on TRIGIN*** option. For details on the input trigger, TRIGIN, see below.
- Choose to enable the ***Break on Trace Overflow*** option. With this option enabled, once the trace buffer is full, the program is stopped.

You may access the **Trigger/Trace Settings** dialog box from the Main STVD7 Menu by selecting ***Tools>Trigger/Trace Settings***, or by clicking the Trigger/Trace Settings icon in the **Tools** toolbar.

5.8.1 External output trigger (TRIGOUT)

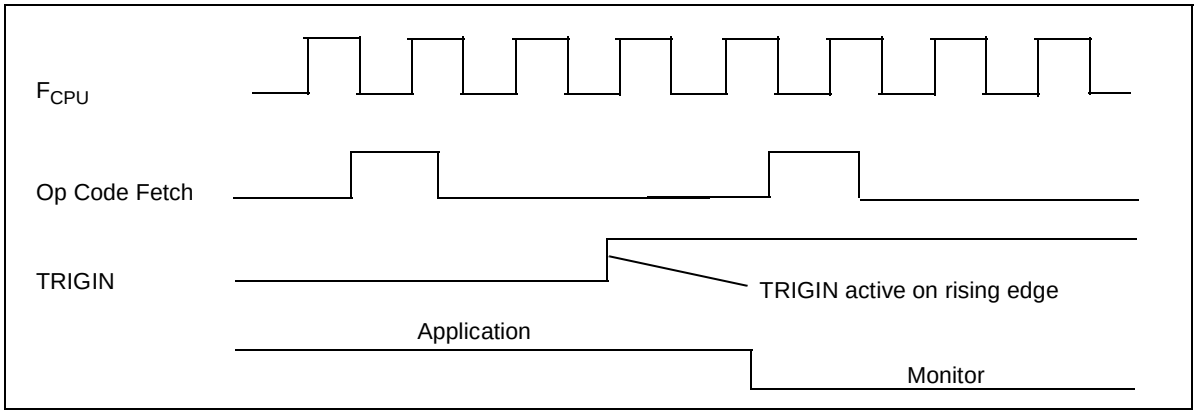
The evaluation board of the ST7MDT2-DVP2 development board features a special outlet (called **TRIGOUT**,) through which an external signal can be triggered out. The TRIGOUT pin is located on the development board next to the passive probe flat connector (Ref.:TP46).

From the Trigger/Trace Settings dialog box, you can choose the hardware event mode for the external signal (***EVT_OFF/EVT_ON*** or ***EVT_HIT***).

5.8.2 Input trigger (TRIGIN)

The ST7MDT2-DVP2 Development Board provides a special inlet (**TRIGIN**, Ref.:TP46) that can be used to transmit a signal to stop the execution of your application upon the occurrence of an external event (***Break on TRIGIN***). The ***Break on TRIGIN*** option is available in the **Trigger/Trace Settings** dialog box shown above. If this option is selected, on reception of a rising edge signal from the TRIGIN pin, the program is stopped after the execution of the current instruction.

A schematic diagram is shown below:



Caution: ***TRIGIN*** *and* ***PROBE****[0..2] information are only available in the* ***Trace Window*** *if the program runs at addresses above the hexadecimal value 2000H. If it is below 2000H, the characters XXX will appear in the corresponding column.*

5.9 Hardware events

Hardware Events are defined events used to control the trigger signal outputs and trace buffer recording.

There are three types of hardware event:

- **Event On (EVT_ON)**: The address where the event begins.
- **Event Off (EVT_OFF)**: The address where the event ends.
- **Event Hit (EVT_HIT)**: The event is active for the cycles in which one particular address is accessed.

For information on how to insert hardware events, refer to the online help available with the STVD7.

For more information on how to use hardware events to control the external output trigger (TRIGOUT) signal or the trace buffer filtering, refer to *Section 5.8* on page 60.

5.10 On-chip peripherals

You can configure certain on-chip peripherals in ST7 Visual Debug' s **MCU Configuration** dialog box (refer to *Section 3.6* on page 22) so that the emulator accurately emulates your target device.

The following options are available on all supported target devices:

Timer_A and Timer_B

Two 16-bit timers, Timer A and Timer B, are available. They consist of a 16-bit free-running counter driven by a programmable prescaler. Both timers feature output compare, pulse width modulation (PWM) and input capture functions with associated registers.

You can choose between two options: **Enabled** and **Disabled**. If you choose the **Disabled** option, the counter will run while the application runs, but when a breakpoint is encountered, the counter will stop. If you choose the **Enabled** option, the counter will continue to run even if a breakpoint is encountered.

Note: The EXTCLK_A pin of the timer is not frozen when the program is stopped.

PWM

The options for this peripheral (when available on the emulated chip) are the same as for Timer_A and Timer_B (above).

Clock

You may choose the clock type (such as **on-board** or **external**, for example) as a microcontroller configuration option.

Note: The clock types available can vary depending on the target device. Refer to Section 5.11 on page 64 for an explanation of clock types available).

The development board is shipped with a 16 MHz (TTL) on-board clock.

You may also use an external clock (TTL-compatible, with a maximum frequency of 16 MHz) whose signal is supplied via the mini wrapping pin “EXTCLK” located on the board next to the passive probe flat connector (ref.: TP46).

Note: See Section 5.11: Emulation functional limitations and discrepancies on page 64 for functional discrepancies involving clock sources and clock frequencies.

Note that the board cannot operate with clock signals received from the application board via the OSC1 pin of the probe. You must use the EXTCLK inlet instead.

However, the application board can use the development board clock via the OSC2 pin of the probe. In this case you must place a solder spot on G1 (see *Table 5* on page 54).

Watchdog

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

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

Halt and Watchdog

There are two options: ***Reset*** or ***No Reset***. If this option is set to Reset, a chip reset will be performed each time the Watchdog is enabled and a Halt instruction is encountered in the executable code. If this option is set to No Reset, no chip reset will be performed.

5.11 Emulation functional limitations and discrepancies

Not all development kit features are available or applicable for all target MCUs. Supported target MCUs are divided into two groups:

Group 1 (based on ST72C334)	Group 2 (based on ST72511)
ST72124 J2/J4 ST72314 J2/J4 ST72314 N2/N4 ST72334 J2/J4 ST72334 N2/N4	ST72532 R4 ST72311 R6/R7/R9 ST72512 R4 ST72511 R6/R7/R9

The following is a list of functional limitations and discrepancies applicable when using the development kit as an emulator (as compared to the actual target device features):

Emulation Function/Feature	Target Device Group	Limitation or Discrepancy
CPU Clock Options	**Groups 1 & 2**	You are required to select the clock option for your target device when using the development kit with the STVD7 (refer to *page 63* for instructions on how to choose clock options). **For target devices in Group 1 and 2**, you can choose between two clock options: **external** or **on-board**. • The **external** option refers to a clock external to the development board (for example, a clock on the user application board), connected via the EXTCLK pin. • The **on-board** option refers to the on-board 16 MHz clock generator provided on the development board. The **Internal RC**, **External RC** and **xrd Resonator** options are not available on the MDT2-DVP2 development board.
External Clock Frequency	**Groups 1 & 2**	When you use the external clock option (via the EXTCLK pin) f_{CPU} must be greater or equal to 600 kHz for full functionality (see *Operation at low CPU frequencies* below). The maximum frequency of the external clock is 16 MHz.

Emulation Function/Feature	Target Device Group	Limitation or Discrepancy
Slow Mode	**Groups 1 & 2**	In *slow mode* (Miscellaneous Register slow mode bit = 1), the EXTCLK frequency must take values between 8 MHz and 16 MHz.
Operation at low CPU frequencies	**Groups 1 & 2**	Operation of the development kit at low f_{CPU} values ($f_{CPU} \leq 600$ kHz) may result in malfunctions and/or STVD7 crashes when you attempt to step through the application. At $f_{CPU} \leq 600$ kHz, you may perform Run and Continue commands, but if you use the Step commands, you may encounter the following error message: `"Error: gdi-error in dvp:connection failure.verify input/output cable"` This message occurs because the low f_{CPU} results in slow communication between the host PC and the development board—so slow that the connection times out and disrupts the debugging sesstion. This restriction in the value of f_{CPU} is valid both for *normal mode* operation (Miscellaneous Register slow mode bit = 0) or *slow mode* operation (Miscellaneous Register slow mode bit = 1).
Temperature Tolerance	**Groups 1 & 2**	Independent of the target device used, the very large temperature tolerance scope of the ST72334 and ST7251x family target devices (-40°C to +85°C) is not applicable to the development board. The development board has been designed to function at ambient temperature.
Low Voltage Detection	**Groups 1 & 2**	The Low Voltage Detection (LVD) feature is **supported only when the supply voltage is 5 V**. It is implemented using the on-board push button (see *Figure 10* on page 47) which causes the chip to reset. Your application will be able to detect that the reset was caused by LVD by reading a flag. The flag location for Group 1 devices is the CRSR register. It does not exist for Group 2 devices.

Emulation Function/Feature	Target Device Group	Limitation or Discrepancy
Supply Voltage	**Groups 1 & 2**	Only a development board application supply voltage of 5 V is supported as opposed to any voltage in the 3 to 5.5 V range for the actual target devices.

APPENDIX A: EMC CONFORMITY AND SAFETY REQUIREMENTS

This development board 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 have its shield connected to ground.
- All ferrites provided with the development kit must be attached as described in the hardware installation instructions of the relevant user manual(s).
- Your development board must be placed on a conductive table top, made of steel or clean aluminumor covered by an antistatic surface (superficial resistivity equal to or higher than 0.5 MΩ/cm^2), grounded through a ground cable(conductive cable from protected equipment to ground isolated through a 1 MΩ resistor placed in series).
  All manipulation of finished goods be made at such a grounded worktable.
- The worktable must be free of all non-antistatic plastic objects.
- An antistatic floor covering grounded through a conductive ground cable (with serial resistor between 0.9 and 1.5 MΩ) should be used.
- 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 development board, you 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...	THEN...
Error Message (when starting the STVD7 for DVP): **"Connection Error (LPT1): Interconnection failure. Verify your input/output cable**."	Ensure that: • The parallel cable is connected directly between the development board 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 development board is not recommended. • The development board is powered on. • The parallel cable used is the one supplied with the kit by STMicroelectronics. • STVD7 and Windows Epromer are not running at the same time. If this is the case then shut down both and ensure that only one or the other is running at any one time. 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*, below.
Error Message (during STVD7 session): **"Emulator power off has been detected**."	Ensure that your development board is powered on and shutdown and restart your STVD7 session.
ISP does not work.	Ensure that: • The target MCU mounted on your application board has a power supply of 5 V. • The Reset pin (pin 6) of the ISP connector is not overloaded. If this is the case set the resistor and capacitor values of the reset pin circuitry to the recommended values given in the target MCU user data sheet. On older ST7MDT2-DVP2 development kit boards, you may need to perform an ISP hardware/software patch as detailed in the application note AN1363/0401, entitled *Workaround to ISP Mode Limitation in ST7MDT1-DVP2 and ST7MDT2-DVP2*, available from ST's website at: **http://mcu.st.com**

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 ST7MDT2-DVP2 development board.

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.

5.12 Running the hardware test

The **Hardware Test** in the STVD7 for DVP lets you check that your development board 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 77).

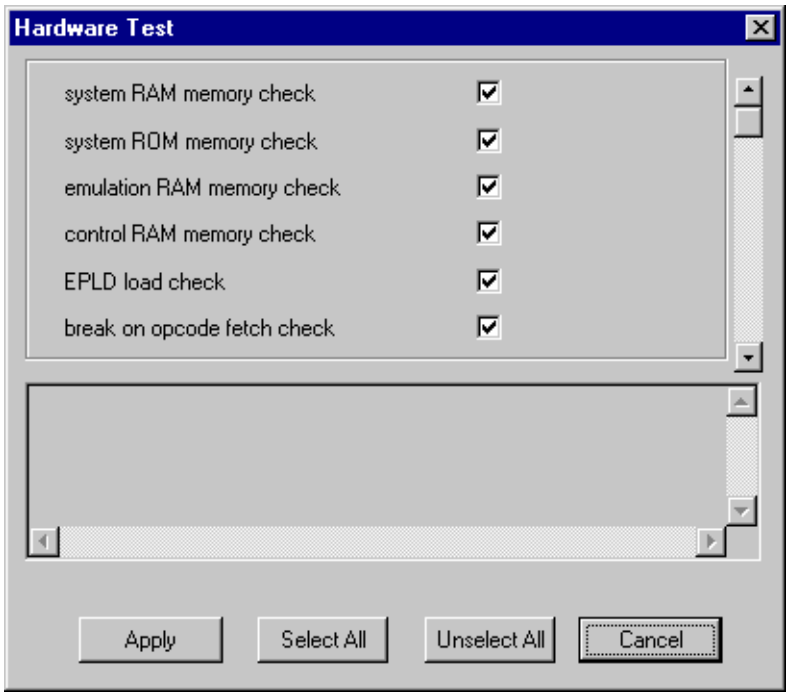
You may open the **Hardware Test** dialog box by:

- selecting, from the Main Menu, ***Emulator>Hardware Test***

- clicking on the Hardware Test icon in the **Emulator** toolbar.

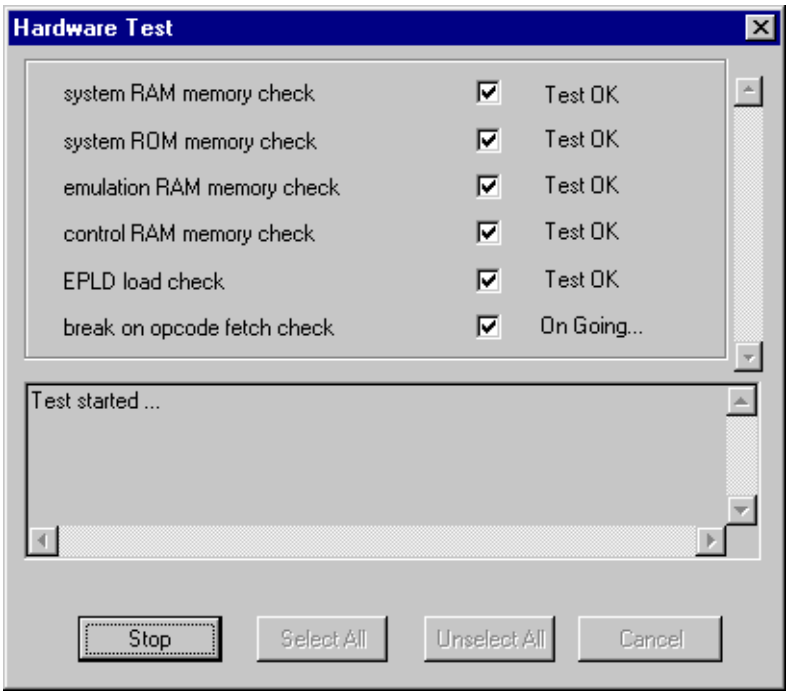
Caution: *Be cautious in performing a Hardware Test on the emulator while an application is open. The opened application WILL BE corrupted by the hardware testing process. If you find that your application has been corrupted, simply close the application, and reopen it.*

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



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 at right:



APPENDIX C: GLOSSARY

CAN

The Controller Area Network (CAN) protocol is becoming more and more widely accepted in Europe and throughout the world. It enables the creation of networks inside a vehicle or an industrial system with high tolerance to error in noisy environments. The Controller area network peripheral conforms to the CAN specification 2.0 active and 2.0B passive. The interface has three 10-bit transmit/ receive buffers and two 12-bit message acceptance filters. The Baud rates are programmable up to 1Mbit/s.

Development Board

The main component of the development kit, the development board consists primarily of a CPU which is capable of emulating the family of MCUs supported by the development kit, and a number of peripherals that allow you to perform debugging and programming functions. The development board contains a parallel port to communicate with your PC and so allow the running and debugging of applications designed for your target MCU. There are TRIGIN, TRIGOUT and Trace Probe pins to respectively input and output signals. There are also a variety of means by which you can use the development board to program target MCU devices: via the on-board socket, via the In Situ Programming (ISP) port.

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.

EEPROM

Electrically Erasable Programmable Read-Only Memory. A non-volatile type of memory that can be erased and reprogrammed by program instructions. Since no special power supplies or ultra-violet light source is needed, the contents of this kind of memory can be changed without removing the MCU from the application system.

EPP

Enhanced parallel port communication standard.

EPROM

Erasable Programmable Read-Only Memory. A non-volatile type of memory that can be erased by exposure to an ultra-violet light source. MCUs that have EPROM are easily recognized because the package has a quartz window to allow exposure to the UV light. If the EPROM MCU is packaged in an opaque plastic package, it is called a “one-time programmable” OTP MCU because there is no way to expose the EPROM to a UV light source.

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.* (Refer to the ST7 MCU FAMILY DATABOOK or the datasheets provided on the “MCU on CD” CD-ROM).

LVD

Low Voltage Detection. This is a feature available on all of the ST7 MCUs supported by the ST7MDT2-DVP2. A LVD push button on the development board allows you to simulate what occurs when the MCU detects that the supply voltage is below a given threshold.

ISP

In Situ Programming. Provided you have an In Situ Programming connector on the application board containing the target device, you can use the ST7MDT2-DVP2 ISP functionality and the Windows Epromer to directly program the target device. Note that not all ST7 MCUs support the ISP functionality. Refer to *Section 4.2.2* on page 41 for more details.

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.

OTP

One Time Programmable. Also referred to as OTPROM (One Time Programmable Read-Only Memory). A non-volatile type of memory that can be programmed but

cannot be erased. An OTP ROM is an EPROM MCU that is packaged in an opaque plastic package—it is called a one-time programmable MCU because there is no way to expose the EPROM to a UV light source.

Passive Probe

A printed card having connector pins that allow you to connect the ST7MDT2-DVP2 to the MCU socket of the user application board. Using the passive probe allows the development board to emulate a target device embedded in your application. The passive probe is connected to the development board by two flat 50-pin cables.

PLD

Programmable Logic Device.

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.

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, the ST7MDT2-DVP development kit or an HDS 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. It should include a socket for inserting the ST7 device or the passive probe.

ZIF Socket

Zero Insertion Force Socket. This type of programming socket is mounted directly on the development board. To program an MCU, you insert it into the appropriate socket (i.e. the SDIP56/SDIP42 combo socket or the TQFP64 ZIF socket).

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 development kit.

2 Date of purchase.

3 Order Code: Refer to the side of your development 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 a label on the developmentboard.

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

Yamaichi sockets

You can order additional Yamaichi QFP sockets directly from Yamaichi at:

http://www.yamaichi.de/Pu/quad_flat_pack/spec/a21-ic149.htm

Index

C

CAN interface
- connecting a CAN demo board 51

clock
- external 63
- internal 63
- normal mode 65
- option limitations 64
- slow mode 65
- type 63

complementary power supply 49

D

development board
- connecting to PC 12
- definition of 73
- eproming architecture 40
- jumpers and solder points 50
- layout 47

development kit
- as device programmer 40
- delivery checklist 11
- emulation architecture 48
- emulation features 47
- emulation limitations/discrepancies 64
- functions of 6
- system requirements 11

E

ECP
- definition of 73

EMC
- compliant environment 13

emulation limitations/discrepancies 64

emulator kit
- installing software for 15
- software and documentation for 8

Epromer

EXTCLK pin 63

F

finished goods
- manipulation of 67
- safety requirements 67

Footprint for TQFP64 devices 42

H

hardware
- events (see STVD7)
- installing 12
- linking to PC 48
- power supply 49
- test 70

I

In Situ Programming (see ISP)

installation
- STVD7 15

ISP 41
- definition of 74

J

jumper
- descriptions 50
- forbidden user settings 13, 50
- settings for CAN 51

L

load
- binary files 26

low voltage detection
- limitations of 65

M

MCU
- supported types 5

MCU configuration 33

MCU memory
- configuring 35
- types 35

Miscellaneous register 65

O

OTP
- definition of 74

Index

P

parallel port
- PC ... 48
- troubleshooting connection problems ... 70

passive probe
- definition of ... 75
- general configuration ... 6
- parts reference ... 11
- pinout ... 53

PC parallel port ... 48
peripherals
- configuring target ... 34

pin descriptions ... 53
power supply ... 49
- complementary ... 49

programming
- device installation ... 42
- eproming architecture ... 40

project settings
- modifying ... 28

Q

QFP64 Probes ... 14
QFP64/TQFP64 ... 59

R

ROM size ... 35

S

Sales types ... 49
SDIP42 devices ... 42
SDIP56 devices ... 42
slow mode ... 65
software
- updates ... 78

solder point descriptions ... 50
ST7 Visual Debug (see STVD7)
STVD7
- about ... 17
- build context ... 32
- contexts ... 32
- creating a workspace ... 22
- debug mode ... 32
- hardware events ... 62
- installing ... 15
- main features ... 17
- MCU configuration ... 33
- opening binary files ... 26
- opening workspaces ... 24
- supported application files ... 19
- supported toolchains ... 19
- switching between contexts ... 33
- toolchain paths ... 16
- trigger/trace settings ... 60
- workspaces ... 18

supply voltage
- limitations of ... 66

support
- contact numbers for ... 77
- for development kit ... 77

T

target device
- definition of ... 75
- groups of supported ... 64
- list of supported ... 5
- programmable memory types ... 39
- programming methods ... 39

temperature tolerance
- limitations of ... 65

TQFP64 ... 59
TQFP64 devices ... 42
trigger/trace settings (see STVD7)
triggers
- external output (TRIGOUT) ... 61
- input (TRIGIN) ... 61

TRIGIN ... 61
TRIGOUT ... 61
troubleshooting ... 69
- connection error ... 69

U

user application board
- definition of ... 75

uses for development board ... 6

W

Windows Epromer

configuring 44
getting started 43
Winee (see Windows Epromer)
workspaces
creating new..... 22
saving..... 30

Z

Zero insertion force (ZIF) socket 41, 42
definition of 76

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