ST7 FAMILY

ST7MDT1 TRAINING BOARD

EXERCISE MANUAL

Release 3.3

Ref: DOC-ST7MDT1-TRAIN

ST7 FAMILY

ST7MDT1 TRAINING BOARD

EXERCISE MANUAL

November 2000

REF- DOC-ST7MDT1-TRAIN

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.

MDT1 TRAINING BOARD

Exercise Manual for the ST7 MDT1 Training Board

INTRODUCTION

This Exercise Manual is intended for people getting starting with ST7 microcontrollers or writing ST7 Assembly or C Code for the first time. It provides practical guidelines for performing the exercises using the ST7 MDT1 Training Board and familiarizes users with ST7 peripherals and software tools.

A CD-ROM containing the exercise sessions is included with this document. The CD-ROM is used to install the exercise sessions on the hard disk drive of your PC along with all the environment files (located in the INIT directory) and source files for the solutions (located in the RESULT directory). The batch files used to compile, assemble and link your programs are also provided. To simplify the use of the Hiware C exercises, use the default configuration when installing the software.

To install the exercise sessions, insert the CD-ROM into the CD driver, and the autorun program is then self-executed. Click on “Install your Development Tools”, “ST7 Tools”, and finally “Exercise Session”.

The exercises that you will develop in order to become familiar with the MDT1 Training Board are described below.

Table of Contents

Table of Contents

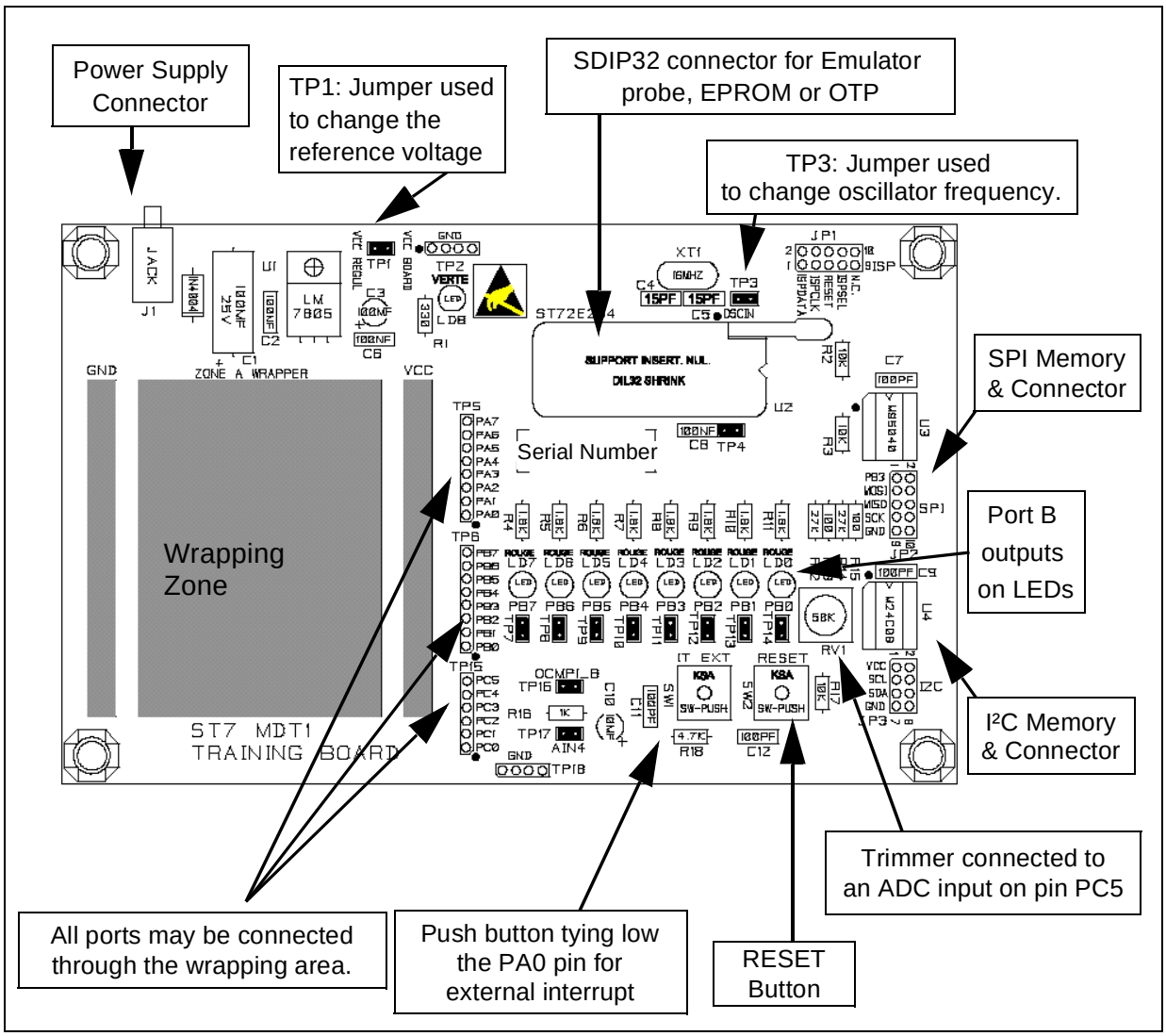
1 ST7 MDT1 TRAINING BOARD DESCRIPTION

The ST7 MDT1 Training Board is designed to support both an Emulator probe or an ST7 device (FLASH, EPROM, OTP or ROM) in a SDIP32 package. To do this, a reset system and a 16-MHz crystal are connected to dedicated pins.

1.1 OVERVIEW

The ST7 MDT1 Training Board is supplied ready-to-use in order to test and verify all the exercises given in this manual. Additional hardware functions can be implemented by installing components in the wrapping zone.

Figure 1. ST7 MDT1 Training Board

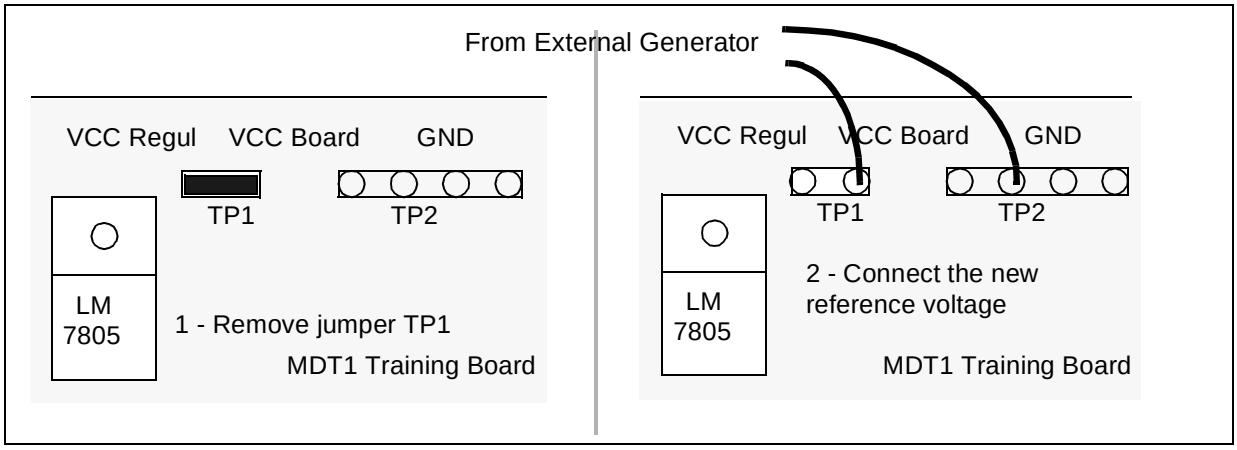


For more details, please refer to the schematic drawing in the appendix.

1.2 POWER SUPPLY

This board is supplied with an AC/DC converter and a 5-volt regulator (V_{CC} = 5 V). The reference voltage may be modified by removing jumper TP1 and connecting the new reference voltage as shown in Figure 2.

Figure 2. How to Change the Voltage Reference

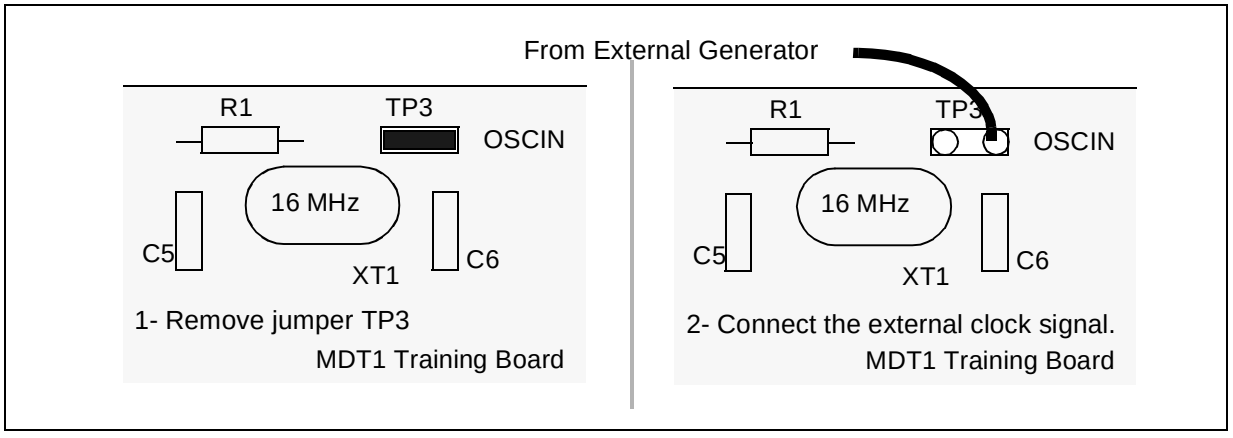


When the Emulator probe is connected, the LEDs can be activated (when bit PBx is set) and port signals will be displayed even if the board is not powered-on. The A/D Converter and the push-button switch requires a 5V power supply (pins V_{CC} & GND) to be operational.

1.3 OSCILLATION SYSTEM

The board is designed to be used with an on-board 16-MHz crystal. But, an other clock signal may be connected via the OSCIN pin. In this case, remove jumper TP3 and connect the external clock signal to the OSCIN pin as shown in Figure 3.

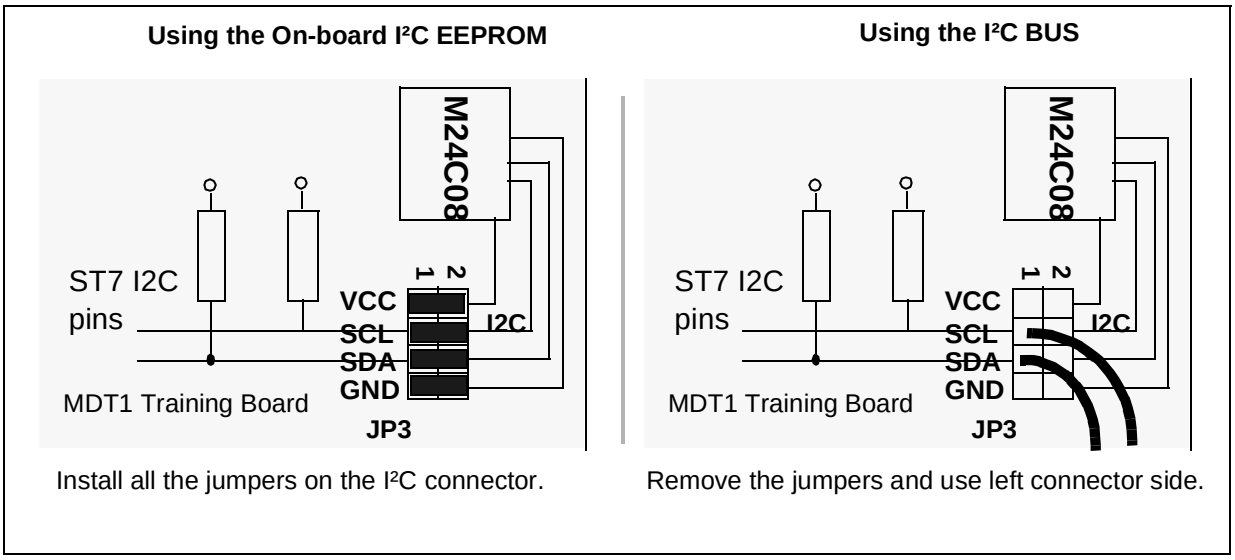
Figure 3. How to Change the Oscillator Frequency



1.4 I²C SERIAL COMMUNICATION

An external EEPROM memory with an I²C serial interface is connected to the I²C pins of the ST7 socket as shown in the following figure. The serial EEPROM may also be disconnected and the I²C bus used for other communications. This hardware configuration is used to implement application note AN971 “I²C Communications between ST7 and E2PROM”.

Figure 4. How to use the I²C Connection



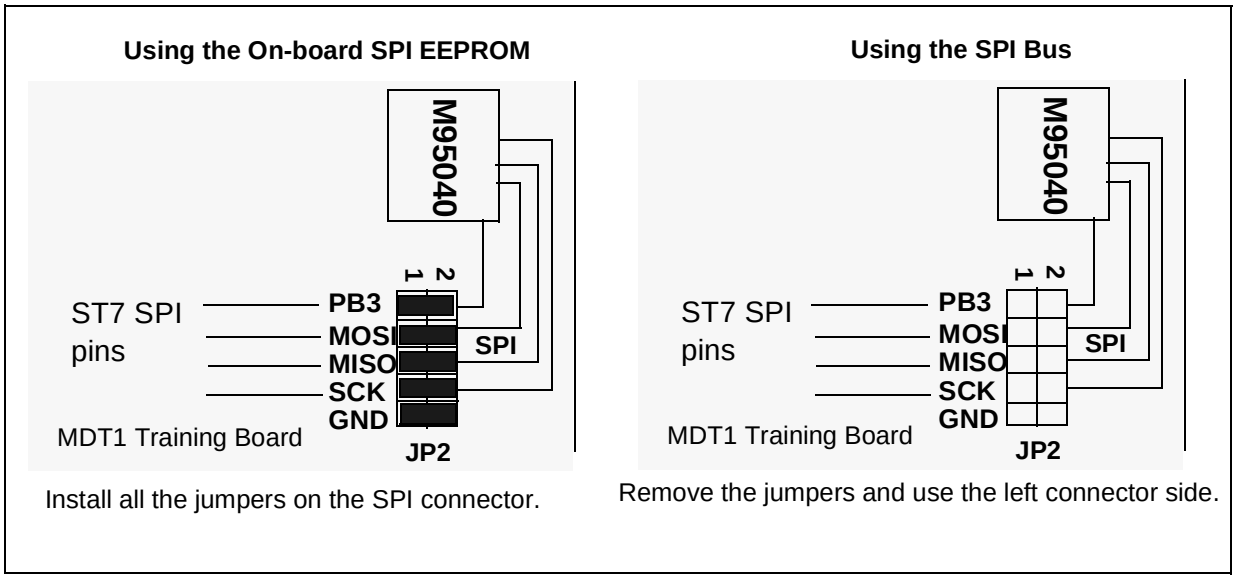
Install all the jumpers on the I²C connector.

Remove the jumpers and use left connector side.

1.5 SPI SERIAL COMMUNICATION

An external EEPROM memory with an SPI serial interface is connected to the SPI pins of the ST7 socket as shown in the following figure. The serial EEPROM may also be disconnected and the SPI bus used for other communications. This hardware configuration is used to implement application note AN970 “SPI Communications between ST7 and E2PROM”.

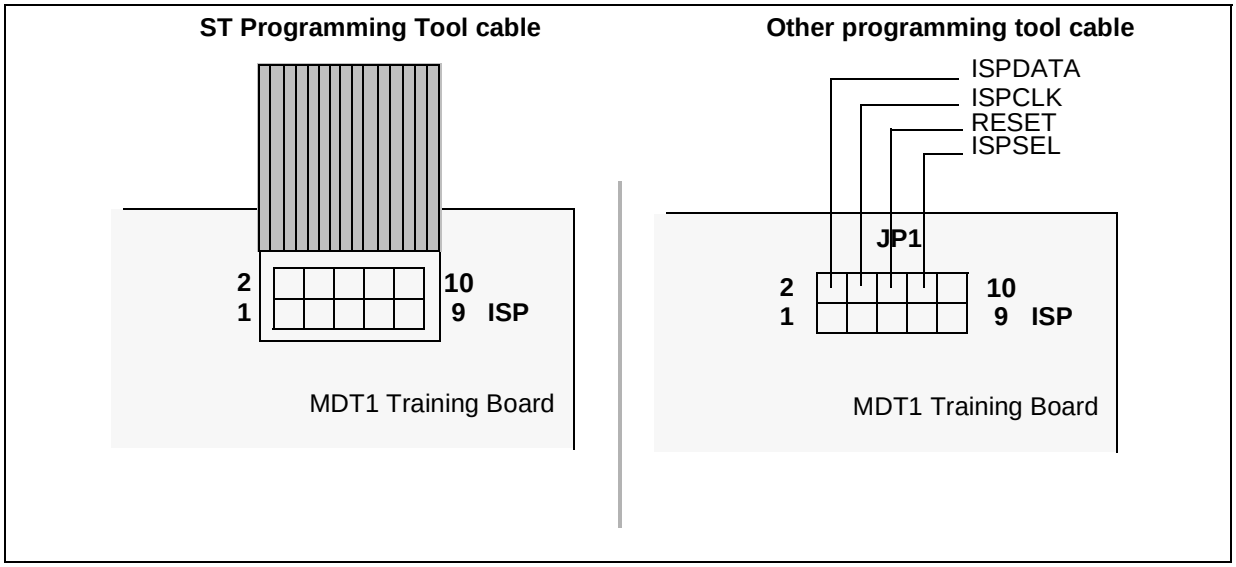
Figure 5. How to use the SPI Connection



1.6 ISP (IN-SITU PROGRAMMING) CONNECTOR

This connecter is used to program Flash devices using ISP protocol. If the ST Programming Tool is used, connector JP1 must be connected to the cable provided with the ST Programming Tool and the board has to be powered-on.

Figure 6. How to use the ISP Connector



This connector may also be used with a tool other than the one provided by ST. In this case, use the connections as described in Figure 6.

1.7 I/O PORT FUNCTIONALITY

All I/O port pins are accessible through external connectors (TP4 for port A, TP13 for port B and TP15 for port C).

1.7.1 Port A

The PA0 pin can be used to generate an external interrupt. It is connected to the ground via the push-button switch.

1.7.2 Port B

This port is connected to 8 LEDs. It can be used to visualize 8-bit words (e.g. the result of an A/D conversion).

Table 1. Port B Alternate Functions

PB0	ICAP1_A	Timer A Input Capture 1
PB1	OCMP1_A	Timer A Output Compare 1
PB2	ICAP2_A	Timer A Input Capture 2
PB3	OCMP2_A	Timer A Output Compare 2
PB4	MOSI	SPI Master Out Slave In Data
PB5	MISO	SPI Master In Slave Out Data
PB6	SCK	SPI Serial Clock
PB7	SS	SPI Slave Select

1.7.3 Port C

The PC5 pin is connected to the trimmer output. An analog voltage can be converted on Channel 5 by the A/D Converter.

The PC1/OCMP1_B pin can be connected to an analog filter via jumper TP14. The filter output can be connected to an analog input via jumper TP16.

The other pins have a double use:

– As shown on the following table, these pins are alternate functions of Timer B. They can be used to display the output signal of Timer B on an oscilloscope.

– As analog input pins.

Table 2. Port C Alternate Functions

PC0	ICAP1_B / AIN0	Timer B Input Capture 1	Analog Input 0
PC1	OCMP1_B / AIN1	Timer B Output Compare 1	Analog Input 1
PC2	CLKOUT / AIN2	CPU Clock Out	Analog Input 2
PC3	ICAP2_B / AIN3	Timer B Input Capture 1	Analog Input 3
PC4	OCMP2_B / AIN4	Timer B Output Compare 1	Analog Input 4
PC5	EXTCLK_A / AIN5	Timer A External Clock	Analog Input 5

2 SOFTWARE INSTALLATION

To do the following exercises, the following software packages must be installed.

2.1 ST7 ASSEMBLY TOOLCHAIN

The ST7 Toolchain is installed from the ST7 CD-ROM (welcome.pdf page) or directly from the ST Internet site (st7.st.com).

When the software installation is completed, click on “Finish” and reboot the PC in order for the autoexec.bat modifications to be taken into account.

2.2 STVD7 DEBUGGER

We use the new 32-bit STVD7 Debugger developed by STMicroelectronics. This is a basic IDE (Integrated Development Environment) that uses the same environment to edit, build and debug applications.

This software is available from the STMicroelectronics Internet site (mcu.st.com) or from the “MCU ON CD” CD-ROM.

2.3 REALIZER

The Realizer is a graphic software application developed by ACTUM, a third-party manufacturer. The latest version of this software, Realizer II, may be obtained by contacting ACTUM (www.actum.com).

2.4 HIWARE AND COSMIC EVALUATION VERSIONS

Both these third-parties have developed a C-Compiler for ST7 microcontrollers. An evaluation version of each compiler can be installed from the “MCU ON CD” CD-ROM.

2.5 EXERCISE SESSIONS

The exercise sessions may be installed from the “MCU ON CD” CD-ROM (“Install your Development Tools”, ”ST7 Tools” and ” Exercise Sessions”).

The exercises are located in 3 directories:

– Doc: This directory contains the exercise booklet included with the Training Board.
– Init: This directory contains all the files for each exercise. In the main modules, you will find the corresponding comments, you have to fill them with the right code.
– Result: This directory contains all the results of the exercises.

3 EXERCISES

3.1 EXERCISE 1: ASSEMBLER SYNTAX

The purpose of the first exercise is to improve your knowledge of the ST7 Assembler syntax and the STVD7 environment.

1. Launch the STVD7 Simulator:

- Configure the Toolchain paths as required. The Assembly toolchain path is already entered. For Hiware, the correct path is *C:\Hiware\prog* and for Cosmic: *C:\Program Files\cosmic software\st7eval\cxst7*. If the default paths were selected during the install procedure, a Browse function is provided for entering the correct path.
- Create a new project:
  - Click on "*File, New Workspace*" and enter the requested information (see Figure 7.)
  - Click on the "*Source Directory*" tab in the Workspace window and double click on the directory. The working directory is proposed by default.

2. Modify the files located in the following environment (*C:\exercise\init\asm*, see Figure 8.) to correct the assembler syntax. Some syntax errors have been introduced by replacing correct directives or piece of code by '?????'.
3. When the program has been corrected, **write** the batch file compile.bat to assemble, link and create a .S19 file (assign a new name to this file). The batch file also generates the S19, SYM, MAP and OBJ files. For further assistance, please refer to the Technical Manual.
4. Then, click on "*Project, Build*" or directly on the Build shortcut to build your application.
5. Detected errors are displayed in the Output window. Double-click on the error to display the line of code. Correct the errors and then rebuild.
6. In order to debug, click on the blue "D" displayed in the tool bar. Then, select "*Tools, MCU Configuration*" to configure the MCU (ST72254G2) and the option byte (Watchdog software,...).

Now, the rest is up to you...

Figure 7. STVD7 New Project

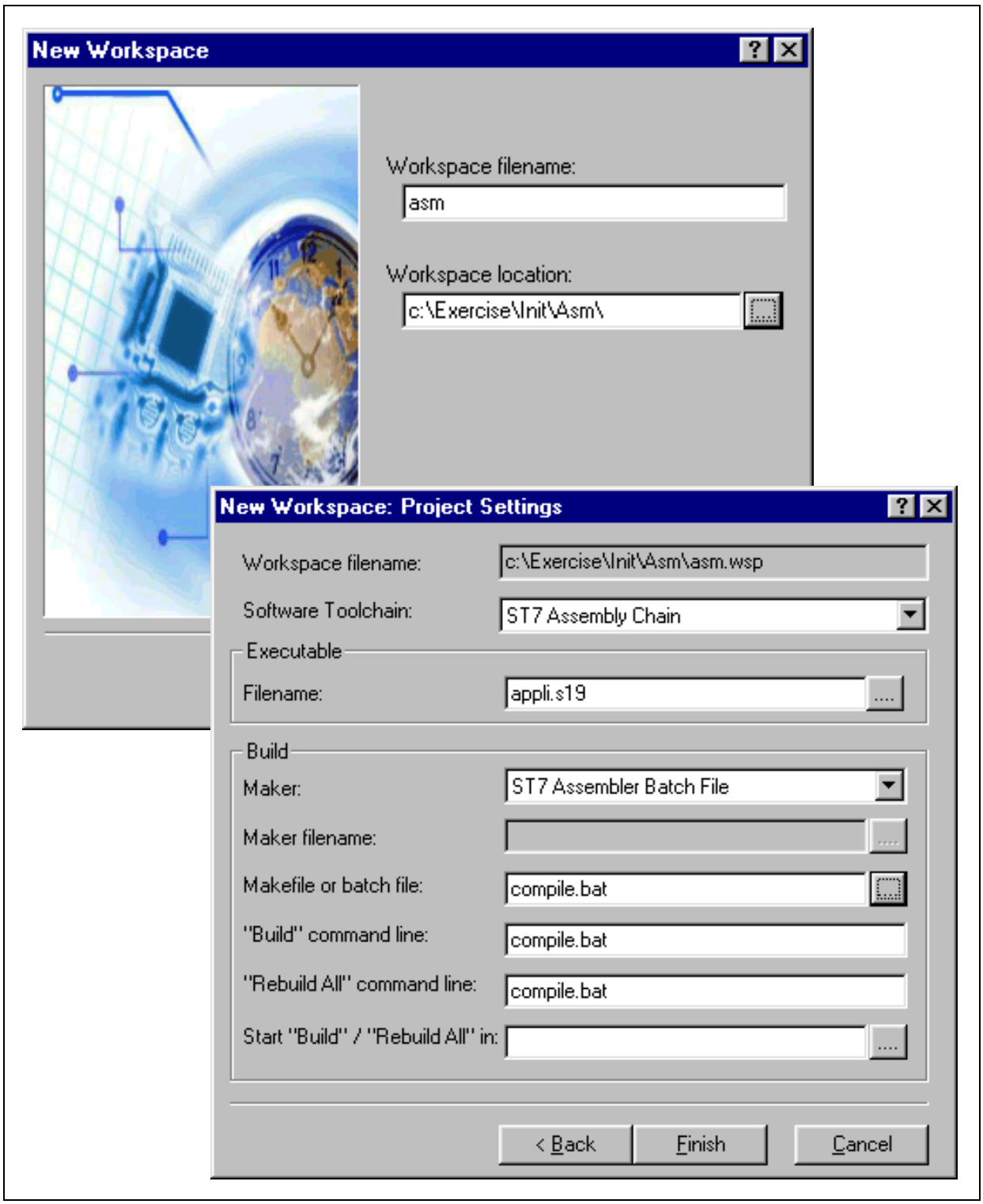
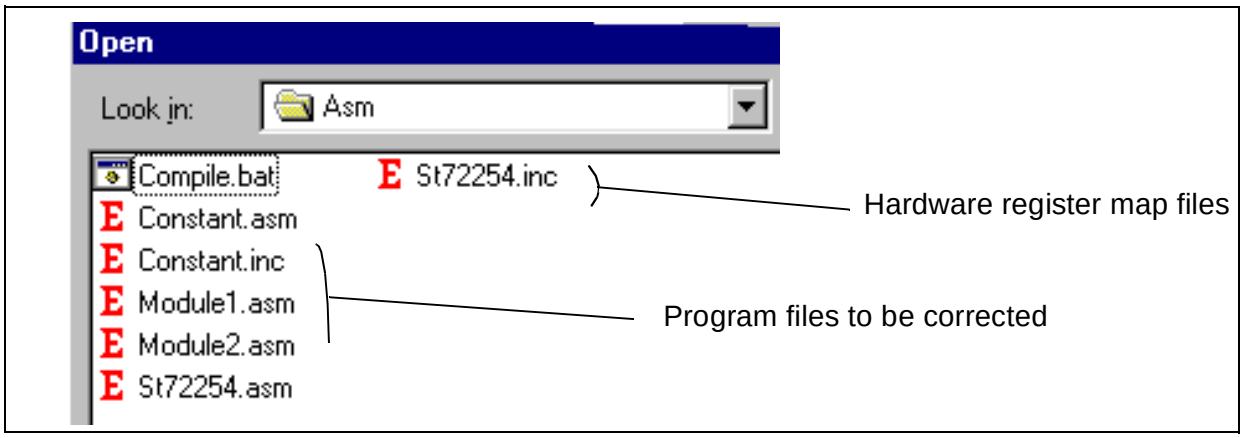


Figure 8. Environment Files



3.2 EXERCISE 2: USE OF THE 16-BIT TIMER IN ASSEMBLY LANGUAGE

Starting with a similar program frame (New Project: File, New Workspace), design a software program for the ST72254 that:

1. Generates a PWM signal on pin PB1 (frequency: 40 kHz, duty cycle: 20%).

2. Generates an interrupt every period and toggles the PB0 pin defined as a push-pull output (square wave at 20 kHz).

Refer to the ST72254 datasheet for the correct use of the timer in PWM mode.

The microcontroller uses a 16-MHz crystal, and must be configured in Normal mode. For the best accuracy, select the “Divider by 2” function for the timer clock.

- Write a main program that:
  - Configures the PB0 pin as a push-pull output.
  - Enters the calculated value for the pulse length and the period length in the TAOC1R and TAOC2R registers using the datasheet formula.
  - Configures the 16-bit timer in PWM mode.
  - Enables the interrupt process and wait for interrupts.
- Write an interrupt routine which toggles the PBO pin and clears interrupt flags ICF1and ICF2.

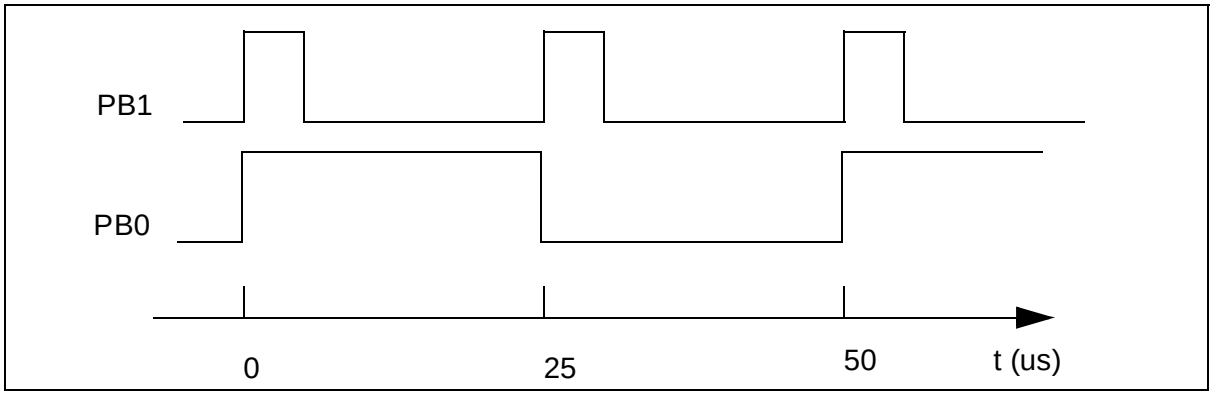
In this exercise, the entire structure is given, but only the comment lines have been kept. You have to enter the correct ST7 code by using the comments.

3. Check the validity of your software by visualizing the signal on the PB0 and PB1 pins with an oscilloscope or by using the ST7 Waveform Editor after performing an STVD7 Debugger simulation.

When an application is run on the simulator, a port.out file that contains all the pin changes is automatically generated. It is used to check the operation of the application without using any hardware.

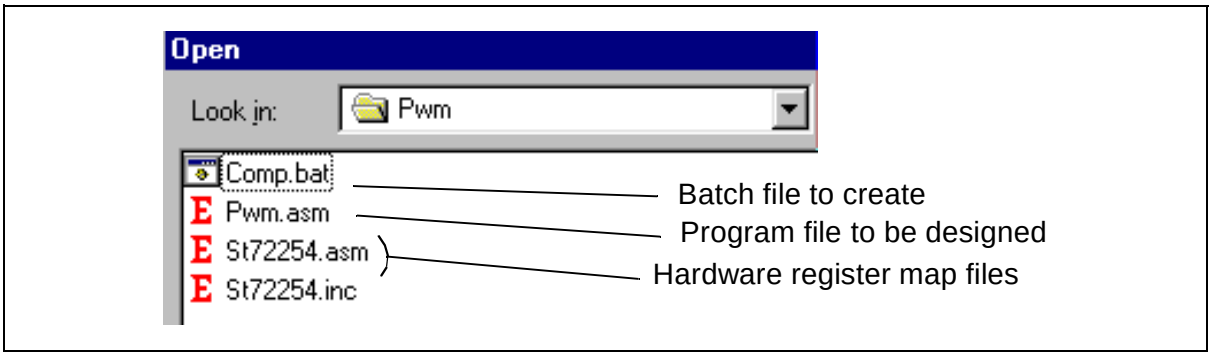
You should obtain the following timing diagram:

Figure 9. PWM Timing Diagram



The hardware registers and mapping file (ST72254.asm) are printed in the appendix. The labels of the hardware registers are also listed in this section. The environment files are shown in Figure 10.

Figure 10. Environment Files

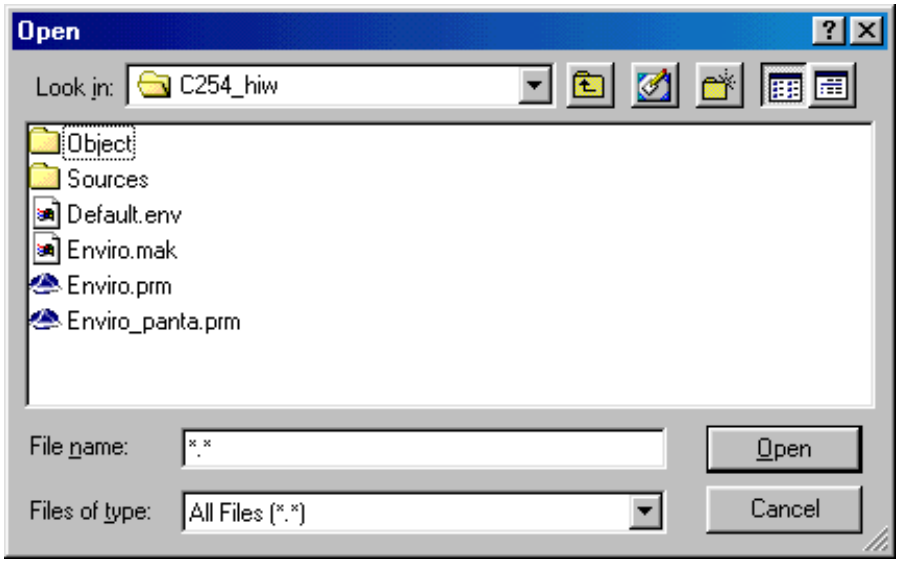


3.3 EXERCISE 3A: USE OF THE 16-BIT TIMER IN HIWARE C LANGUAGE

3.3.1 Purpose of the Exercise

The purpose of this exercise is to use the Hiware C Toolchain to write a program that performs the same PWM functions as in the previous exercise, but using the following environment.

Figure 11. Environment Files



Files that can be used for both the STVD7 and PANTA toolchains are located in the root directory.

- For STVD7: The Hiware C Toolchain paths are configured through the Default.env file. Therefore, the existing file must be adapted to selected the Hiware installation path. The linker parameter file (enviro.prm) and the makefile (enviro.mak) are also provided.
- For PANTA: The linker parameter file (enviro_panta.prm) is provided.

All files are ready-for-use, except for the main.c and itenviro.c files. In order to be able to use these files, the following procedure must be completed:

- In the main.c file, write the instructions that:
  - Configure pin PB0 as a push-pull output.
  - Enter the calculated value for the pulse length and the period length in the TAOC1R and TAOC2R registers using the datasheet formula.
  - Configure the 16-bit timer in PWM mode.
  - Enable the interrupt process and wait for interrupts.
- In the itenviro.c file, write an interrupt function that:
  - Toggles pin PBO.
  - Clears interrupt flags ICF1 and ICF2.

3.3.2 Hiware C Evaluation Program with STVD7

When using the STVD7 Debugger, carry out the following procedure:

- Create a new workspace by specifying the Hiware toolchain in the project settings.
- Enter the name of the makefile (enviro.mak) in the project settings.
- Adapt the provided default.env file, if necessary.
- Modify the source files, according to the purpose of this exercise.
- Build.

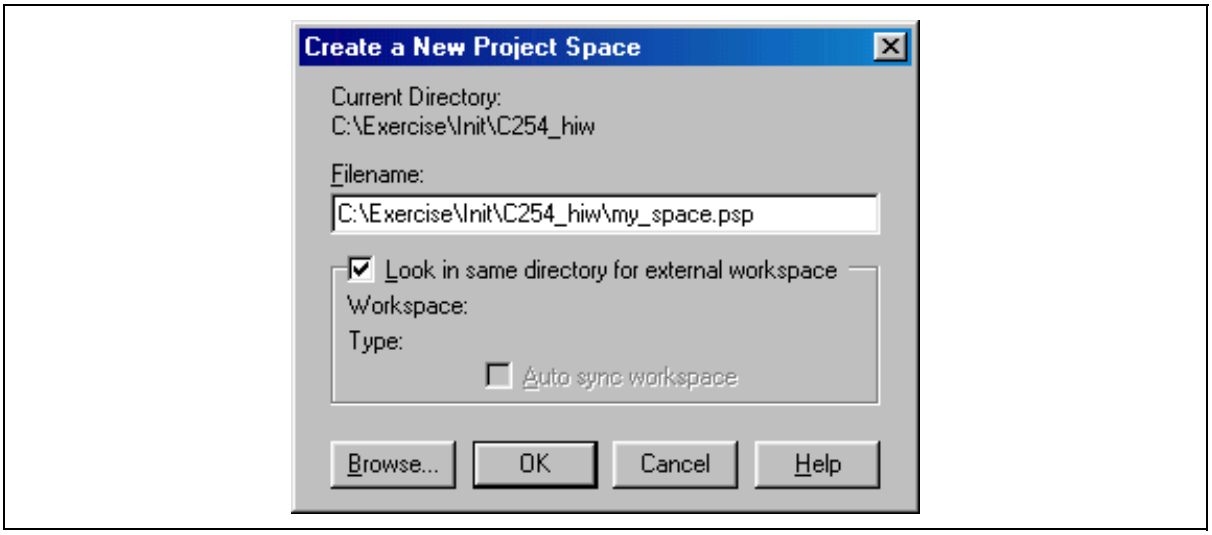
Then, you can debug your application by using STVD7 in Debug mode.

3.3.3 Hiware C Evaluation Program with PANTA

When using the PANTA IDE, carry out the following procedure:

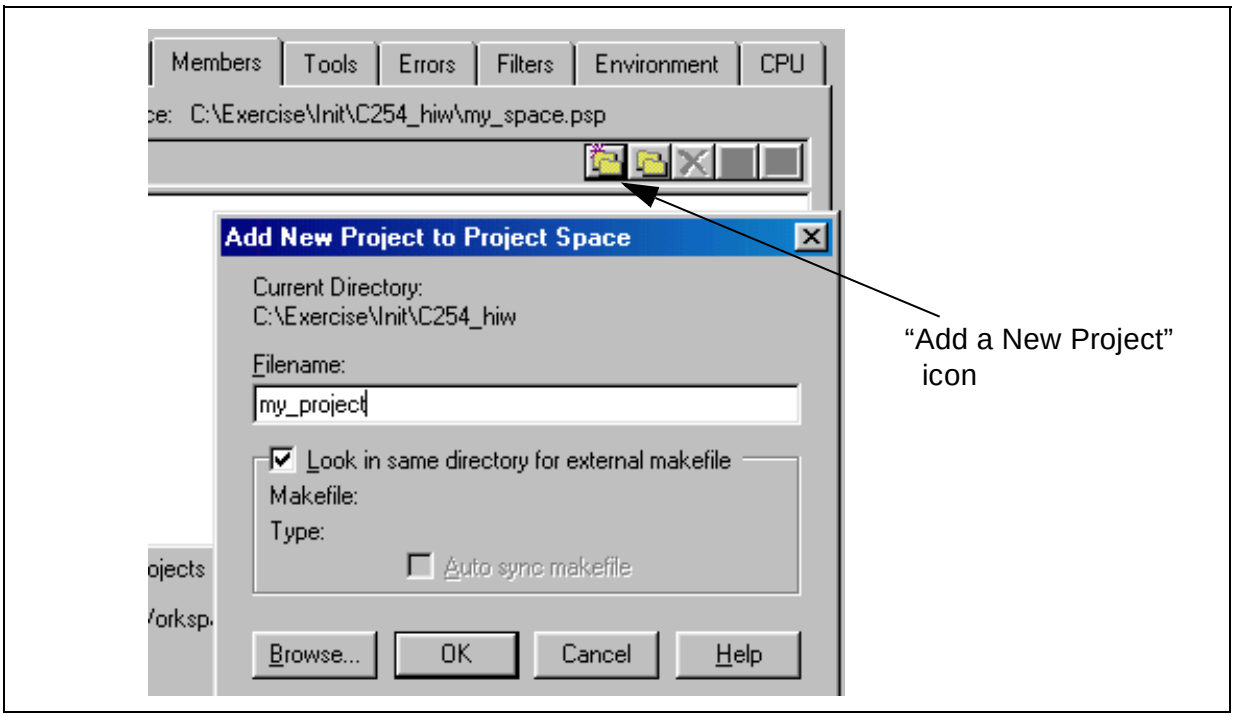
- Create a new project space. A project space is a collection of projects. Within this space, projects which are linked can be grouped. To create a new project space, open PANTA and click on "*Project, Project Space, New*".
- Enter the path of your new project space. In this example, it is *C:\Exercise\Init\C254_hiw \my_space.psp*. (You can browse your directory tree by clicking the *Browse* button.)

Figure 12. Create a Project Space



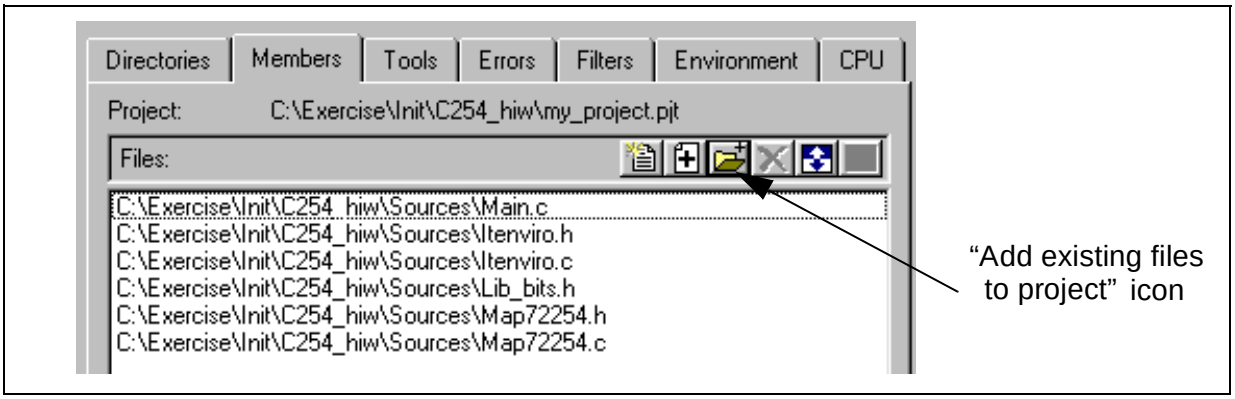
- In the Project Properties window, enter a new project (same directory) by clicking on the first icon (see Figure 13.). Choose a name for the project itself (my_project.pjt, for example).

Figure 13. New Project



- In the Project Properties window, in the Members sheet, add all required sources files (*.c, *.h, *.prm,...) by clicking on the third icon (see Figure 14.).

Figure 14. Files Added to Project



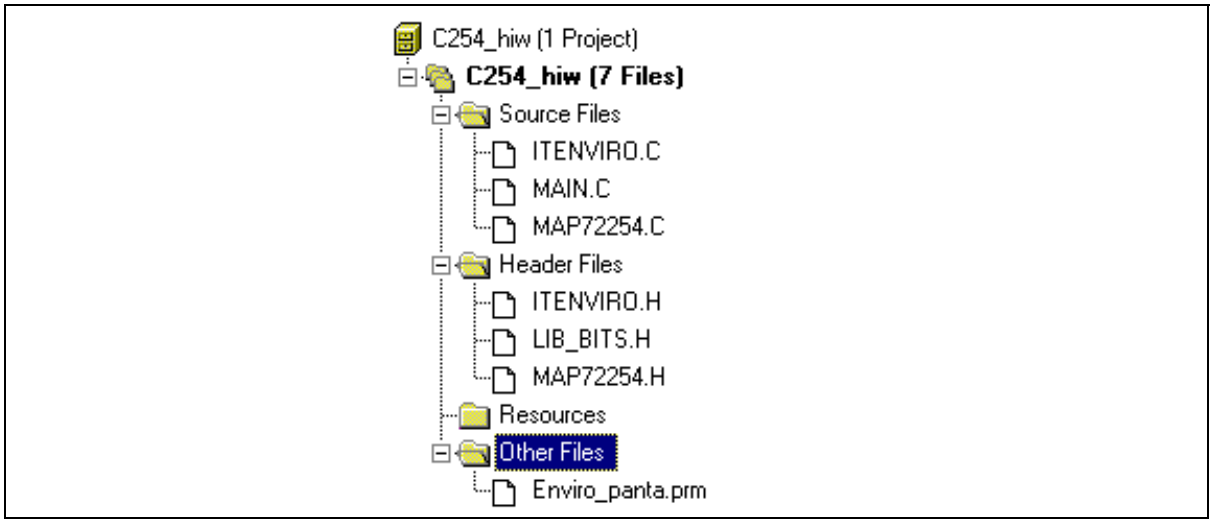
- In the CPU sheet, select ST7, leave the MCU derivative field with its default setting (except if you add your own prm file in C:\Hiware\templates\st7\prm), and select the debugging method (Simulator or ST MCU target interface for the DVP or emulator).
- In the Environment sheet (NB: this step is not necessary if all project files are located in the same directory):
  - Add a new path for the General Paths ("*.\sources*") by clicking on the white cross icon (see Figure 15.)

Figure 15. Add a Path to the Project

– Modify the existing path (just "." for the current directory) for the Object Paths ("*.\object*").
– Modify the existing path for the Text Paths ("*.\object").*
– Modify the existing path for the Absolute Paths ("*.\object*").
– Add a new path for the Header File Paths ("*.\sources*").

- In the Environment sheet, change the Link Parameter file to "*.\Enviro_panta.prm*"
- Exit the Project Properties window by clicking OK. The left panel of PANTA IDE should look like the following picture.

Figure 16. Project Files in PANTA



- Modify the source files, according to the purpose of this exercise.
- Build by pressing the F7 key or by clicking on "Project, Build". Detected errors are displayed. Double-click on the errors to automatically display the line of code. Correct the errors and then build again.

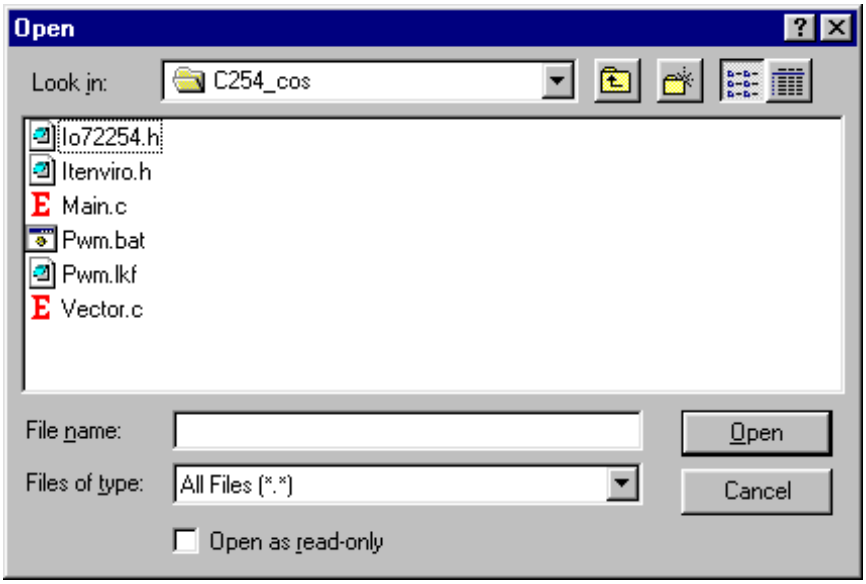
Then, you can debug your application by using the HIWAVE Debugger (select "*Project, Debug*").

You can also use STVD7 by opening STVD7, creating a new workspace and selecting the generated .abs file.

3.4 EXERCISE 3B: USE OF THE 16-BIT TIMER IN COSMIC C LANGUAGE

The purpose of this exercise is to use the COSMIC C Toolchain to write a program for performing the same PWM functions as in the previous exercise, but with the following environment.

Figure 17. Environment Files



The linker parameter file (pwm.lkf) has been put in the working directory.

You can use STVD7 Debugger or the IDE from COSMIC (IDEA). The working directory is *C:\Exercise\Init\C254_cos*.

All files are ready-for-use, except for the main.c and itenviro.c files. In order to be able to use these files, the following procedure must be completed:

- In the main.c file, write the instructions that:
  - Configure pin PB0 as a push-pull output.
  - Enter the calculated value for the pulse length and the period length in the TAOC1R and TAOC2R registers using the datasheet formula.
  - Configure the 16-bit timer in PWM mode.
  - Enable the interrupt process and wait for interrupts.
- Also in the main.c file, write an interrupt function that:
  - Toggles pin PBO.
  - Clears interrupt flags ICF1 and ICF2.

3.4.1 COSMIC C Evaluation Program with STVD7

When using the STVD7 Debugger, carry out the following procedure:

- Create a new workspace by specifying the Cosmic Toolchain in the project settings.

- Use the given batch file (pwm.bat) to build the application.
- Then, debug the application using the STVD7 Debugger in Debug mode.

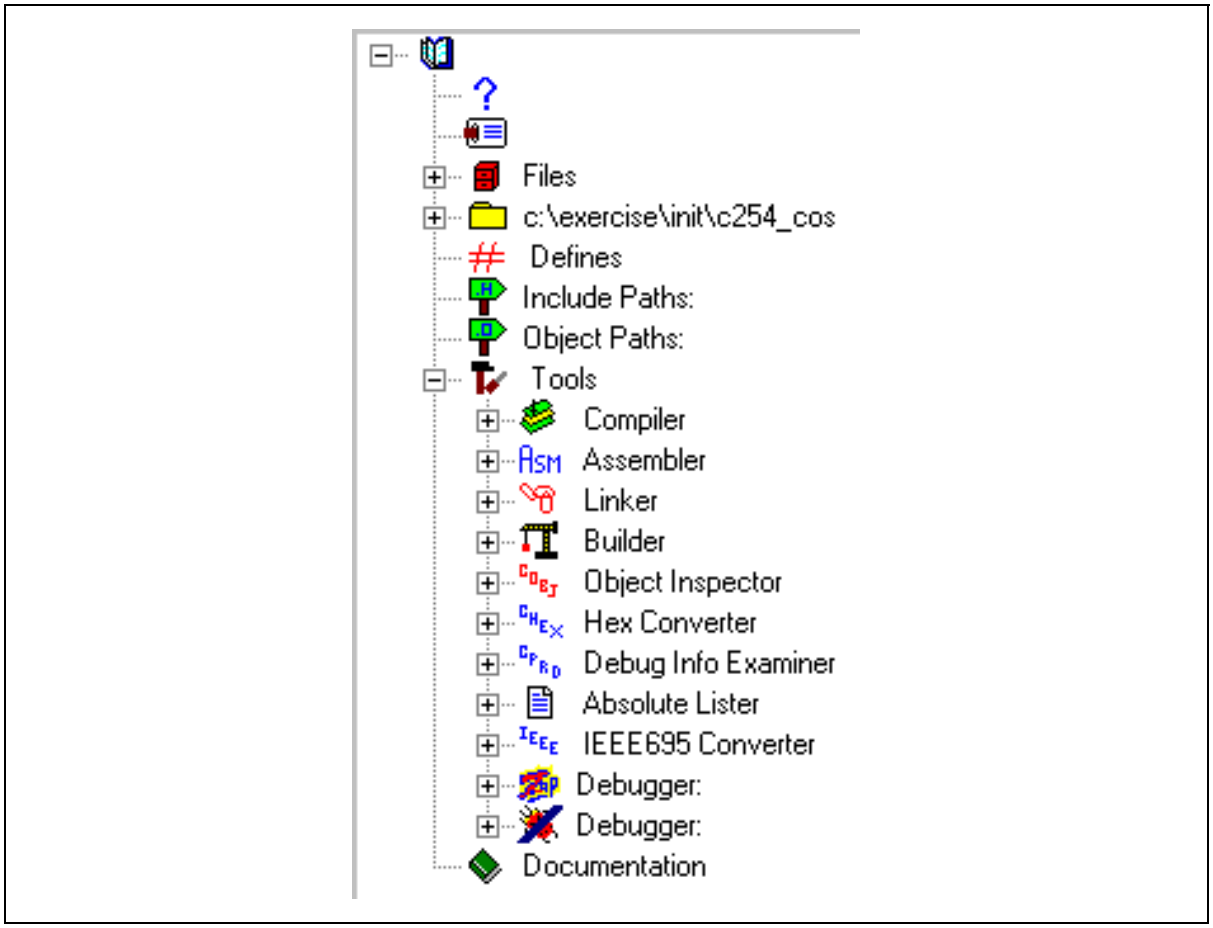
3.4.2 COSMIC C Evaluation Program with IDEAST7

Cosmic has developed an IDE called IDEA for all ST7 devices.

To use the IDEAST7, carry out the following procedure:

- Open IDEA and click on "*Project, New*".
- Right-click on the proposed path in the workspace window (by default: *C:\Program Files\cosmic software\st7eval\cxst7*, see Figure 18.) and click on Update. Browse and select the working directory. If another path is selected, modify the path name in the pwm.lkf file.
- Put all .c files in the "*Files*" directory in the workspace window (right-click on *Files* and select *Add file*) so that the make and/or build tools may be used.
- To edit files, right-click on the file to be edited and select *Edit*.

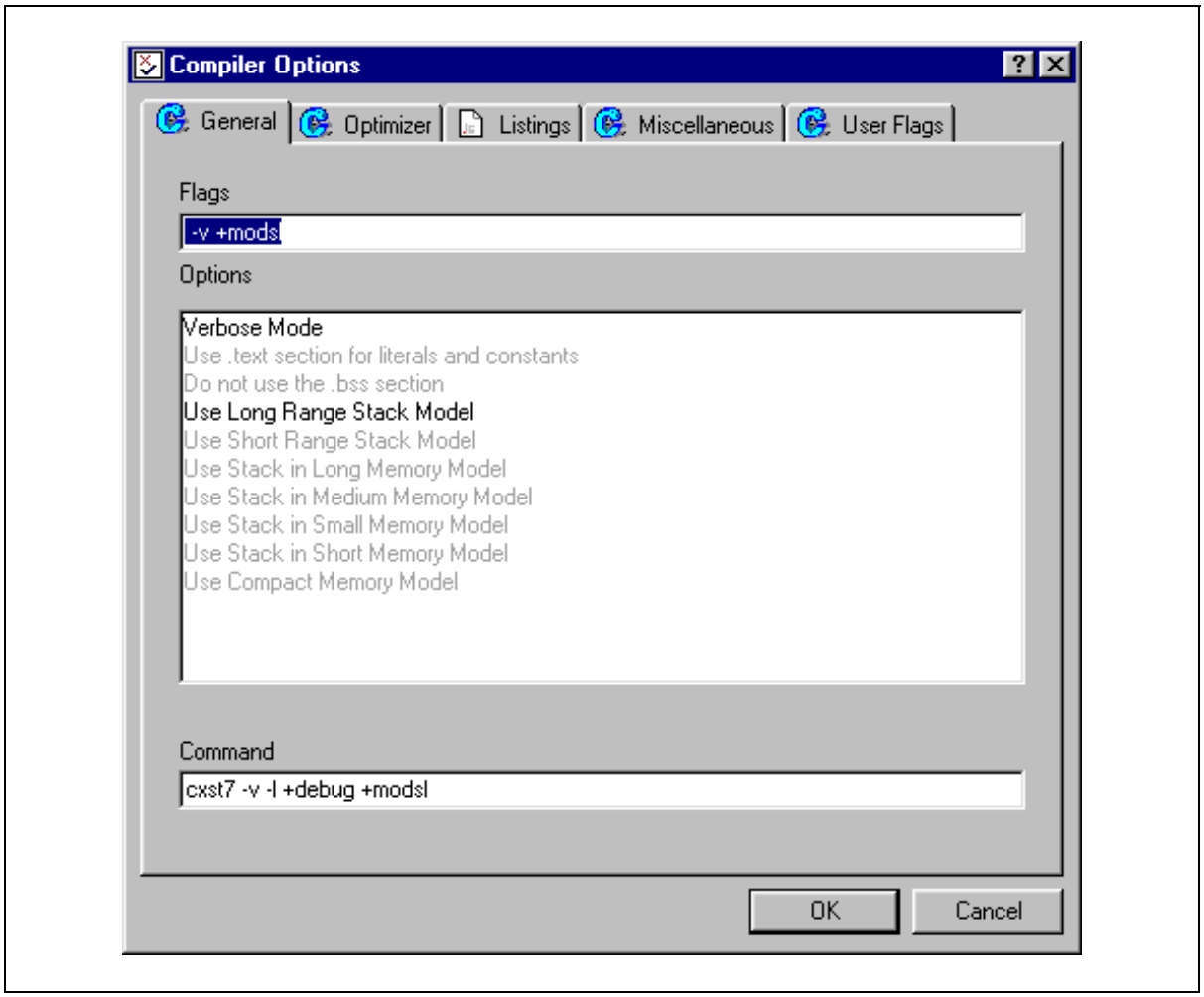
Figure 18. Workspace Window



All tools must be configured. To do this, right-clicking on each tool and select *Options*:

- Compiler (see Figure 19.):
  - In the *General* tab, select the *Long Range Stack model* as the memory model, and *Verbose* mode to be able to see all compiler passes.
  - In the *Miscellaneous* tab, select *Generate Debug Information* to be able to debug the application with ZAP or STVD7.

Figure 19. Tool Configuration



- Linker:
  - In *Output to file*, with the FIND icon, select the path and the name of your file (in the example, pwm.st7).
  - In *Command file*, select the linker file (in the example, pwm.lkf)
- Builder:
  - Check the "*Run User Utility 1*" box and enter: "cvdwarf pwm.st7 pwm.elf". This tool is the CV/DWARF convertor which generates the .elf file that must be loaded into the STVD7 in order to debug the application. Add this tool (cvdwarf.exe) along with the default ones.

When everything is configured, edit and complete the main.c file. Then, click on the Build icon to build the application.

Detected errors are displayed. Double-click on the errors to automatically display the line of code. Correct the errors and then build again.

Select the debugger in the Tools menu in the workspace window and then launch it (for example, the STVD7 simulator in order to generate the port.out file).

3.5 EXERCISE 4: BASIC USE OF THE ADC

The purpose of this exercise is to write a section of code that is able to read an analog voltage. This voltage is the output of the trimmer and is connected to the PC5 pin which is the analog channel 5. The result of the conversion (an 8-bit word) must be output on the LEDs connected to port B.

The ST72254 environment in assembly language, the structure of the main module and the batch file for compiling efficiently is located in the ADC directory.

The program in the exo4.asm module must be written as follows:

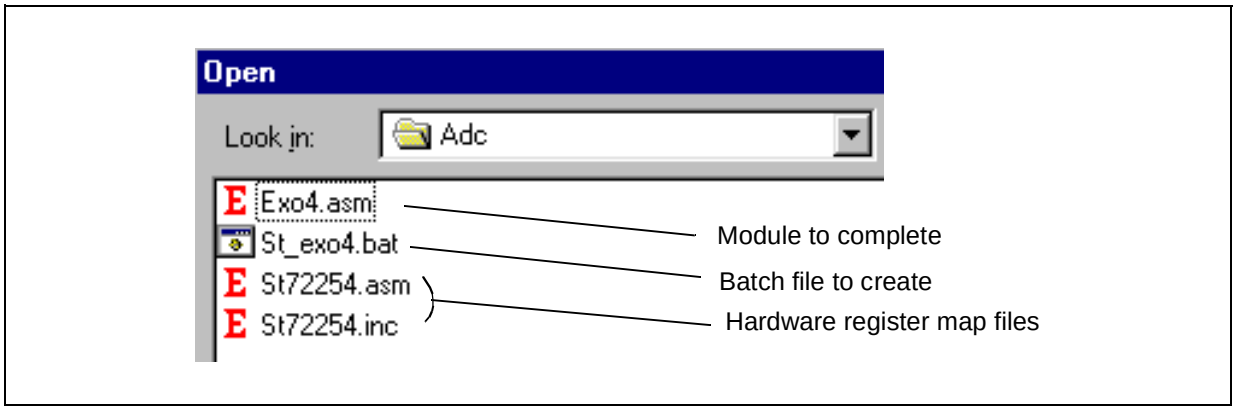
– An ‘init’ sub-routine where port B and PC5 have to be initialized,

– An ‘init-adc’ sub-routine where the ADC is initialized, including a 30 µs wait loop.

– The main program which waits for the end of the current conversion and outputs the result on port B.

To begin, create a new project under STVD7. The working environment file is described in Figure 20.

Figure 20. Working Environment



3.6 EXERCISE 5: SWITCH ON A LED EVERY 0.5 SECONDS

In Assembly language, this exercise can be done in 3 steps.

– Switch on and off a LED every 0.5 seconds.
– Shift the LED from PB_x to PB_{x+1} with the previous time base.
– Double the frequency by pressing the push button switch on the board.

The results of each step of your application can be checked on the Training Board.

3.6.1 Step 1

The purpose of the first step is to use the Output Compare capability of the timer to create a real-time clock. The internal time basis is 0.5 sec. Every 0.5 seconds, a timer interrupt switches on a LED connected to PB0 and switches it off after the same period.

The microcontroller uses a 16-MHz crystal, so it must be configured in Slow mode and the “Divider by 8” function selected for the timer clock.

- Write a routine called “init” which contains the port B0 configuration (push-pull output) and puts the MCU in Slow mode.
- Write a routine called “init_timer” which contains the code used to configure the timer (output compare interrupt enable, $f_{timer} = f_{cpu} / 8$) and fills the Output Compare 1 register with the time base value. This value is declared as a word and is initialized in the constant files.
- Write the “tima_rt” timer interrupt routine which checks if the OCF1 bit of the status register generates the interrupt, switches the LED connected to PB0 on or off, adds the value corresponding to 0.5 sec to the contents of TAOC1R register and clears the OCF1 or OCF2 bit to clear the interrupt request.
- The main program calls the two initialization routines and enters an infinite loop after resetting the interrupt mask.

3.6.2 Step 2

Using the same time base and the previous code:

- Modify the timer interrupt routine to shift the activated LED from PB_x to PB_{x+1}. The content of port B is stored in the 8-bit “shiftdata” word. This data has to be declared in the constant files and initialized in the main program.
- In the “init” routine, configure all port B pins as push-pull outputs.

There will be no wait time between the moment the activated LED switches from PB7 to PB0.

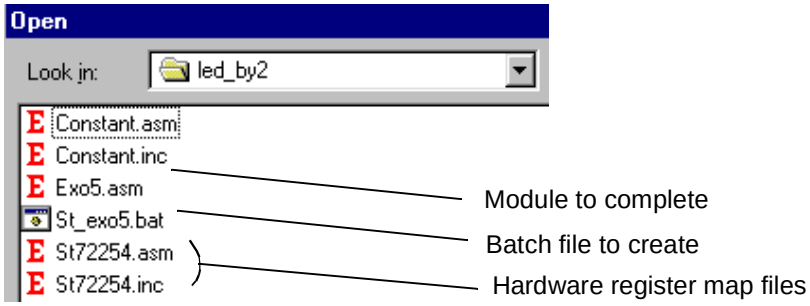
3.6.3 Step 3

Using an external interrupt on PA0 (pressing the push-button to hold the PA0 pin low) double or divide by 2 the timer frequency.

- Write the ‘ext0_rt’ external interrupt routine which processes the frequency change each time it detects that the button has been pressed. The timer frequency can be changed either by modifying the CC0 and CC1 bits of the TACR1 register or the TAOC1R register value.
- Modify the “init” routine to configure the PA0 pin as an input with interrupt and to select the rising or falling edge interrupt sensitivity.

To begin, create a new project under STVD7. The working environment file is described in Figure 21.

Figure 21. Environment File



3.7 EXERCISE 6: ADC AND LED WITH REALIZER

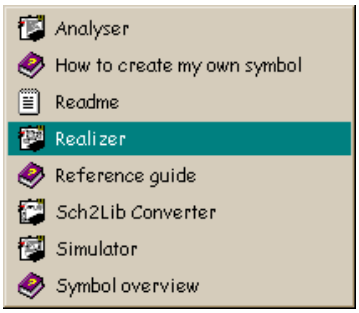
This exercise uses the REALIZER software to read the analog input of a port pin (PC5) and to display the voltage value on LED indicators as follows:

0V- 1.5V	1st LED connected to PB0
1.5V- 3.5V	2nd LED connected to PB1
3.5V- 5V	3rd LED connected to PB0

3.7.1 Installing ST7 REALIZER Software on your PC

Double-click on the REALIZER icon to start the tool after having installed the software.

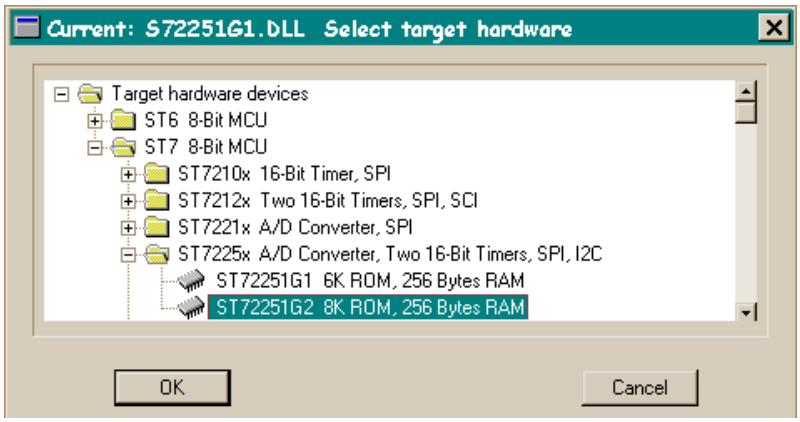
Figure 22. REALIZER Menu



3.7.2 Drawing the Application Schematic Diagram

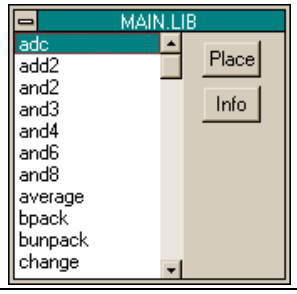
- Select the following from the *File* menu:
  - *New Project*: Enter the name of the directory where you would like your schematic diagram to be stored (c:\exercise\init\realizer\).
  - *New*: Enter the filename of the schematic diagram (adcode.sch).
- Select the ST72254G2.dll hardware from the *Options* / *Select Hardware* menu.

Figure 23. Target Hardware Selection Menu



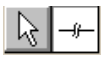
- Select the symbols from the *Object* / *Library Symbol* / *Mainlib* menu.

Figure 24. Main Library Menu



Select and place the symbols using these windows. You may also obtain useful information about the symbols by clicking on *Info*.

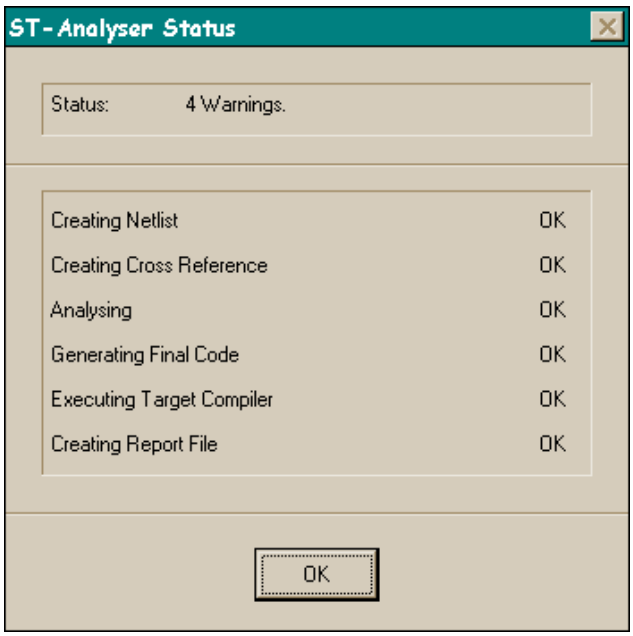
- Place them on your diagram. In this application, the *adc, comp, inv, or, and, digin* and *digout* symbols may be used. Connect them with wires using the smart icons.



3.7.3 Analysing the Created Schematic Diagram

Once the schematic diagram is finished, launch the *Analyse* / *Go* menu and check for any errors. The following window may be displayed.

Figure 25. ST Analyser Status Window

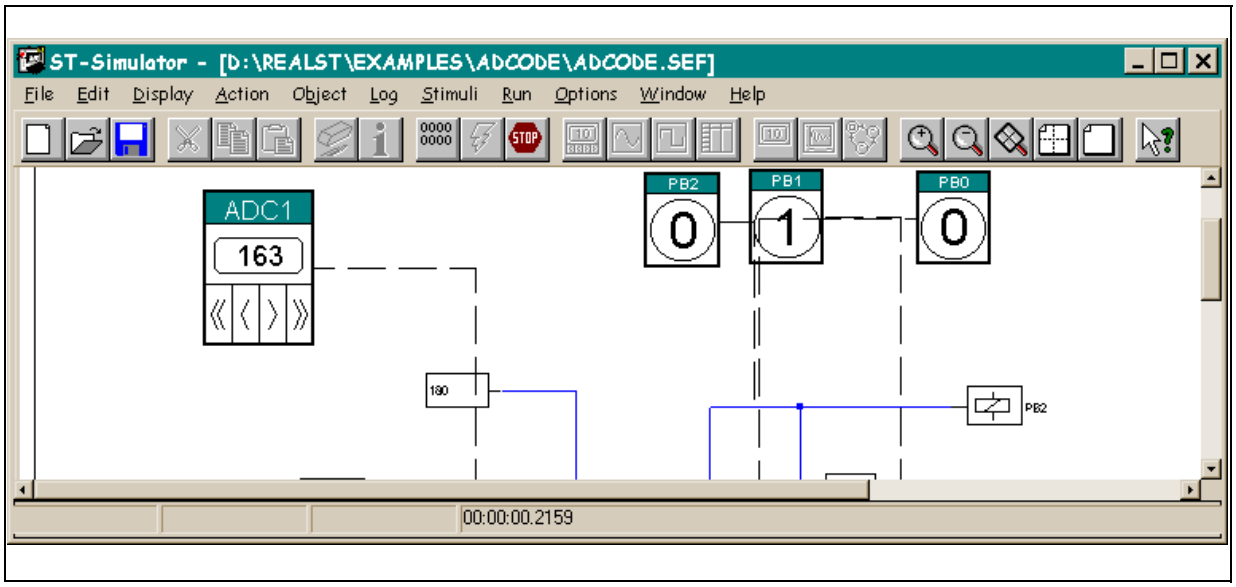


This action will create the asm, lst, map, and S19 files. The adcode.s19 file can be directly used to burn a chip with the Windows EPROMer or it can be loaded in the WGDB7 debugger.

3.7.4 Simulating the Created Schematic Diagram

- Launch *Simulate* / *Go menu. I*f no error is detected during *Analyse*, a new window will be displayed.
- Select *File* / *New* and enter the filename of the schematic diagram (adcode.sch). The schematic diagram will be displayed on the screen.
- Click on the wires to be monitored. Input and Output probes will be suggested (Bold Icons in menu bar). Select the one required.
- Once all probes are implemented, launch *Run* / *Initialise* and *Start* the simulation. Check the correct behaviour of your application through the schematic simulation window.

Figure 26. ST Simulator Window



3.7.5 Checking with the ST7MDT1 Training Board

Check that the adcode.asm, adcode.lst and acode.s19 files have been generated. Load the WGDB7 with the adcode.s19 file and check the application by plugging the emulator probe on the MDT1 Training Board.

You can also burn a chip with an EPB, a starter kit or a development kit using the Windows EPROMer software.

4 APPENDIX 1: ST72254 REGISTER AND MEMORY MAPPING FILE

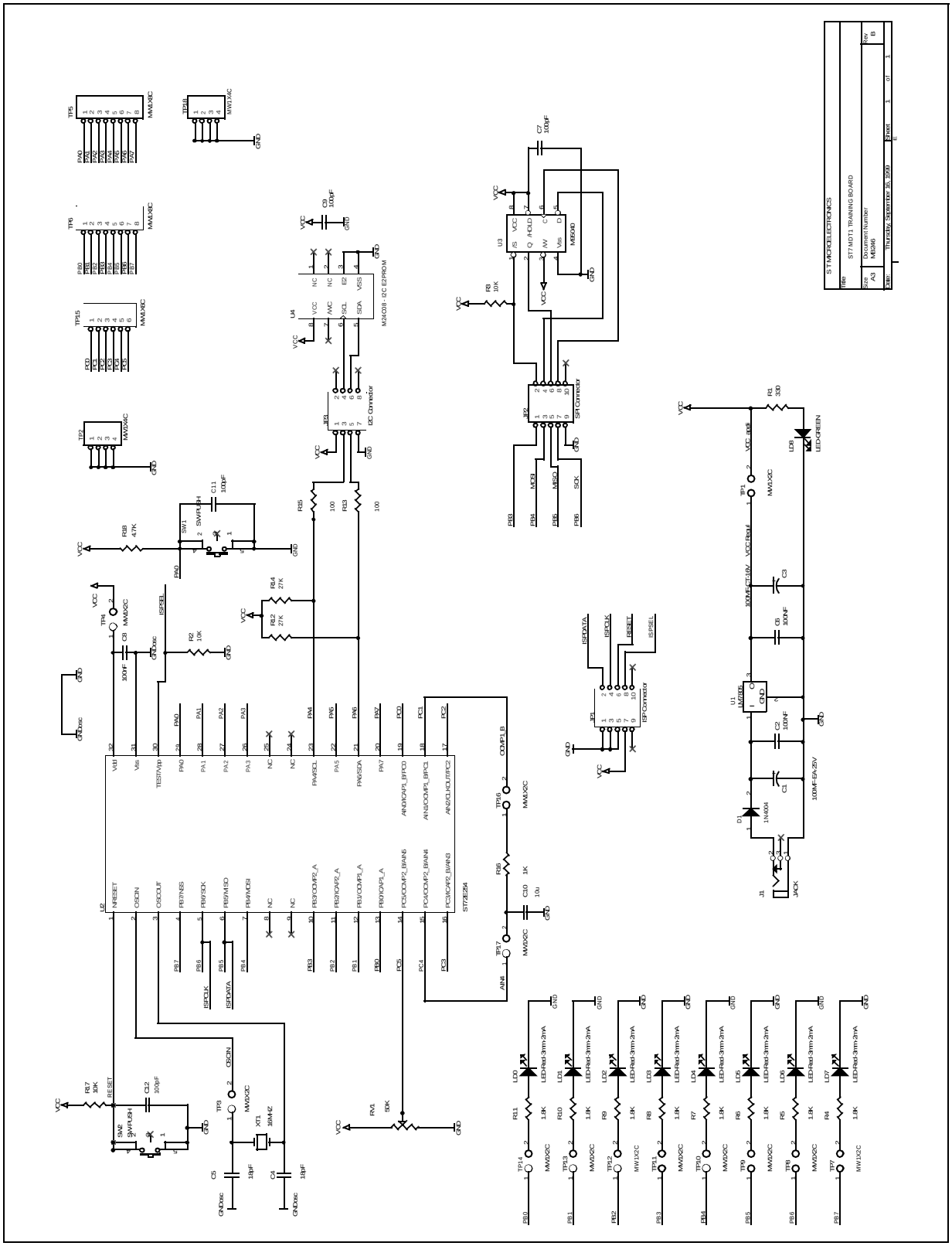
The following lists the contents of the ST72254.asm hardware register file.

```
st7/
;******************************************************************************
; TITLE:                ST72254.ASM
; AUTHOR:               CMG Microcontroller Application Team
; DESCRIPTION:          ST72254 Register and memory mapping
;******************************************************************************

  BYTES                 ; following addresses are 8 bit length
;*****************************************************************
  segment byte at 0-71 'periph'
;*****************************************************************
;*****************************************************************************
;             I/O Ports registers
;*****************************************************************************
.PCDR   DS.B                    1       ; port C data register
.PCDDR  DS.B                    1       ; port C data direction register
.PCOR   DS.B                    1       ; port C option register
                        DS.B            1       ; not used
.PBDR   DS.B                    1       ; port B data register
.PBDDR  DS.B                    1       ; port B data direction register
.PBOR   DS.B                    1       ; port B option register
        DS.B                    1       ; not used
.PADR   DS.B                    1       ; port A data register
.PADDR  DS.B                    1       ; port A data direction register
.PAOR   DS.B                    1       ; port A option register
        DS.B                    1       ; not used

reserved0  DS.B  20                     ; unused
;*****************************************************************************
;             Miscellaneous registers
;*****************************************************************************

.MISCR 1 DS.B                   1       ; miscellaneous register
;*****************************************************************************
;             SPI registers
;*****************************************************************************
.SPIDR  DS.B                    1       ; SPI data register
.SPICR  DS.B                    1       ; SPI control register
.SPISR  DS.B                    1       ; SPI status register
;*****************************************************************************
;             Watchdog register
;*****************************************************************************
.WDGCR  DS.B                    1       ; watchdog register
reserved1 DS.B 3                        ; unused
;*****************************************************************************
;             I2C registers
;*****************************************************************************
.I2CCR  DS.B                    1       ; i2c control register
.I2CSR1 DS.B                    1       ; i2c status register 1
.I2CSR2 DS.B                    1       ; i2c status register 2
```

```
.I2CCCR DS.B                    1       ; i2c clock control register
.I2COAR1 DS.B                   1       ; i2c own add register 1
.I2COAR2 DS.B                   1       ; i2c own add register 2
.I2CDR  DS.B                    1       ; i2c data register
reserved3 DS.B                  2       ; unused
;***********************************************************************
;             timer A registers
;***********************************************************************

.TACR2  DS.B                    1       ; timer A control register 2
.TACR1  DS.B                    1       ; timer A control register 1
.TASR   DS.B                    1       ; timer status register
.TAIC1HR DS.B                   1       ; timer A input capture 1 high register
.TAIC1LR DS.B                   1       ; timer A input capture 1 low register
.TAOC1HR DS.B                   1       ; timer A output compare 1 high register
.TAOC1LR DS.B                   1       ; timer A output compare 1 low register
.TACHR  DS.B                    1       ; timer A counter high register
.TACLR  DS.B                    1       ; timer A counter low register
.TAACHR DS.B                    1       ; timer A alternate counter high register
.TAACLR DS.B                    1       ; timer A alternate counter low register
.TAIC2HR DS.B                   1       ; timer A input capture 2 high register
.TAIC2LR DS.B                   1       ; timer A input capture 2 low register
.TAOC2HR DS.B                   1       ; timer A output compare 2 high register
.TAOC2LR DS.B                   1       ; timer A output compare 2 low register

MISCR2  DS.B                    1       ;Miscellaneous register 2

;***********************************************************************
;             timer B registers
;***********************************************************************
.TBCR2  DS.B                    1       ; timer B control register 2
.TBCR1  DS.B                    1       ; timer B control register 1
.TBSR   DS.B                    1       ; timer B status register
.TBIC1HR DS.B                   1       ; timer B input capture 1 high register
.TBIC1LR DS.B                   1       ; timer B input capture 1 low register
.TBOC1HR DS.B                   1       ; timer B output compare 1 high register
.TBOC1LR DS.B                   1       ; timer B output compare 1 low register
.TBCHR  DS.B                    1       ; timer B counter high register
.TBCLR  DS.B                    1       ; timer B counter low register
.TBACHR DS.B                    1       ; timer B alternate counter high register
.TBACLR DS.B                    1       ; timer B alternate counter low register
.TBIC2HR DS.B                   1       ; timer B input capture 2 high register
.TBIC2LR DS.B                   1       ; timer B input capture 2 low register
.TBOC2HR DS.B                   1       ; timer B output compare 2 high register
.TBOC2LR DS.B                   1       ; timer B output compare 2 low register

reserved5 DS.B 32    ; unused
;***********************************************************************
;             ADC registers
;***********************************************************************
.ADCDR  DS.B                    1       ; adc data register
.ADCCSR DS.B                    1       ; adc control status register
  ; DS.B  14

;***********************************************************************
```

```
  segment byte at 80-FF 'ram0' ;Zero Page
;*****************************************************************
  WORDS                              ; following addresses are 16 bit length

;*****************************************************************
  segment byte at 100-17F 'stack'
;*****************************************************************
;*****************************************************************
  segment byte at E000-FFDF 'rom'
;*****************************************************************
;*****************************************************************
  segment byte at FFE0-FFFF 'vectit'
;*****************************************************************
    end
```

5 APPENDIX 2 : MDT1 TRAINING BOARD SCHEMATIC DIAGRAM



"THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNEXION WITH THEIR PRODUCTS."