ST7MDT1-DVP2
Development Kit
User Manual

Release 1.1

June 2000

Table of Contents

Table of Contents

1 INTRODUCTION

Thanks for choosing the ST7MDT1-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 ST7MDT1-DVP2 Development Kit for the ST72254 series 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 63.*

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

The ST7MDT1-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 37).

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 ST7 MDT1-DVP2:

Supported Devices
ST 72254 G1 ST 72254 G2
ST 72216 G1
ST 72215 G2
ST 72104 G1 ST 72104 G2
ST 72251 G1 ST 72251 G2
ST 72213 G1
ST 72212 G2
ST 72101 G1 ST 72101 G2

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 ST7MDT1-DVP2 Development Kit in an ST7 MCU Emulator configuration. *Figure 2* shows the development board of the ST7MDT1-DVP2 Development Kit in an ST7 MCU Programming Board configuration.

Figure 1: Using the Development Board as an ST7 MCU Emulator
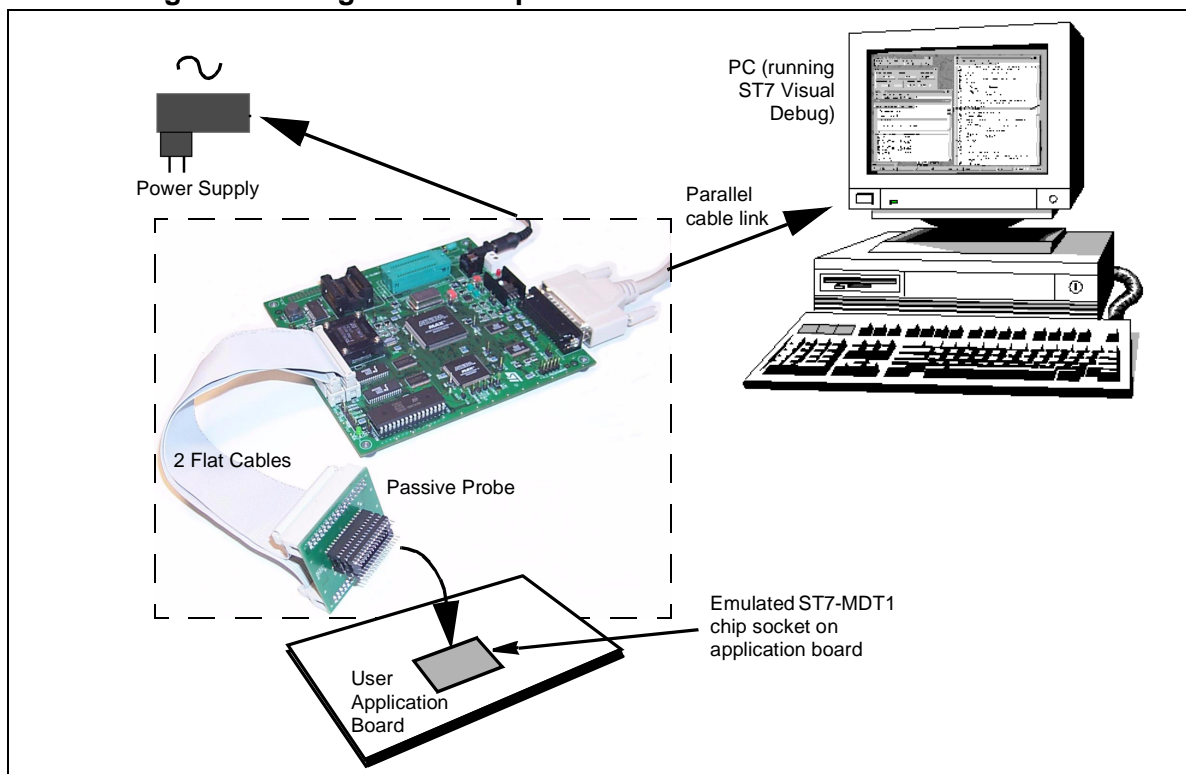
Figure 3 shows how you can set up the Development Board to perform in situ programming of devices on an application board.

Figure 2: Using the Development Board as an ST7 MCU Programming Board

PC (running Windows Epromer)

Power Supply

Parallel cable link

SDIP32 ZIF Socket for programming ST7MDT1-GP MCUs

SO28 ZIF Socket (available as an option) for programming ST7MDT1-GP MCUs
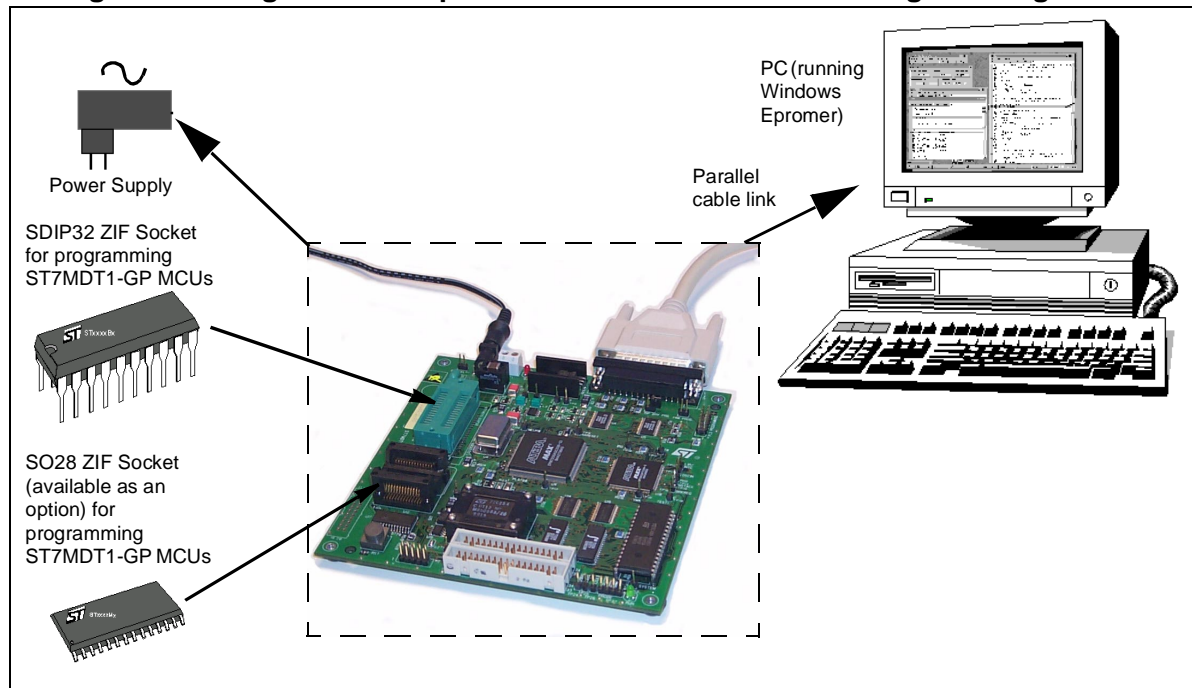
Figure 3: Using the Development Board for In Situ Programming (ISP)

PC (running Windows Epromer)

Power Supply

Parallel cable link

Connect ISP probe to ISP target connector

ISP Target Connector

User Application Board

Target Device to be programmed
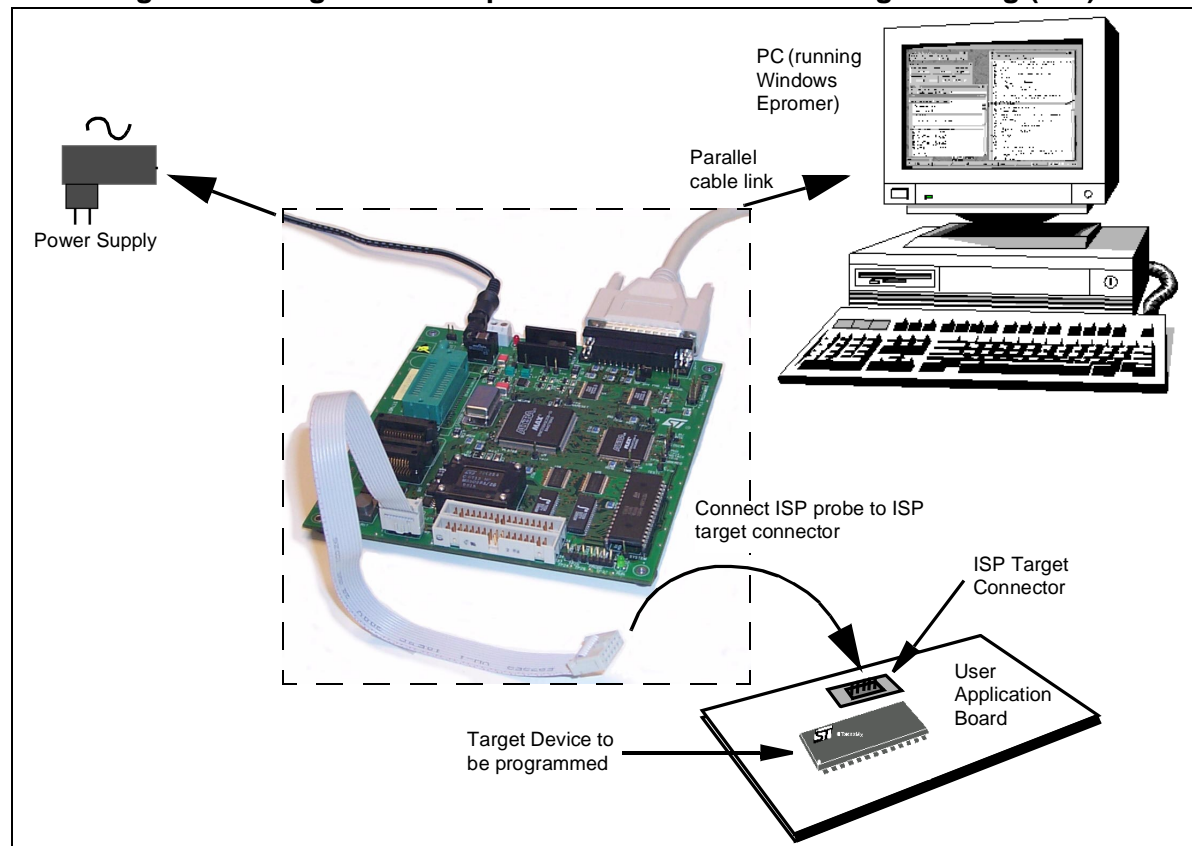
1.2   Software and Documentation for the Development Kit

The *MCU-on-CD* CD-ROM contains:

- ST7 Tools, comprising the following software:

  - The source-level graphic debugger, STVD7, that operates with the ST7 Development Kit, an ST7 Emulator or an ST7 simulator,

  - The ST7 Assembly chain, composed of an assembler, linker, librarian and formatter,

  - The ST7 Windows Epromer to program your MCU target devices.

- Third-party C compiler and toolchain demos (Hiware and Cosmic).

- ST7 application notes (with sources), training slides and exercises, this manual (in PDF version), and other useful reference materials.

- Datasheets for the ST7 MCU family.

1.3   Related Documentation

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

- ST7-Family Data Sheets

- ***ST7-Family 8-bit MCUs Product Overview***, Ref. BKST7/2

- ***Software Tools for the ST7 Family***, Ref. Doc-ST7ASMLK-SW

- ST7-Family Programming Manual

1.4   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 67. We'll be glad to help you!

2   GETTING STARTED

2.1   Your System Requirements

The ST7MDT1-DVP2 Development Kit (both hardware and software components) has been designed to work with PCs meeting the following requirements:

- One of the following operating systems: Microsoft® Windows® 95, 98 or NT®.

- Intel® Pentium (or compatible) processor with minimum speed of 100 MHz.

- Minimum RAM of 32 MB.

- 21 MB of free hard disk space to install all of the ST7 tools.

2.2   Delivery Checklist

The ST7MDT1-DVP2 Development Kit contains:

1 One development board (Ref.: MB 269).

2 One parallel cable for PC connection.

3 One 5 V external DC power supply with female connector cable.

4 SDIP32 passive probe (Ref.: DB 382) and two 34-pin flat cables.

5 An SO28 probe adapter (Ref.: DB359) for user application connection.

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

7 One "MCU on CD" CD-ROM (not shown).

8 This manual (not shown).

Figure 4: Development Board Layout

External Power Supply

Other Power Inlet)

Board Power ON (green LED)

Parallel Port Connector

SDIP32 ZIF Device Programming Socket

Sticker with serial and version numbers.

16 MHz Clock

SO28 ZIF Device Programming Socket (not provided with Kit)

ISP MCU Driver

If you wish to supply your application directly from the development board, via the application connector, place a solder spot here (G2 VCC APP). Max. 100 mA.

Push-button to simulate behavior of low voltage detection (LVD) cell of MCU during voltage drop.

10-pin In Situ Programming (ISP) connector

2 x 34-pin HE10 Application Connectors

If you want to connect the emulator clock to OSCOUT, place a solder spot here (G1).

External Trigger Output

External Trigger Input

External Clock Input

Ground

Trace Probe Inputs [2,1,0]

User Application Program Status LED (lights up red when running)

EPROM

Emulation RAM

Control RAM

ST72254 Emulation Chip
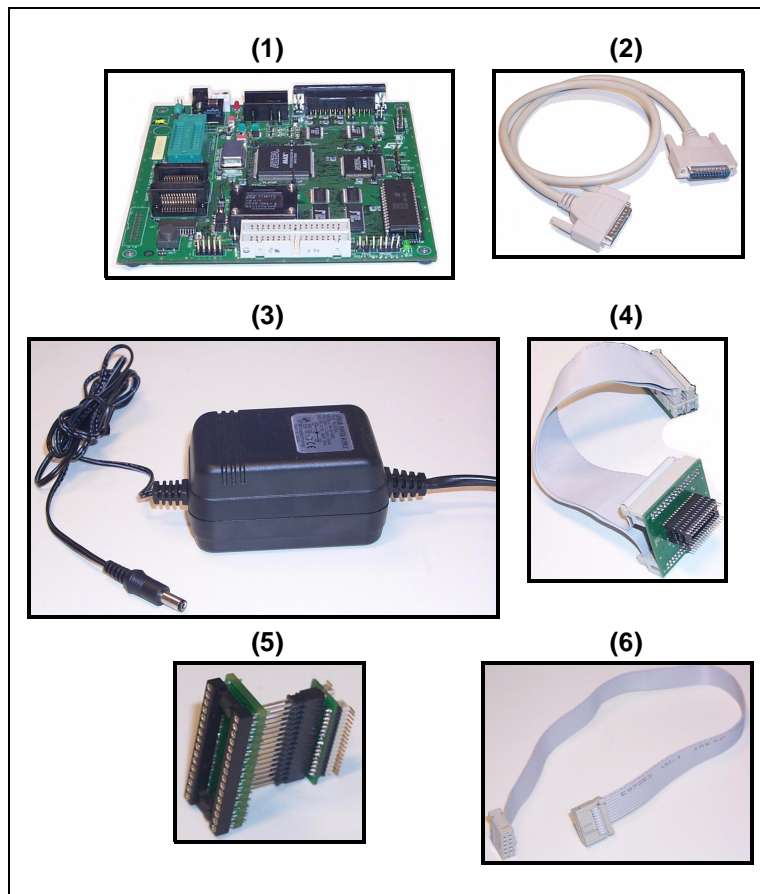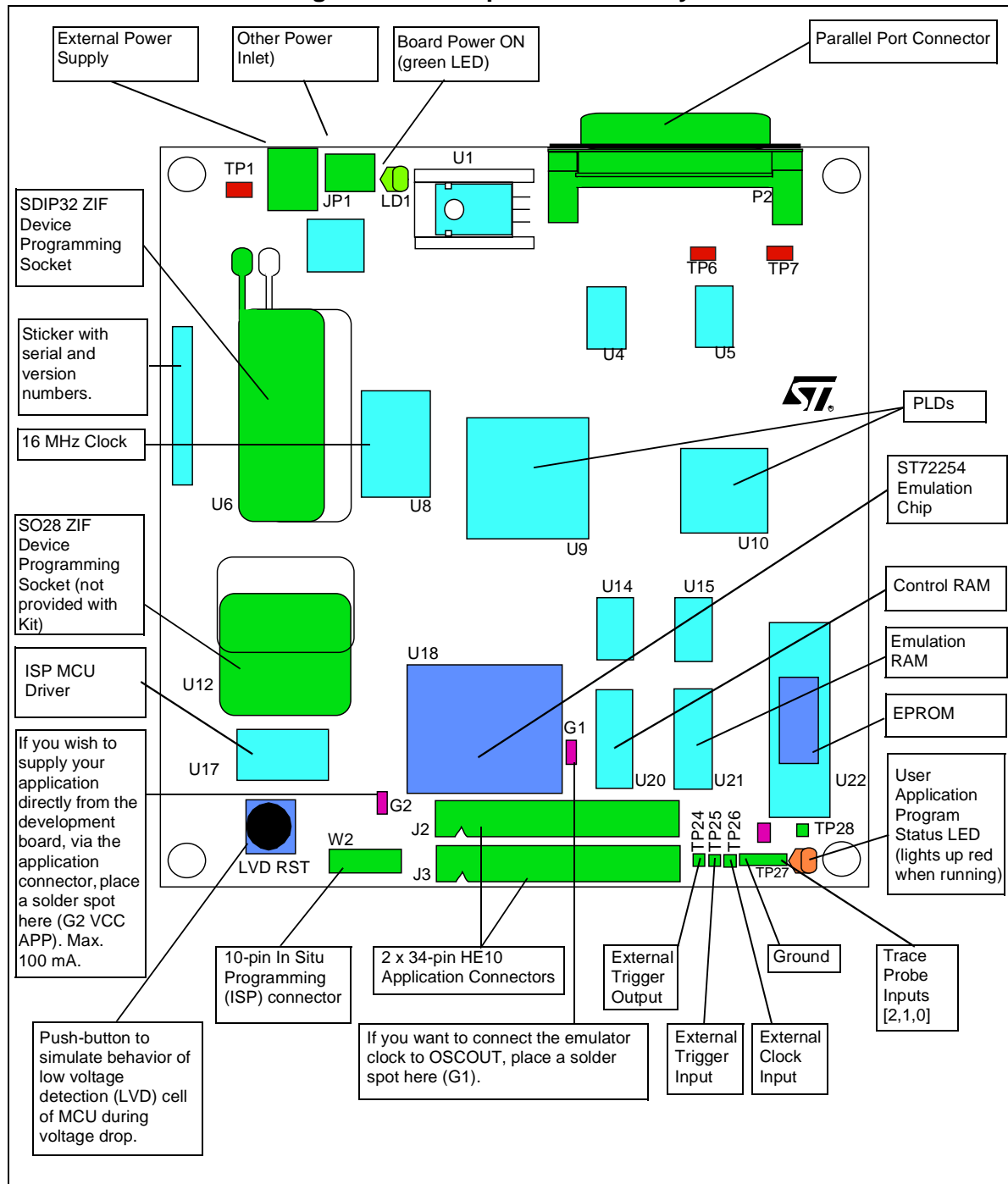
PLDs

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 passive probe to J2 and J3 on the development board using the two flat cables supplied with the development kit.

4 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.

5 Power on the PC and proceed with the installation of the software (refer to *Chapter 3* on page 13).

Caution: *Do not use the jumper connections TP1, TP6 and TP7—they are for factory testing only and modifications to them could cause your development board to malfunction.*

3 STVD7

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

The following sections tell you:

- *Section 3.1*—how to install the STVD7 software,
- *Section 3.2*—how to launch STVD7,
- *Section 3.3*—a little about STVD7's debugging features,
- *Section 3.4*—what a workspace is,
- *Section 3.5*—what toolchains and application files are supported by STVD7,
- *Section 3.6*—how to create a STVD7 workspace,
- *Section 3.7*—how to open existing workspaces,
- *Section 3.8*—how to open binary files,
- *Section 3.9*—how to change your project settings,
- *Section 3.10*—how to save workspaces,
- *Section 3.11*—how to switch from the build context to the debug context,
- *Section 3.12*—how to configure the target MCU in order to debug more accurately and efficiently.

3.1 Installing STVD7

Your 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 and Windows NT® operating systems.

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

1 Close all other open applications on your Windows desktop.

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

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

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

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

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

If you choose a customized installation, you can choose to install any or all of the STVD7 versions, and/or the Windows Epromer and/or the Assembler-Linker.

Note: *As a minimum, in order to use your DVP, you must install STVD7 for DVP.*

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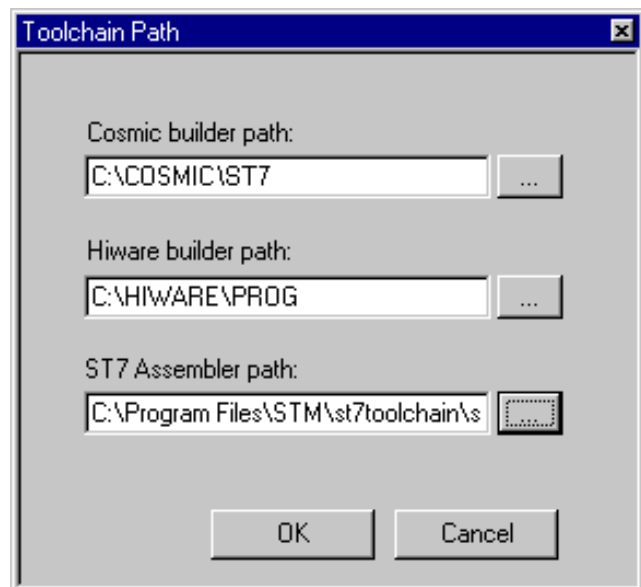
3 If you choose **Cancel,** you will be prompted again to enter the toolchain paths the next time you launch STVD7.

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 20. 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 18).

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 17).

- The **visual environment** consists of the open windows elements along with their current layout, bookmarks and other features.The visual environment is composed of two environments, one in the **Build context** and one in the **Debug context** (see *Section 3.11* on page 29).

- The **debugging information** includes information on breakpoints, memory mapping, advanced breakpoints programs, trace etc...

3.5 **Toolchains and application files**

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

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

• The **ST7 macroassembler toolchain** from STMicroelectronics, which generates either `.s19` or `.hex` application files with various intermediate files, such as `.map` or `.lst` files.

• The **Hiware C or Assembler toolchain**, which generates `.abs` application files with various intermediate files, such as `.o` or `.dbg` files.

• The **Cosmic C or Assembler toolchain** which generates `.elf` application files with various intermediate files, such as `.o` or `.st7` files.

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

• **The toolchain to be used—**Hiware, Cosmic or ST7 macroassembler.

• **The executable application file** (`*.abs`, `*.elf`, `*.s19` or `*.hex` depending on toolchain—refer to *Table 2* on page 19).

• **The maker program** for the toolchain. The maker program can be a part of the toolchain software (such as Hiware's `maker.exe`) or you can choose to use a generic maker such as `Nmake.exe` or `Gmake.exe` (which is provided with the STVD7).

• **The maker batch file** (`*.mak` or `*.bat`). This is a file which you create for each application which spawns the compilation and/or link step each time you wish to build or rebuild. In it, you define the conditions for recompiling, re-linking or both.
Default `*.mak` or `*.bat` files are often included with the toolchains—for example, maker.mak is included with the Hiware toolchain and simply recompiles your application if it detects that the file has been saved since the start of your debugging session. The STVD7 software includes sample `*.mak` and/or `*.bat` files for each toolchain—these are listed in *Table 1*.

Table 1: Sample files included with STVD7

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

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

3.5.1   About application files

The user should verify that the options to include debug information were active during creation of the project files. *Table 2* on page 19 summarizes the way each toolchain functions and lists the different file types (source files, intermediate files and application files) used and produced by the toolchain. The **application file types** and **intermediate file types** necessary to exploit fully the STVD7 capabilities are listed.

Table 2: Toolchain steps and their output files

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

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

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

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

Note: *It is always preferable to have access to all of the files generated by the development toolchain. However, you can load* `*.s19` *or* `*.hex` *binary files directly and have limited debugging capabilities (refer to Section 3.8 on page 24).*

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.
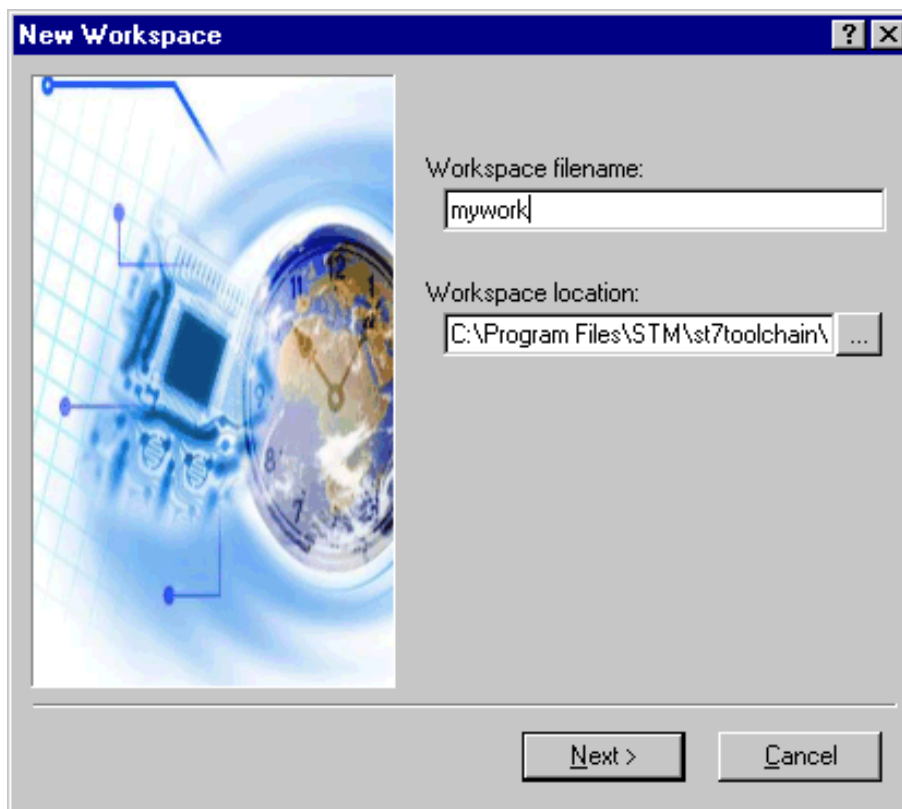
2 Then, click ***Next>***. The **New Workspace: Project Settings** dialog box appears:
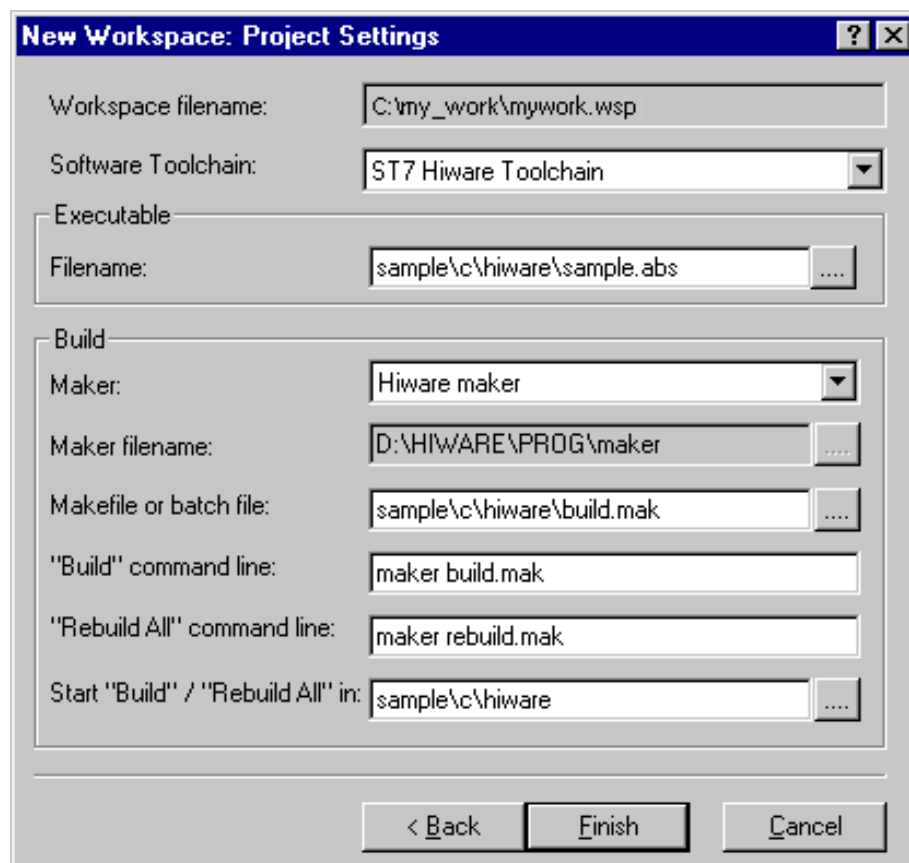
Here you enter your software toolchain, your executable filename and your build parameters either by typing or using the drop boxes.

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 19)—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 18). Here we have chosen

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

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

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

When you create a new workspace, the first time you switch to Debug context (see *Section 3.11* for an explanation of STVD7 contexts), the MCU Configuration window will automatically open to prompt you to choose you target MCU and confirm or modify its option and memory configuration (see *Section 3.12* on page 30).

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 18.

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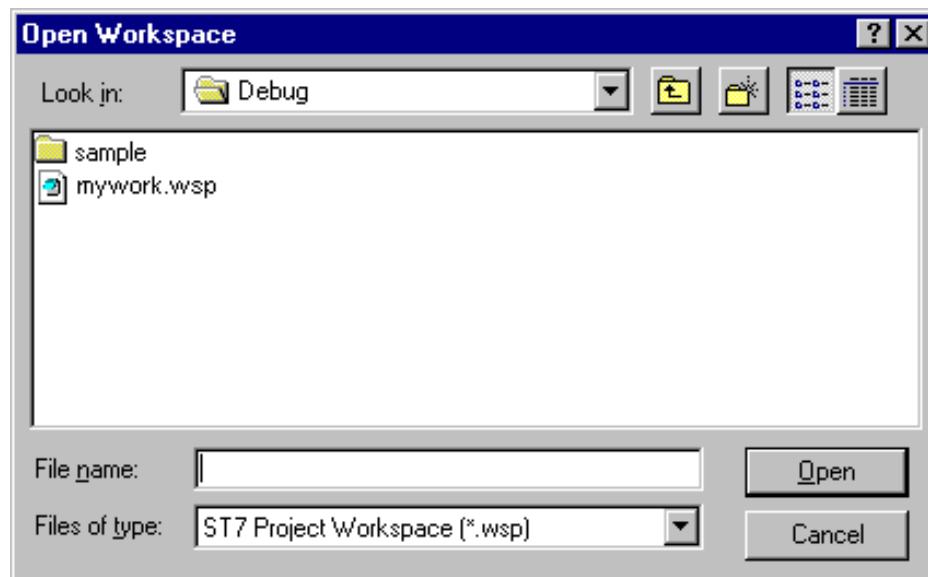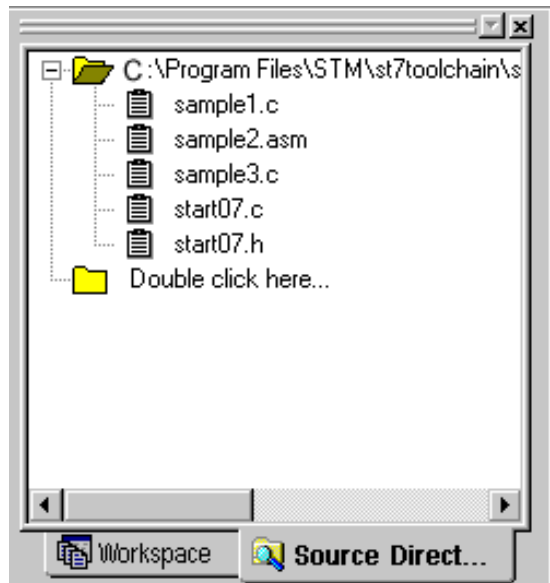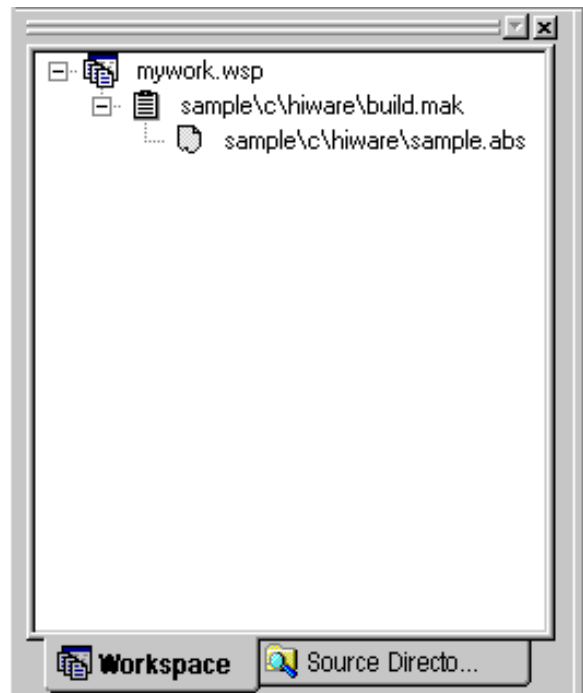
2   The Workspace window opens.

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


Note: *Although the name of the application file is shown in the Workspace window, it has not yet been loaded into the emulation memory—see page 24.*

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

3   If there are no source files shown in the Source Directory tab of the Workspace window, or you wish to list additional files stored in another folder, you may browse to them by clicking the ***Double Click here...*** folder. The **Add Source**

Directory window pops up allowing you to enter or browse for a new directory, and filter out the file types of interest.
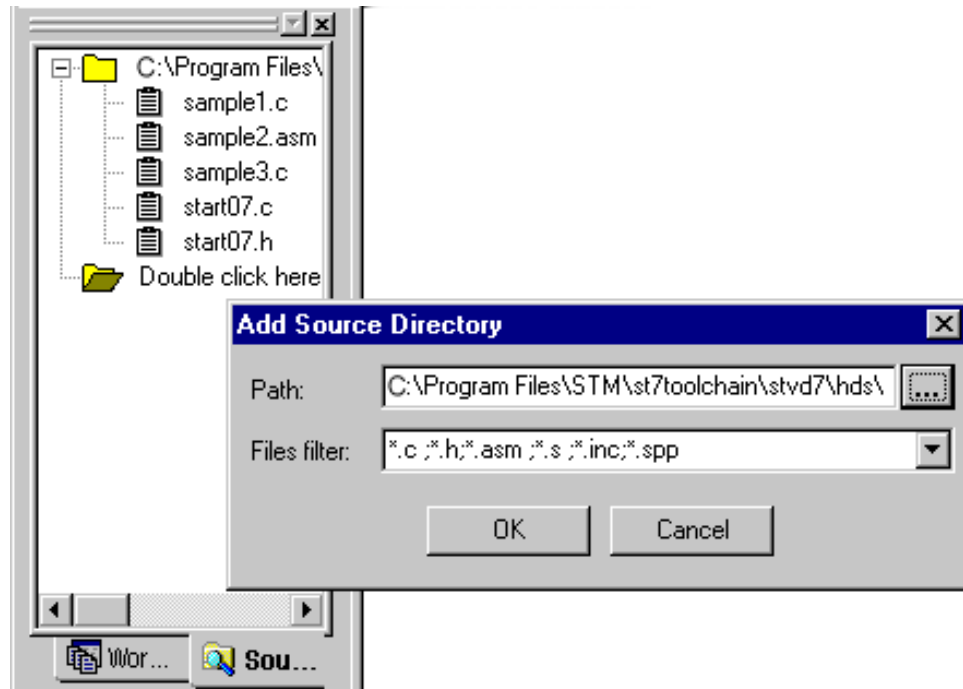
4 To load the application file, as well as any intermediate files, click the Debug icon or the Reset Chip icon. The application and symbols will be loaded. Before you can start debugging, you must set the target hardware device by configuring the MCU.

3.8 Opening binary files

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

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

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

2 Browse to the folder where your binary file is stored, and select **All files (*.*)** in the *Files of type* field.
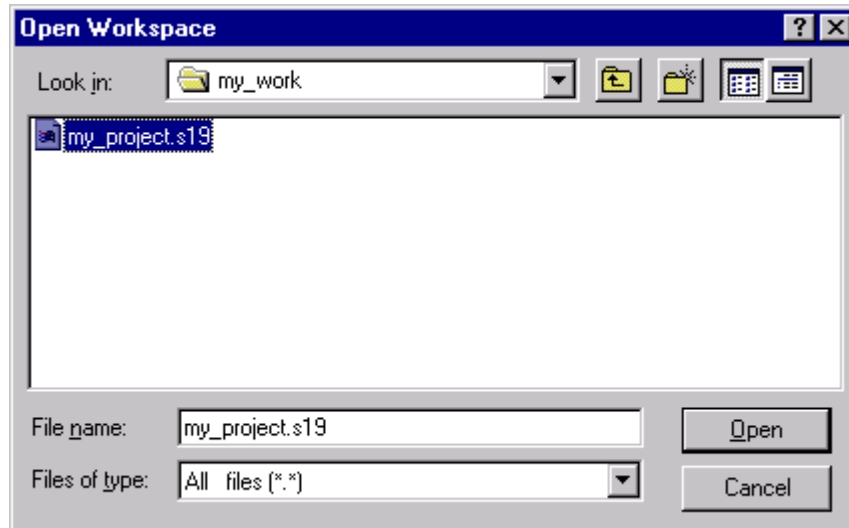

**3** Select your binary file (`*.hex` or `*.s19`) and click **Open**.

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

3.9 Changing your project settings

The Project menu contains the **Build** and **Rebuild All** commands you need to recompile your application after having made changes to it in the course of debugging. You may also access your project or toolchain settings in the event you wish to change them.
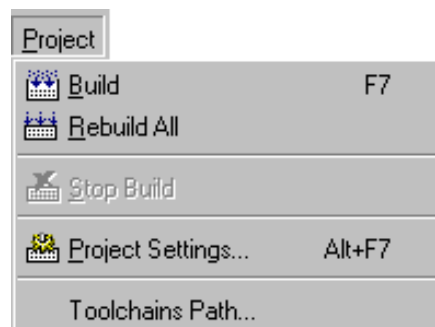
From the main menu, select **Project>Project Settings**.
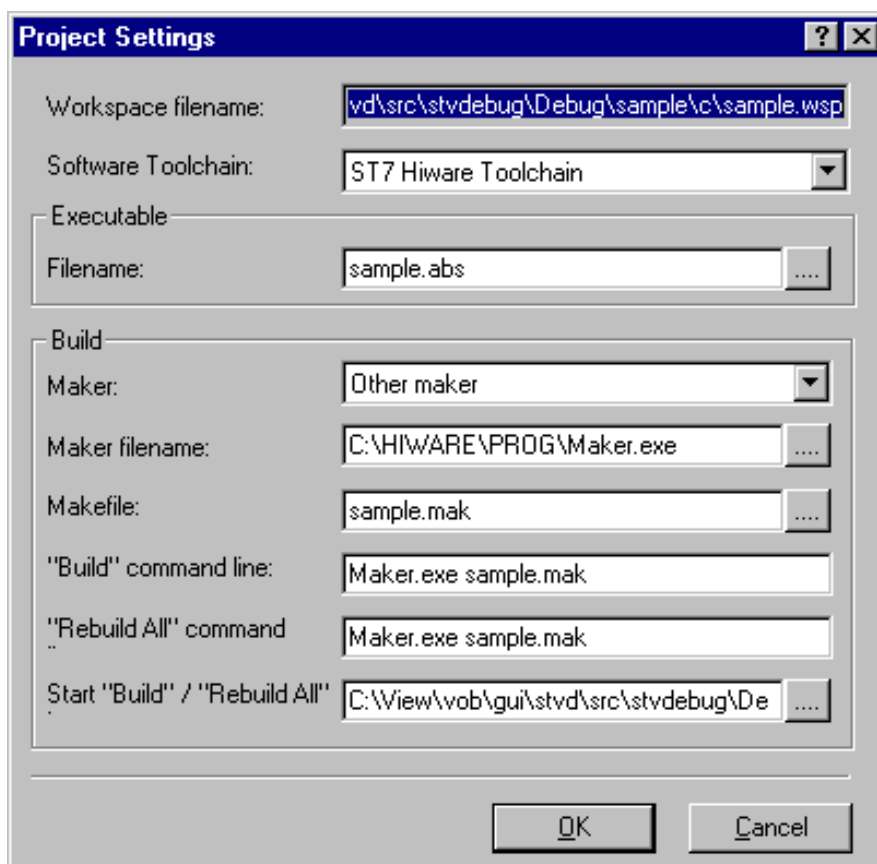
You can change your settings here and continue running your application. When you exit STVD7, the system will ask you if you want to save these settings in the workspace you have been working in. If you choose **yes**, these will become your new workspace settings; if you choose **no**, these settings will be lost.

The ***Toolchain Path...*** item invokes the following window:
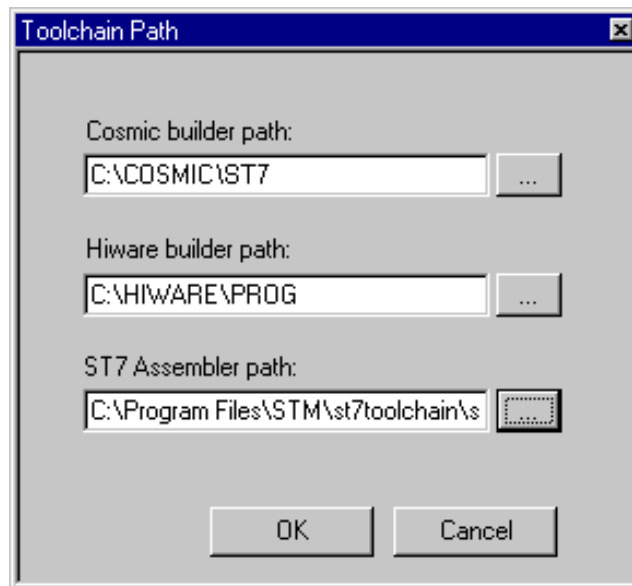
In this window, you can define your builder and/or Assembler paths. Clicking [....] opens a browser window.

3.10   Saving workspaces

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

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

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

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

2   In the Options window that opens (see *Figure 5 on page 28*), 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.
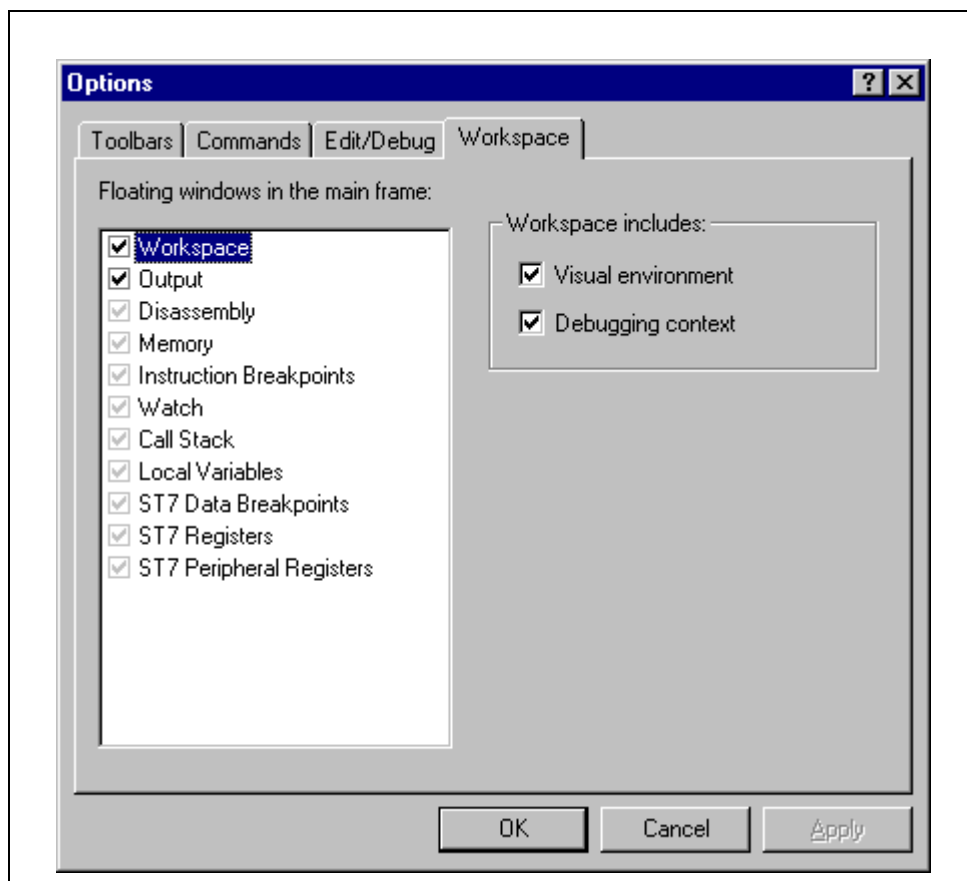

Figure 5: Options window

1 Click *Apply* to confirm your settings.

2 Click *OK* close the dialog box.

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

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

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

3.11 Debug context and Build context

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

Briefly, the two contexts are different in that:

- In the build context, you can open and close workspaces and build or re-build the application executable file.

- In the debug context you set the emulated MCU configuration (this step is described in *Section 3.12* on page 30) and debug the executable file created while in the build context.

3.11.1 Build Context

The build context is the context set when starting STVD7. In this context, it is not necessary to be connected to 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.

3.11.2 Debug Context

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

- Loading, running and stopping the application.

- Defining the MCU configuration (MCU options and memory mapping).

- Viewing source and disassembled code.

- Setting instruction breakpoints with a counter and/or condition.

- Setting data breakpoints.

- Viewing local variables, memory and ST7 registers.

- Viewing history of execution from the trace buffer or with the Call Stack feature analyzing the performance of a piece of code.

3.11.3   Switching between contexts

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

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

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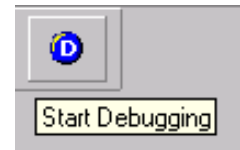
3.12   Configuring the MCU

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

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

1   First, ensure that you are in **Debug context** by clicking on 🔵 . (STVD7 has two contexts: Debug context and Build context—these are described in *Section 3.11*.)

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

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

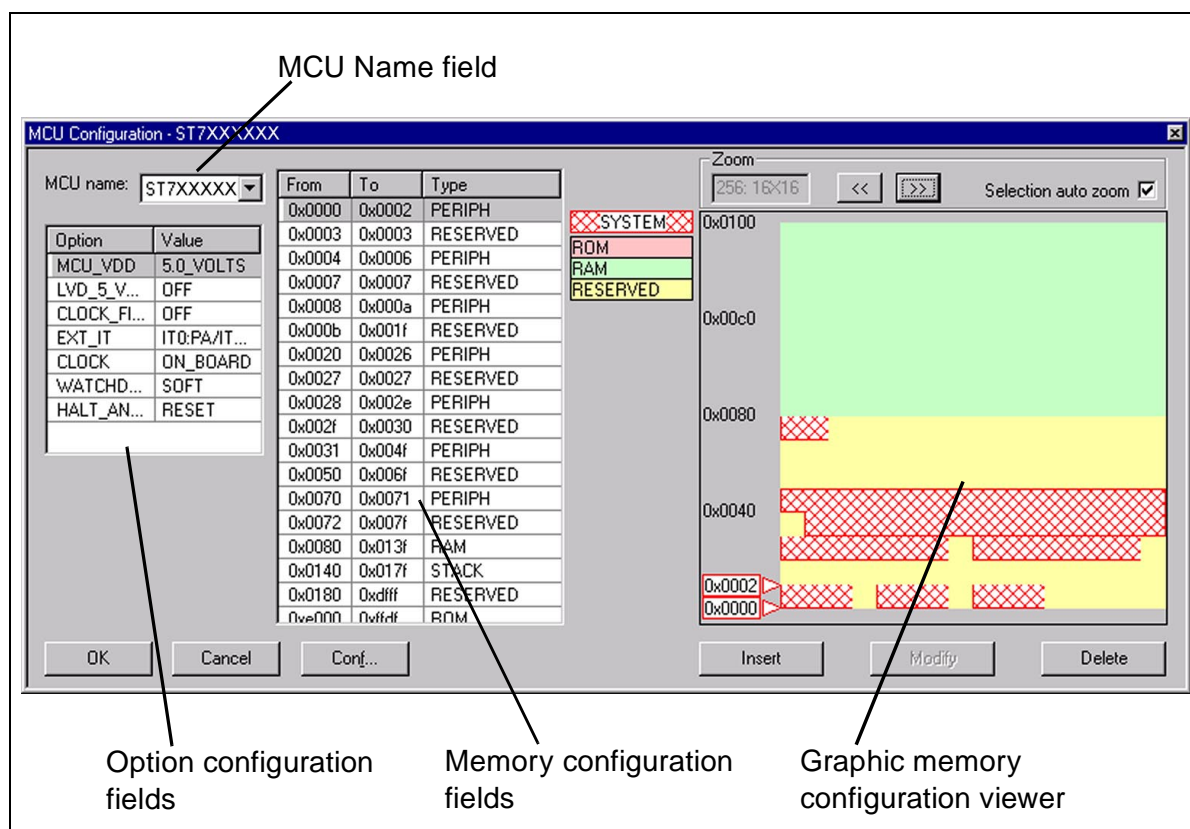An example of a typical MCU Configuration window is shown in *Figure 6*.

MCU Name field

Option configuration fields

Memory configuration fields

Graphic memory configuration viewer

Figure 6: 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.

Memory
configuration fields

Graphic memory
configuration viewer
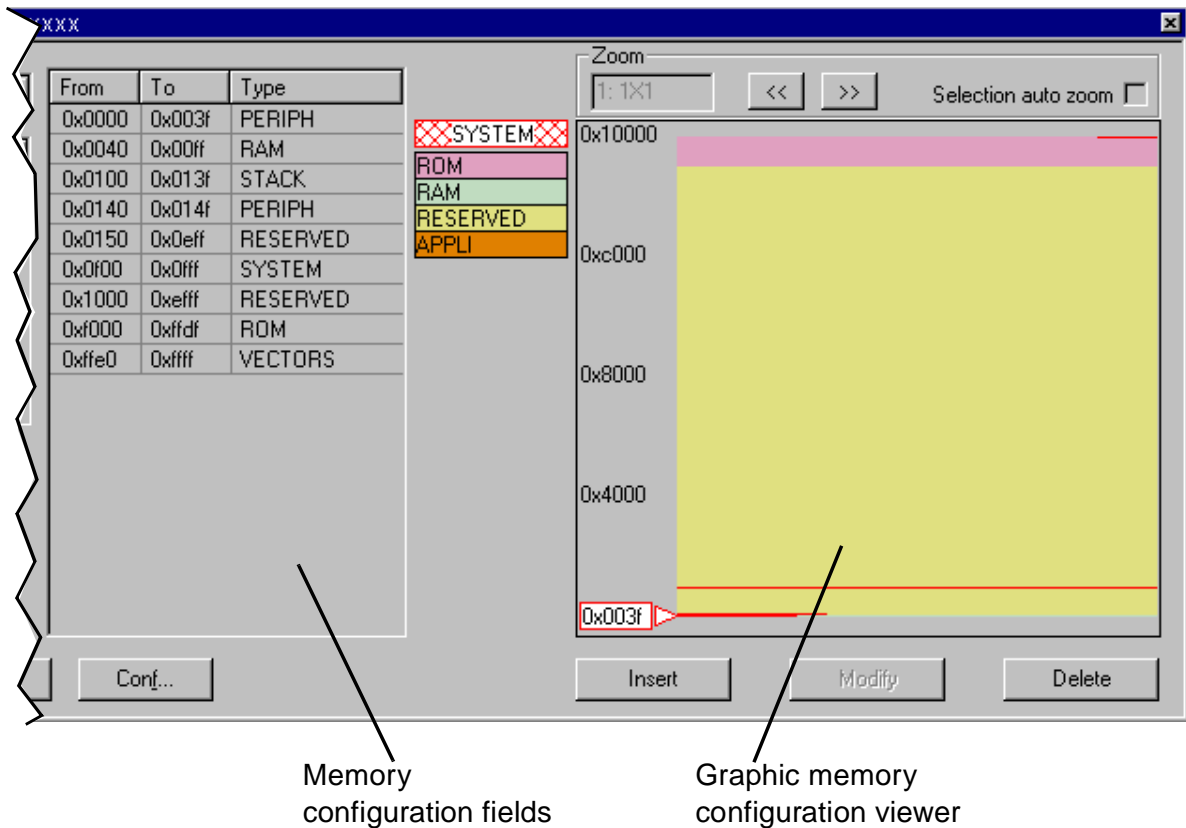
There are two methods for configuring the memory settings on the MCU: by
typing in the start and stop addresses of each memory zone into the **memory
configuration window**, and by graphically moving the memory zone
boundaries in the **graphic memory configuration viewer** (see *page 34* for
more instruction).

Memory zone types

The left column of the **memory configuration window** indicates the address
range of each memory zone. The right column indicates the memory type of each
zone. Depending on your target MCU, the available memory types may be:
Peripherals, RAM, ROM, Stack, System, EEPROM, Reserved, Vectors,

Application. Some of these zones can have their type and size modified, others cannot be modified. Their definitions and properties are explained as follows:

- **Peripherals**: Microcontroller internal or rebuilt peripherals registers. Their properties are defined as in the microcontroller user manual. This memory cannot be modified.

- **RAM**: Random-Access-Memory of the microcontroller. This memory type can be modified.

- **ROM**: Read-Only Memory of the microcontroller. Write protected. This memory type can be modified.

- **Stack**: Stack of the microcontroller. This memory type cannot be modified.

- **System**: The 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 user manual. This memory type cannot be modified.

- **Reserved**: This memory zone is reserved as on the microcontroller. It is not allocated to any use and is write protected. This memory type cannot be modified.

- **Vectors**: This memory zone contains the user interrupt vectors zone. It is write protected. This memory type can be modified.

- **Application**: This memory type is microcontroller-specific. The user can add memory or peripheral resources on its hardware. It is not available on every 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.
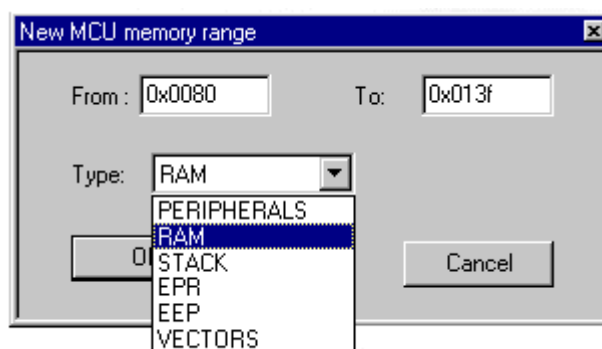
To create a new zone of any type:

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.

3.13   **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 ST7MDT1-DVP2 Development Kit, you may program the MCUs shown in *Table 3*.

Table 3: Programmable Devices

Supported Device	Device Group	Programmable Memory Type[1]	Programming Method(s)[2]
ST 72254 G1 ST 72254 G2	Group 1 (based on ST72254)	EEPROM	ZIF Sockets[3] and ISP
ST 72216 G1	Group 1 (based on ST72254)	EEPROM	ZIF Sockets and ISP
ST 72215 G2	Group 1 (based on ST72254)	EEPROM	ZIF Sockets and ISP
ST 72104 G1 ST 72104 G2	Group 1 (based on ST72254)	EEPROM	ZIF Sockets and ISP
ST 72251 G1 ST 72251 G2	Group 2 (based on ST72251)	EPROM and OTPROM	ZIF Sockets
ST 72213 G1	Group 2 (based on ST72251)	EPROM and OTPROM	ZIF Sockets
ST 72212 G2	Group 2 (based on ST72251)	EPROM and OTPROM	ZIF Sockets
ST 72101 G1 ST 72101 G2	Group 2 (based on ST72251)	EPROM 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 *Programming Methods* on page 38.

3) All SDIP32 packages are programmable using standard Development Kit. SO28 packages can be programmed by adding the SO28 ZIF socket to the footprint provided at U12. The ENPLAS product reference for the socket (not supplied with the kit) is **ENPLAS OTS28-1.27-04**.

4.1   Device Programmer Features

Figure 7: General Eproming Architecture
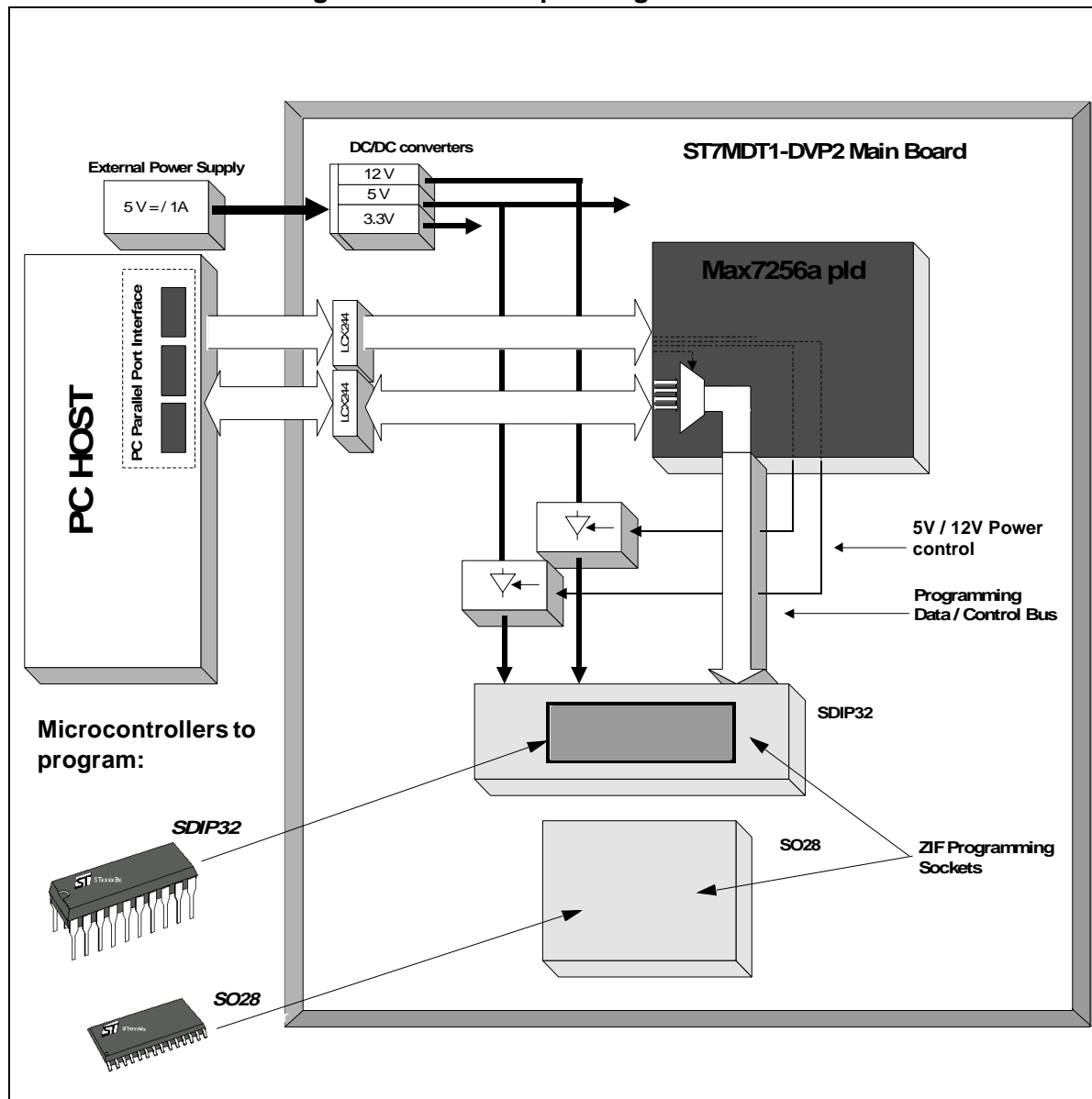
4.2   Programming Methods

4.2.1   ZIF Sockets

The ST7MDT1-DVP2 development board is provided with a SDIP32 **Zero Insertion Force (ZIF)** socket which allows the programming of SDIP32 packaged MCUs.

A footprint is also provided on the development board, for the optional mounting of an SO28 ZIF socket. The reference number for the SO28 ZIF socket (not supplied with the kit) is **ENPLAS OTS28-1.27-04**.

4.2.2   In Situ Programming

In addition to classic MCU programming using ZIF sockets, the ST7MDT1-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 37) 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.

The Windows Epromer fully supports the ISP functionality.

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.*
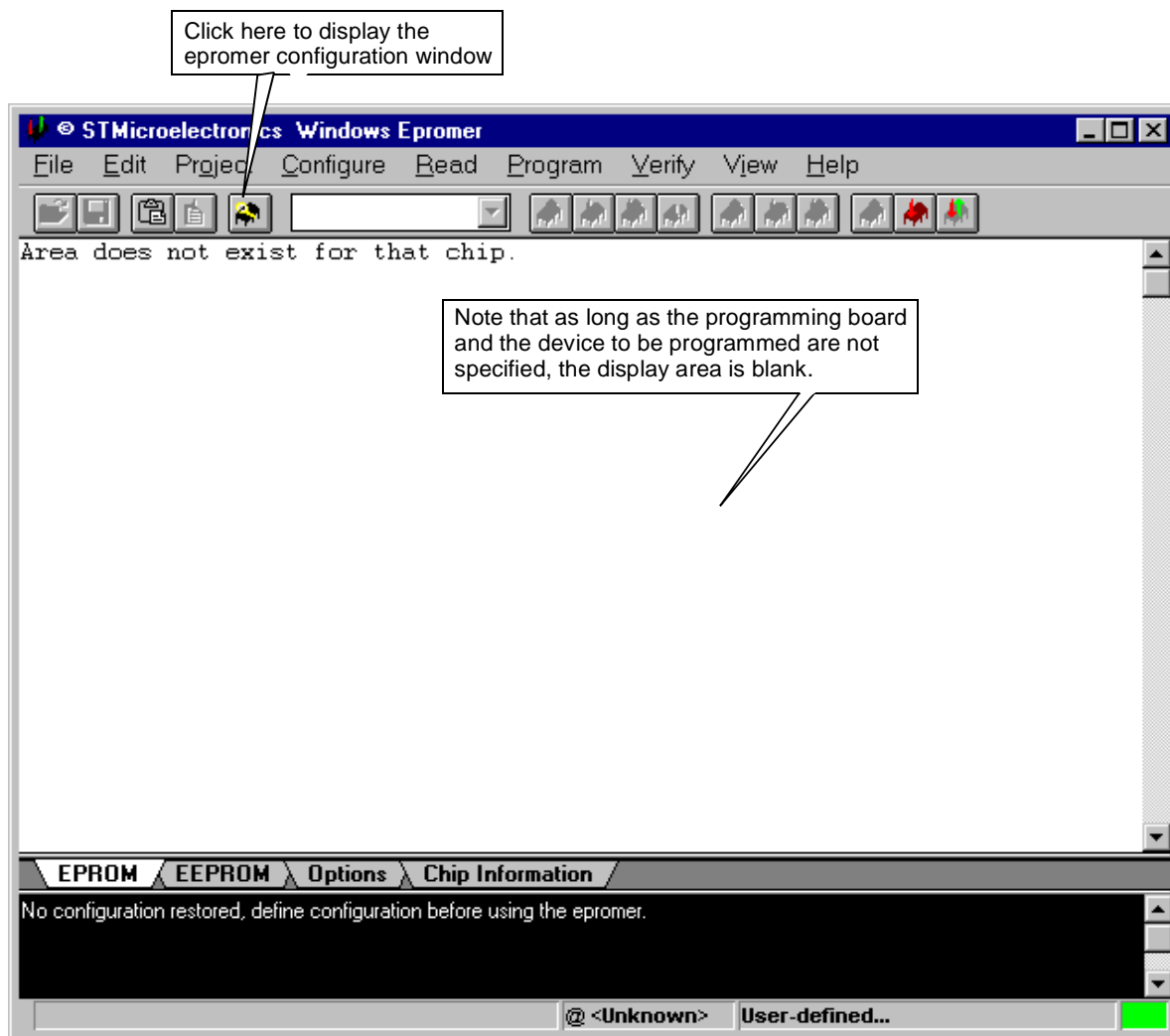
Place the device into the zero insertion force (ZIF) socket mounted on the board (location U6 for SDIP32 devices) with the erasure window on top and pin 1 matching the triangle with a "1" beside it on the board.

Align dot on device (pin 1) with the triangle on board.

Target Device

ZIF Socket

Erasure window
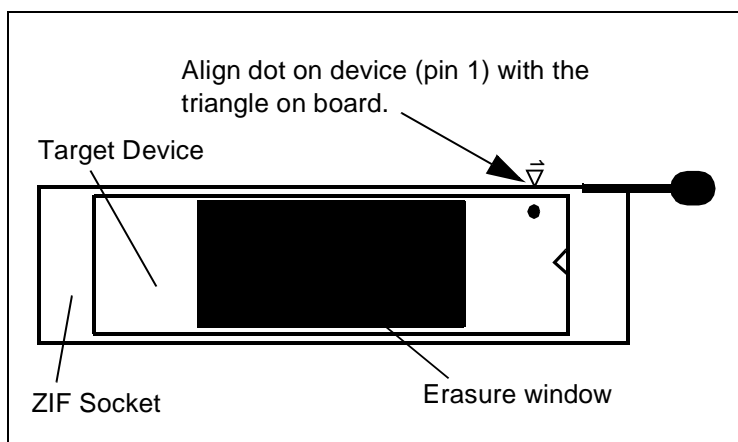
4.4   **Starting the Windows Epromer**

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

The Epromer main window appears:

Click here to display the epromer configuration window

Note that as long as the programming board and the device to be programmed are not specified, the display area is blank.
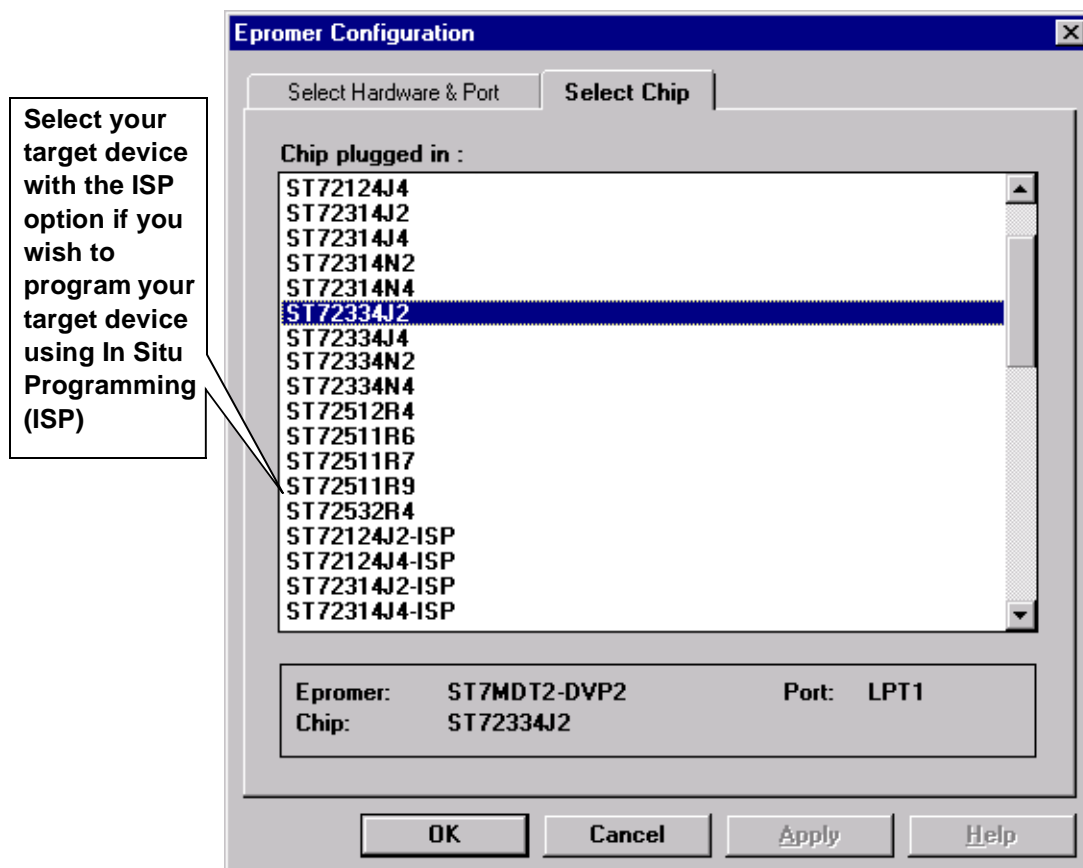
4.5   Configuring the Epromer

Follow these steps:

1 In the main window tool bar, click the [icon] icon to open the epromer configuration dialog box:
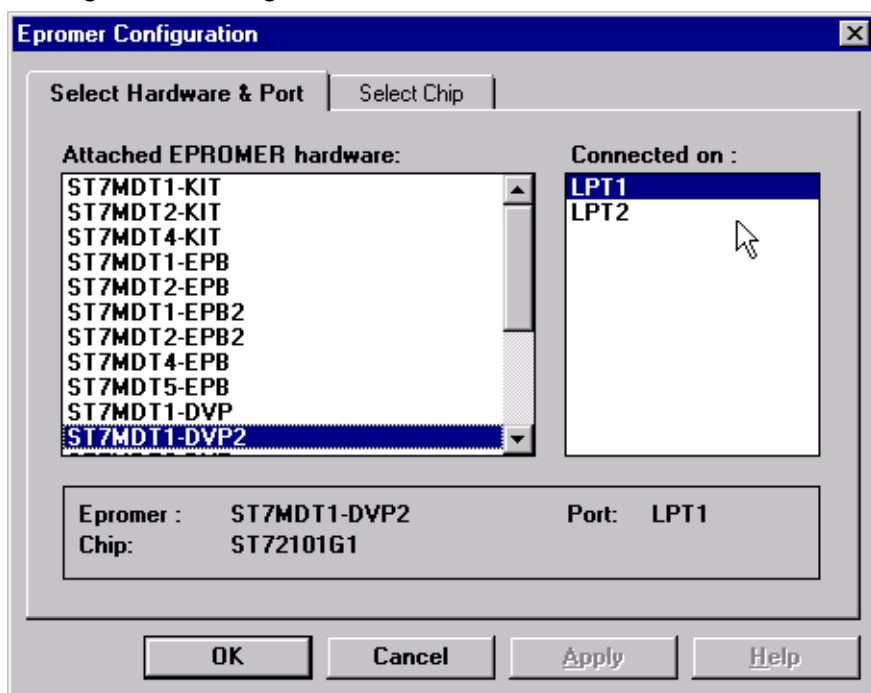
2 From the list, select the programming board for this ST7MDT1-DVP2 Development Kit:
ST7MDT1-DVP2.

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 ST7MDT1-DVP2 Development Kit.
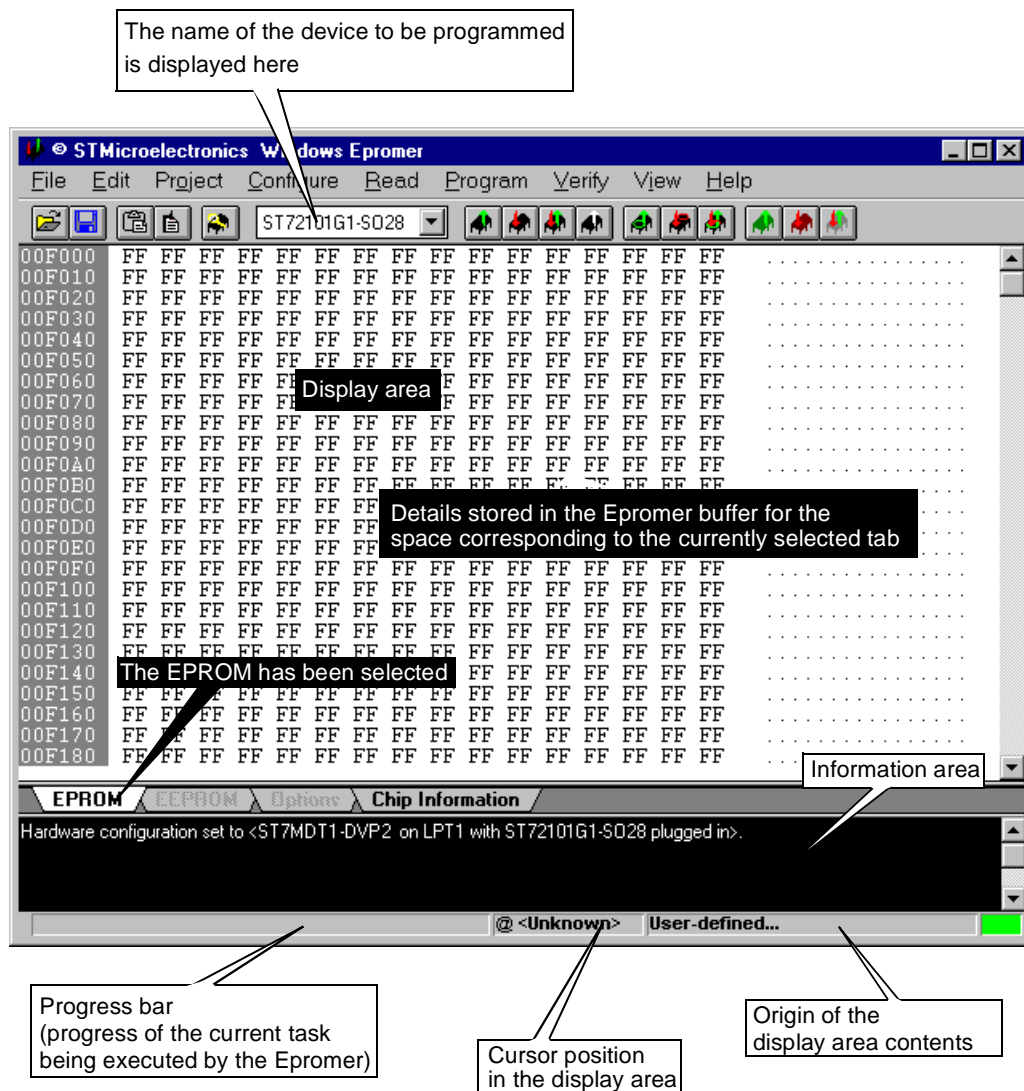
The list box shown below appears.

Select your target device with the ISP option if you wish to program your target device using In Situ Programming (ISP)

```
Epromer Configuration                                    ☒

   Select Hardware & Port     Select Chip

   Chip plugged in :
   ST72124J4                                          ▲
   ST72314J2
   ST72314J4
   ST72314N2
   ST72314N4
   ST72334J2
   ST72334J4
   ST72334N2
   ST72334N4
   ST72512R4
   ST72511R6
   ST72511R7
   ST72511R9
   ST72532R4
   ST72124J2-ISP
   ST72124J4-ISP
   ST72314J2-ISP
   ST72314J4-ISP                                      ▼

   Epromer:    ST7MDT2-DVP2          Port:   LPT1
   Chip:       ST72334J2

        OK          Cancel         Apply         Help
```

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 ISP functionality.*

6 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.

The name of the device to be programmed is displayed here

Display area

Details stored in the Epromer buffer for the space corresponding to the currently selected tab

The EPROM has been selected

Information area

Progress bar
(progress of the current task being executed by the Epromer)

Cursor position
in the display area

Origin of the
display area contents

7   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 Programming Methods on page 38 for details on which programming methods can be used with your target device.*

5   HARDWARE FEATURES

5.1   Link to PC

The ST7MDT1-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.*

5.2   Power Supply

A plug-in power supply pack is supplied with the ST7MDT1-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
ST7MDT1-DVP2/EU	220 V
ST7MDT1-DVP2/UK	240 V
ST7MDT1-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.

5.2.1   Supplying the Application Board

You may supply your application board directly from the development board via the VDD pin in the P3 probe connector, after a solder spot has been placed in G2 on the development board. This method is not recommended.

5.3   Connections to External User Resources

5.3.1   Pin Descriptions

You may connect an application board to the ST7MDT1-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.

Table 4 shows the pin assignment for the passive probe connector on the application board.

Table 4: SDIP32 Passive Probe Pin Assignment

Probe Pin No.	Pin Name/Description	Probe Pin No.	Pin Name/Description
1	RESET	17	C2 Port
2	OSCIN (OSC1)	18	C1 Port
3	OSCOUT[1] (OSC2)	19	C0 Port
4	B7 Port	20	A7 Port
5	B6 Port	21	A6 Port
6	B5 Port	22	A5 Port
7	B4 Port	23	A4 Port
8	Not used	24	Not used
9	Not used	25	Not used
10	B3 Port	26	A3 Port
11	B2 Port	27	A2 Port
12	B1 Port	28	A1 Port
13	B0 Port	29	A0 Port
14	C5 Port	30	Not used (VPPTEST)
15	C4 Port	31	VSS (Ground)
16	C3 Port	32	VDD[2]

1) The emulator clock is connected to OSCOUT only when a solder spot is placed in G1 on the development board.

2) The emulator is only connected to VDD if a solder spot is placed in G2 on the development board. The application voltage can be either 3.3 to 5 V only.

Table 5: SO28 Passive Probe Pin Assignment

Probe Pin No.	Pin Name/Description	Probe Pin No.	Pin Name/Description
1	RESET	15	C2 Port
2	OSCIN (OSC1)	16	C1 Port
3	OSCOUT[1] (OSC2)	17	C0 Port
4	B7 Port	18	A7 Port
5	B6 Port	19	A6 Port
6	B5 Port	20	A5 Port
7	B4 Port	21	A4 Port
8	B3 Port	22	A3 Port
9	B2 Port	23	A2 Port
10	B1 Port	24	A1 Port
11	B0 Port	25	A0 Port
12	C5 Port	26	Not used (VPPTEST)
13	C4 Port	27	VSS (Ground)
14	C3 Port	28	VDD[2]

1) The emulator clock is connected to OSCOUT only when a solder spot is placed in G1 on the development board.

2) The emulator is only connected to VDD if a solder spot is placed in G2 on the development board. The application voltage can be either 3.3 to 5 V only.

5.4   On-Chip Peripherals

You can configure certain on-chip peripherals in STVD7's **MCU Configuration** dialog box (refer to *Section 3.6* on page 20) so that the emulator accurately emulates your target device.

The following options are available on all supported target devices:

Clock

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

Note: *The clock types available can vary depending on the target device. Refer to Section 5.6.1 on page 53 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, max. 16 MHz) whose signal is supplied via the mini wrapping pin "EXTCLK" located on the board next to the passive probe flat connector (ref.: TP26). For Group 1 MCU's you may also choose an on-chip clock.

Note: *In **normal mode** (Miscellaneous Register slow mode bit = 0), acceptable CLOCKIN frequencies can take any value between 16 MHz and 250 kHz.*

In **slow mode** (Miscellaneous Register slow mode bit = 1), the EXTCLK frequency must take values between 16 Mhz and 8 Mhz.

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

However, the application board can use the development board clock via the OSCOUT pin of the probe. In this case you must place a solder spot on G1 (see *Table 4* on page 46).

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.

The following options are only available on the following microcontrollers: **ST72254G1/G2, ST72216G1, ST72215G2, ST7104G1/G2**.

MCU VDD

Allows you to set the supply voltage for the debugging session at either 5 V or 3.3 V.

LVD 5 Volts

Allows you to enable (**ON**) or disable (**OFF**) the LVD (Low Voltage Detection). This option only works when MCU VDD is set to 5 V. In this case, the selected low voltage detection threshold will be 4.3 V.

Clock Filter

When ON, the internal cell clock filter of the emulation chip is enabled. Refer to the datasheet for your MCU for more information on the internal clock filter.

EXT IT

Allows you to specify to which port(s) external interrupts will be attributed.

IT0:PA/IT1:PB-PC means that external interruptions at level 0 (IT0) will attributed to port A, while external interruptions at level 1 (IT1) will be attributed to ports B and C.

IT0:PA-PC/IT1:PB means that the external interruptions at level 0 (IT0) will be attributed to ports A and C, while external interruptions at level 1 (IT1) will be attributed to port B.

Halt and Watchdog

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

5.5   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 used hardware events to control the external output trigger (TRIGOUT) signal or the trace buffer filtering, refer to *Section 5.5.1* on page 50.

5.5.1   Trigger/Trace Settings

The Trigger/Trace Settings dialog box (shown below), allows you choose a variety of options:
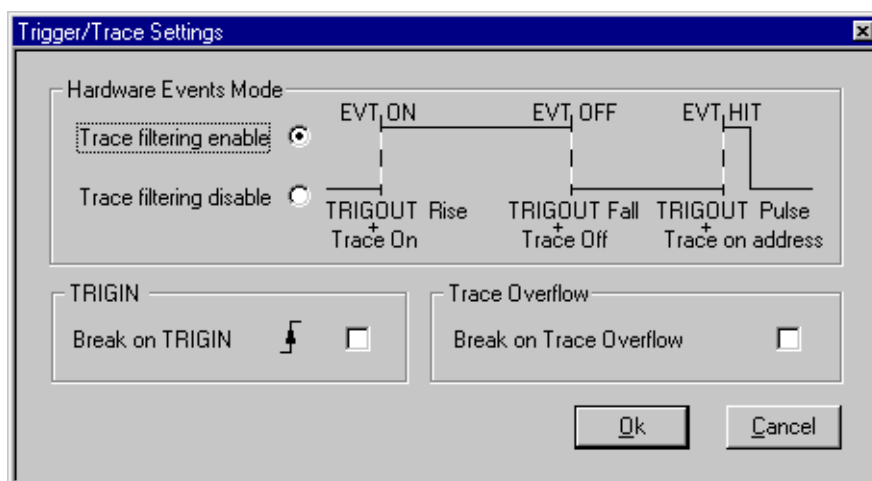
- Select between two hardware event modes: trace filtering enabled or trace filtering disabled. In the first mode, the recording of the trace buffer is controlled using the same hardware events as those used to control the signals from the external output trigger. See below for details on the external 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.

5.5.2   External Output Trigger (TRIGOUT)

The ST7MDT1-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 board next to the passive probe flat connector (Ref.:TP24).

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**).

5.5.3   Input Trigger (TRIGIN)

The ST7MDT1-DVP2 Development Board provides a special inlet (**TRIGIN**, Ref.:TP25) 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:
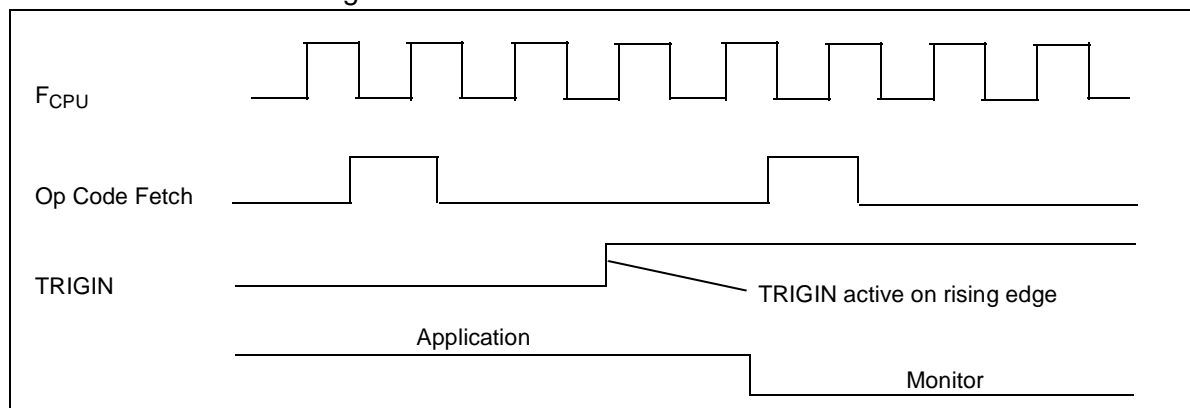
5.6   MDT1-DVP2 Emulation Features

Figure 8: Emulation Architecture
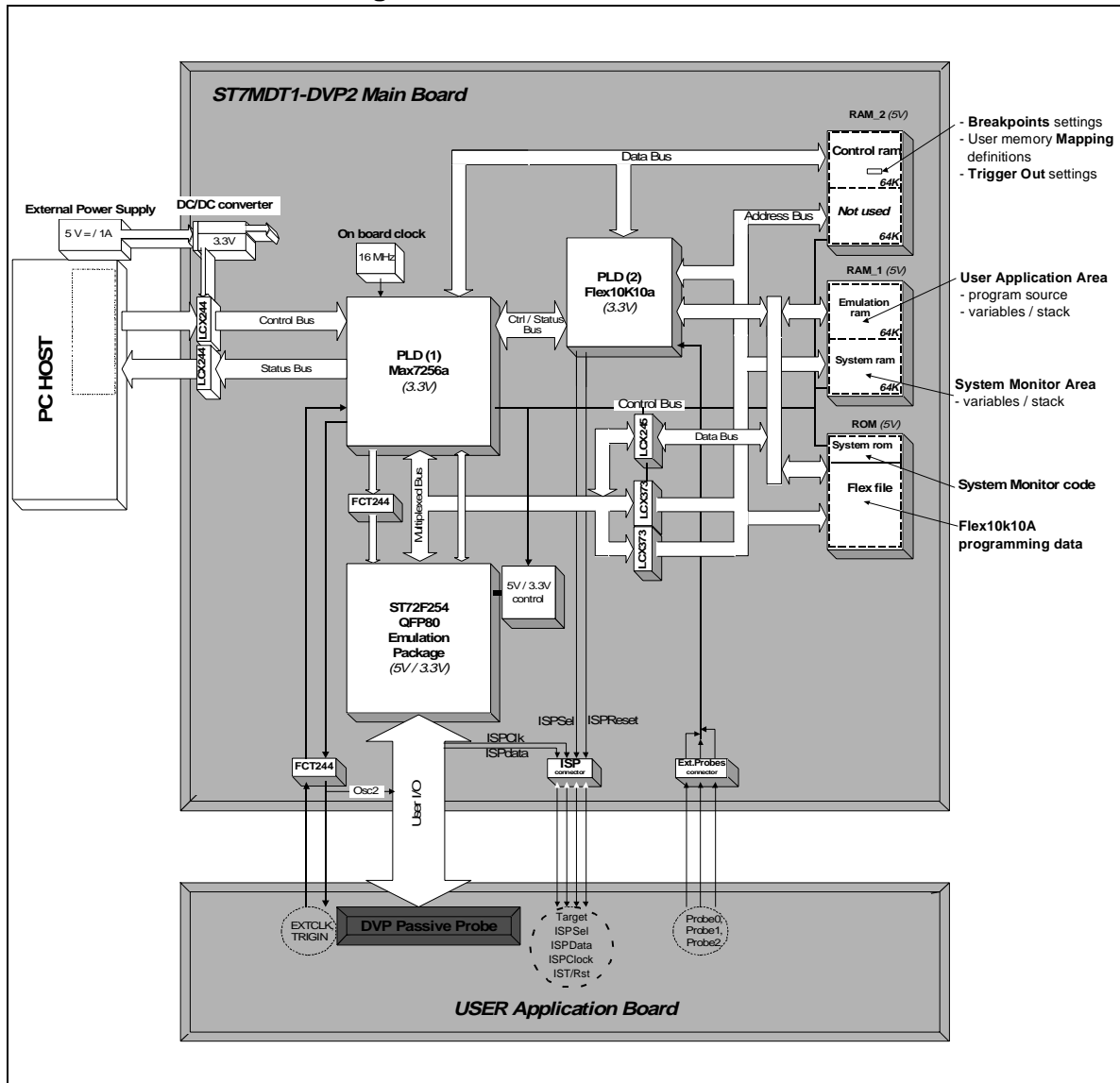
5.6.1   Emulation Functional Limitations and Discrepancies

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 ST72C254)	Group 2 (based on ST72251)
ST72254 G1/G2 ST72216 G1 ST72215 G2 ST72104 G1/G2	ST72251 G1/G2 ST72212 G2 ST72213 G1 ST72101 G1/G2

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 48* for instructions on how to choose clock options). **For target devices in Group 1**, you can choose between three clock options: **external**, **on-board** or **on-chip**. - 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 **on-chip** option refers to the **internal RC oscillator** of the CPU itself. **For devices in Group 2**, you may **only** choose between on-board and external clock options. The **External RC** and **xrd Resonator** options **are not available** on the MDT1-DVP2 development board.
Temperature Tolerance	**Groups 1 & 2**	Independent of the target device used, the very large temperature tolerance scope of the ST72254 target device (-40°C to +85°C) is not applicable to the development board. The development board has been designed to function at ambient temperature.

Emulation Function/Feature	Target Device Group	Limitation or Discrepancy
Slow Mode	**Groups 1 & 2**	Slow mode will not have the same significance for devices in both groups. **For devices in Group 1**, the setting of this bit allows the use of another two bits to set the division rate of the CPU clock. **For devices in Group 2**, one bit in the Miscellaneous Register implies the division by 32 of the internal CPU clock frequency. For more information, refer to the definition of the Miscellaneous Register in the target device datasheet.
Low Voltage Detection	**Group 1**	For those target devices in **Group 1**, the Low Voltage Detection (LVD) feature is **supported only when the supply voltage is 5 V** (refer to *LVD 5 Volts* on page 48 for more details). It is implemented using the on-board push button (see *Figure 4* on page 10) which causes the chip to reset. Your application will be able to detect that the reset was caused by LVD by reading the LVDRF bit in the CRSR register. **Note** that if the LVD push button is used with a Group 2 target device, the chip will be reset, but the reason for the reset will not be recorded in any register.
Supply Voltage	**Group 1**	For target devices in Group 1, the development board application supply voltage should take a value of either 3.3 or 5 V, as opposed to any voltage in the 3 to 5.5 V range for the actual target devices. (The only allowable supply voltage for Group 2 devices is 5 V.)
Analog to Digital Converter	**Group 2**	For those devices in Group 2, the Analog to Digital Converter cell will have a longer **conversion time** in emulation than for the actual target device.
Ports A7, A5, A3, A2, A1, A0	**Group 2**	While the Development Kit allows you to set a **Push/ Pull** configuration for these ports, this is not a supported feature on actual Group 2 target devices.

Emulation Function/Feature	Target Device Group	Limitation or Discrepancy
Stack Size	**Group 2**	The emulation chip's stack is located between addresses 0x100 and 0x17F. While those target devices in Group 1 also have the stack in this same location, the target devices in Group 2 have their stack located between addresses 0x140 and 0x17F. This discrepancy between the emulation chip and Group 2 target devices has the following repercussions during an emulation/debugging session: After a "Stack Underflow" break, the stack pointer (i.e. the SP register) will indicate the address 0x100 (and not 0x140 as would occur with the actual target device). After a "Stack Overflow" break, the stack pointer (i.e. the SP register) will indicate the address 0x13F (and not 0x17F as would occur with the actual target device). Note that if a Stack Overflow break is not validated, any overflow of the stack will erase user variables (if any) situated between addresses 0x100 and 0x13F inclusive.

A SAFETY REQUIREMENTS

A.1 Testers and Tools

Any tester, equipment, or tool used at any production step or for any manipulation of semi-conductor devices must have its shield connected to GROUND.

A.2 Antistatic Equipment

An antistatic equipment should comprise:

- A conductive table top, made of steel or clean aluminum or covered by an antistatic surface (superficial resistivity equal to or higher than 0.5 MΩ/cm2), grounded through a ground cable (conductive cable from protected equipment to ground isolated through a 1 MΩ resistor placed in series).

- An antistatic floor covering grounded through a conductive ground cable (with serial resistor between 0.9 and 1.5 MΩ).

A.3 Manipulation of Finished Goods

Manipulation of finished goods must be made at a grounded worktable.

It is mandatory to wear an antistatic wrist or ankle strap, connected to the antistatic floor covering or to the grounded equipment.

It is mandatory to wear antistatic gloves or finger coats.

Nylon clothing is prohibited during manipulation of parts.

The worktable must be free of any non antistatic plastic objects.

The wearing of the antistatic strap must be controlled every day.

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 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.

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 ST7MDT1-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.

B.3   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 67).

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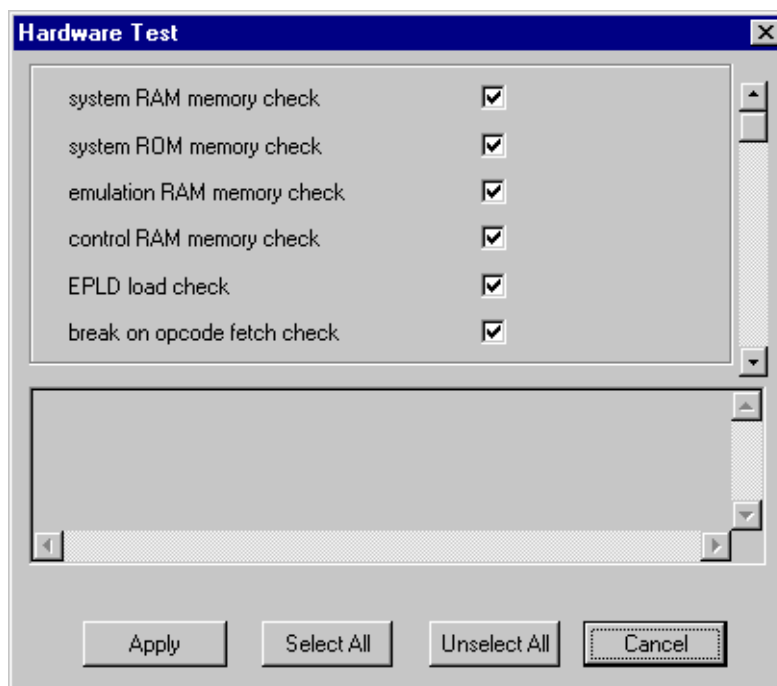
The Hardware tests will be performed one by one, and the results summarized in the dialog box as shown at right:
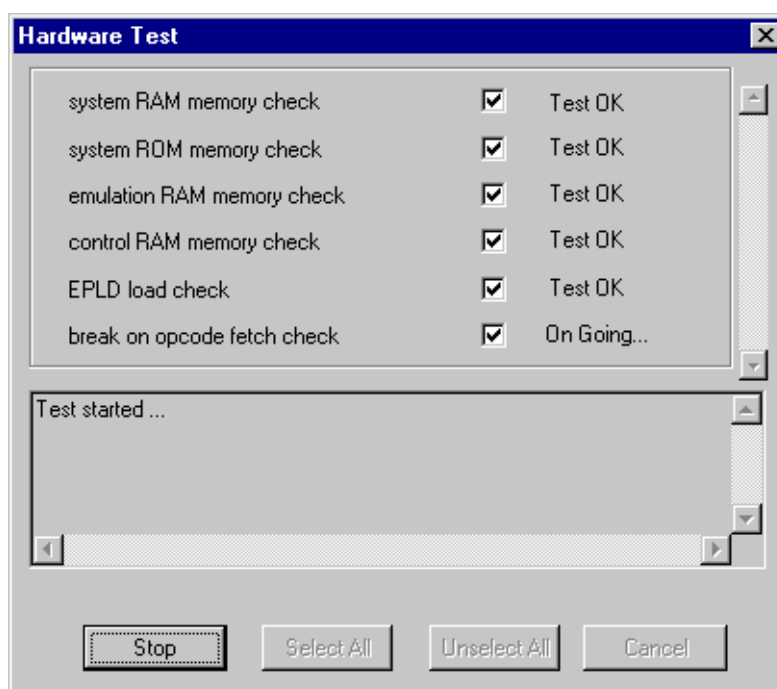
C   GLOSSARY

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 SDIP32 socket, via the In Situ Programming (ISP) port, and optionally, by installing a S028 programming socket.

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 some of the ST7 MCUs supported by the ST7MDT1-DVP2 (i.e. those in Group 1—refer to *Section 5.6.1* on page 53). 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 ST7MDT1-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 *In Situ Programming* on page 39 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 ST7MDT1-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 34-pin cables.

PC (Program Counter)

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

RC network

Resistor-capacitor network.

SO

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

STVD7 (ST7 Visual Debug)

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 MDT1-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 SDIP32 ZIF socket for SDIP32 packages, or, if installed, the SO28 ZIF socket for SO28 packages).

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 kit box. The order code will depend on the region for which it was ordered (i.e. the UK, Continental Europe or the USA).

4 Serial Number: The serial number is located on a label on the development board.

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.

Index

Index