

Interfacing the MPC17C724 with an 8-bit Processor

By: Jorge Ramírez Rivero
RTAC Americas
Mexico

1 Purpose

The main purpose of this application note is to explain the way to interface the MPC17C724 stepper motor driver with an 8 bit HCS08 microcontroller, and how to control it with instructions from the CPU's parallel port. This note has detailed instructions on how to use the MPC17C724 and all the tools needed to interface it.

2 Scope

The MPC17C724 is a compact monolithic dual channel H-Bridge power IC, ideal for portable electronic applications containing bipolar stepper motors or brush DC motors such as those used in camera lenses and shutters. Its Shoot-Through Current Prevention Circuit avoids damage to the component when the load is demanding it too much current. The four driving modes it manages make it a great solution on stepper motor driving.

Contents

1 Purpose	1
2 Scope	1
3 Introduction	2
4 Overview	2
4.1 MPC17C724 Overview	3
4.2 MC9S08QG8 Overview	6
4.3 MC9S08QG8 Demo Board	7
4.4 The 17C724 EVM	13
4.5 Source Code	18
5 References	30

3 Introduction

This document explains the operation of an evaluation platform that highlights the advantages of the MPC17C724, and also the integration of the HCS08QG8 ('QG9) Evaluation Module. This evaluation platform consists of an interface between a CPU (PC/Computer) and a simple plotter via the CPU's parallel port, using an MC9S08QG8 microcontroller, and the MPC17C724 drivers to control the stepper motors for the plotter pens.

This evaluation platform uses a freeware named CStep to create drawings using the CPU and send signals through the CPU's parallel port to move a 3-axis plotter that makes the drawing created on the computer. The source code that the MC9S08QG8 needs to translate the signals of the CPU's parallel port to control the plotter's stepper motors is also provided.

4 Overview

The following figure shows the block diagram of the EVM.

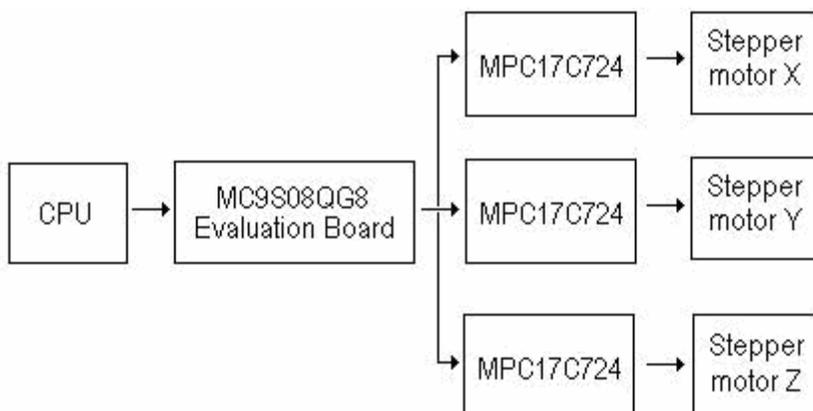


Figure 1. Block diagram of the application

As mentioned above, the drawing is created on the CPU monitor using the CStep application program. A brief tutorial of how to use CStep is included in this document. CStep interfaces the CPU with the 'QG8 evaluation board via the CPU's parallel port. The 'QG8 evaluation board is then connected to the MPC17C724 EVM board. The 'QG8 EVM board sends the parallel port input signals to the MC9S08QG8, which then translates them into signals to control the plotter pen stepper motors, which are sent to the MPC17C724 board. The MPC17C724 drivers drive the plotter pen stepper motors to create the drawing. The plotter has three stepper motors that move a pencil, which is used to make a drawing in the XY plane. The purpose of the motor for the Z axis is only to lift the pencil up and down.

4.1 MPC17C724 Overview

4.1.1 0.4 A Dual H-Bridge Motor Driver IC

The 17C724 can operate efficiently with supply voltages from 2.7 V to 5.5 V and can provide continuous motor drive currents of 0.4 A with low RDS (ON) of 1.0 Ω . It is easily interfaced to low-cost MCUs via parallel 3.0 V - or 5.0 V compatible logic and has built-in shoot-through current protection circuit and undervoltage detector to avoid malfunction.

The 17C724 has four output control modes: Forward, Reverse, Brake, and Tri-State (High Impedance). The H-bridge outputs are designed to be independently Pulse Width Modulated at up to 200 kHz for speed/torque and current control.

4.1.2 Features

- Manufactured in SMOS7 Process Technology
- Built-In 2-Channel H-Bridge Driver
- Provides 4 Driving Modes (Forward, Reverse, Brake, and High Impedance)
- Direct Interface to MCU. Low ON-Resistance, RDS (ON) = 1.0 Ω (Typical)
- Dual Channel Parallel Drive, RDS (ON) = 0.5 Ω (Typical)
- Output Current Driver (IDR) is 400 mA (Continuous)
- Low Power Consumption
- Built-In Shoot-Through Current Prevention Circuit
- Built-In Low-Voltage Shutdown Circuit
- PWM Control Frequency 200 kHz (Max)
- Very Compact Size, Comes in 16-Terminal QFN Package (3 x 3 mm; Terminal Pitch: 0.5 mm)
- Pb-Free Packaging Designated by Suffix Code EP

4.1.3 Pin connections

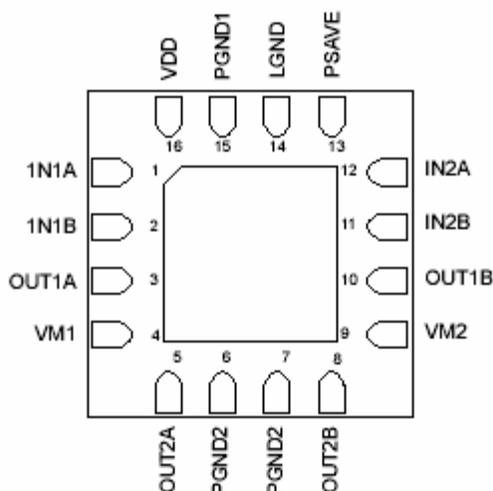


Figure 2. Pin Connections

4.1.4 Pin description

Pin number	Pin name	Pin function	Formal name	Definition
1	IN1A	Logic	Logic Input Control 1A	Logic input control of OUT1A (refer to Table 2)
2	IN1B	Logic	Logic Input Control 1B	Logic input control of OUT1B (refer to Table 2)
3	OUT1A	Output	H-Bridge Output 1A	Output A of H-Bridge channel 1
4	VM1	Power	Motor Driver Power Supply 1	Positive power source connection for H-Bridge 1 (Motor Driver Power Supply)
5	OUT2A	Output	H-Bridge Output 2A	Output A of H-Bridge channel 2
6, 7	PGND2	Ground	Power Ground 2	High-current power ground 2
8	OUT2B	Output	H-Bridge Output 2B	Output B of H-Bridge channel 2
9	VM2	Power	Motor Driver Power Supply 2	Positive power source connection for H-Bridge 2 (Motor Driver Power Supply)
10	OUT1B	Output	H-Bridge Output 1B	Output B of H-Bridge channel 1
11	IN2B	Input	Logic Input Control 2B	Logic input control of OUT2B (refer to Table 2)
12	IN2A	Input	Logic Input Control 2A	Logic input control of OUT2A (refer to Table 2)
13	PSAVE	Input	Input Enable Control	Logic input enable control of H-Bridges to save power
14	LGND	Ground	Logic Ground	Low-current logic signal ground
15	PGND1	Ground	Power Ground 1	High-current power ground 1

Pin number	Pin name	Pin function	Formal name	Definition
16	VDD	Logic	Logic Circuit Power Supply	Positive power source connection for logic circuit

Notes:

- VM1 and VM2 are internally connected (use short PCB traces to connect externally).
- GND, PGND1, and PGND2 are internally connected (same note as VM1 & VM2).

Table 1. 17C724 Pin Definitions

4.1.5 Truth table

Input			Output		VDDDET
PSAVE	IN1A IN2A	IN1B IN2B	OUT1A OUI2A	OUT2A OUT2B	
L	L	L	L	L	Enabled
L	H	L	H	L	Enabled
L	L	H	L	H	Enabled
L	H	H	Z	Z	Enabled
H	X	X	Z	Z	Disabled

Notes:

- Terminal 13 (PSAVE) is pulled up by an internal resistor.
- When VDD is lower than VDDDET while VM is applied, output becomes “Z” (high impedance); however, when PSAVE = “H”, the low-voltage shutdown detection circuit is disabled.

Table 2. Truth Table

H: High

L: Low

Z: High impedance

X: Don't care

4.1.6 Connection diagram

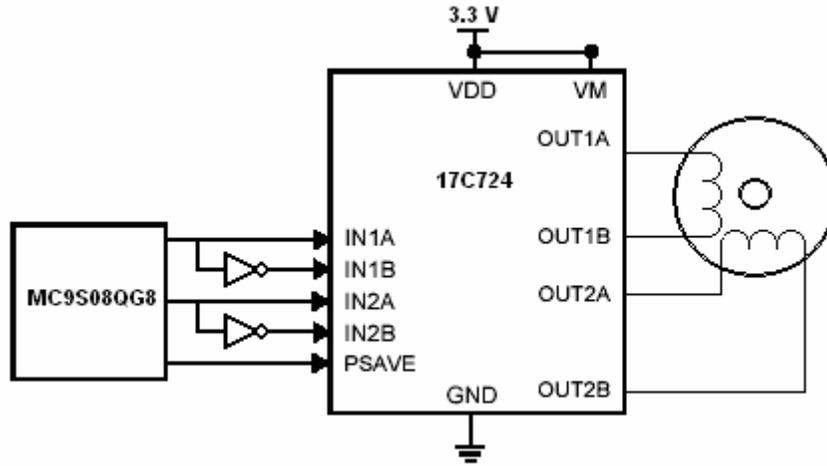


Figure 3. Connection diagram of the MPC17C724

4.2 MC9S08QG8 Overview

4.2.1 Introduction

The MC9S08QG8 is a member of the low-cost, high-performance HCS08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced HCS08 core and are available with a variety of modules, memory sizes, memory types, and package types.

4.2.2 Features

8-Bit HCS08 Central Processor Unit (CPU)

- 20-MHz HCS08 CPU (central processor unit)
- HC08 instruction set with added BGND instruction
- Background debugging system
- Breakpoint capability to allow single breakpoint setting during in-circuit debugging (plus two more breakpoints in on-chip debug module)
- Debug module containing two comparators and nine trigger modes. Eight deep FIFO for storing change-of-flow addresses and event-only data
- Debug module supports both tag and force breakpoints
- Support for up to 32 interrupt/reset sources

Memory options

- FLASH read/program/erase over full operating voltage and temperature
- MC9S08QG8—8 Kbytes FLASH, 512 bytes RAM
- MC9S08QG4—4 Kbytes FLASH, 256 bytes RAM

Input/Output

- 12 general-purpose input/output (I/O) pins, one input-only pin and one output-only pin; outputs 10 mA each, 60 mA max for package
- Software selectable pullups on ports when used as inputs
- Software selectable slew rate control and drive strength on ports when used as outputs
- Internal pullup on the RESET and IRQ pins to reduce customer system cost

4.2.3 External signal description

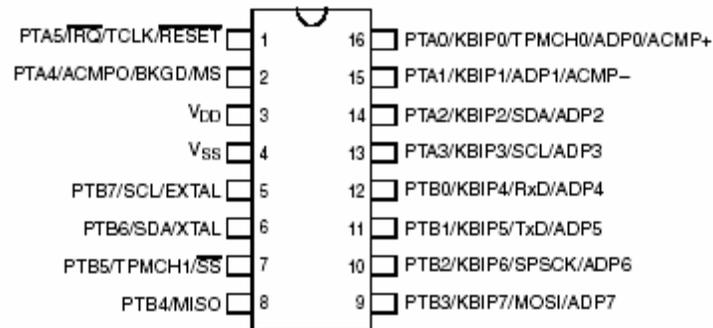


Figure 4. External signal description of the MC9S08QG8

4.3 MC9S08QG8 Demo Board

4.3.1 I/O Port Connector

Figure 5 shows the I/O Port Connector of the MC9S08QG8 Demo Board, and the signals corresponding to each pin.

VDD	1	2	PTA5/RESET/IRQ/TCLK
VSS	3	4	PTA5/RESET/IRQ/TCLK
PTB1/KBI1P5/ADC1P5/TXD1	5	6	PTA4/BKGD/MS/ACMP10
PTB0/KBI1P4/ADC1P4/RXD1	7	8	PTB7/SCL1/EXTAL
PTA2/KBI1P2/ADC1P2/SDA1	9	10	PTB6/SDA1/XTAL
PTA3/KBI1P3/ADC1P3/SCL1	11	12	
PTA5/RESET/IRQ/TCLK	13	14	
PTA0/KBI1P0/ADC1P0/TPM1CH0/AMCP+	15	16	
PTB3/KBI1P7/ADC1P7/MOSI1	17	18	PTA1/KBI1P1/ADC1P1/ACMP1-
PTB4/MISO1	19	20	PTA0/KBI1P0/ADC1P0/TPM1CH0/AMCP+
PTB2/KBI1P6/ADC1P6/SPSCK1	21	22	
PTB5/TPM1CH1/SS1	23	24	
PTA1/KBI1P1/ADC1P1/ACMP1-	25	26	
PTB6/SDA1/XTAL	27	28	
PTB7/SCL1/EXTAL	29	30	
PTA4/BKGD/MS/ACMP10	31	32	

Figure 5. QG8 Demo board I/O port connector diagram

4.3.2 CSTEP

As mentioned before, Cstep is freeware intended to create drawings on your PC, and send them as signals through the PC's parallel port to control plotter pen motors to make the drawing. It can be downloaded from the site www.luberth.com/cstep. This web page also contains information about how to install the software.

The main screen of the program is shown in [Figure 6](#).

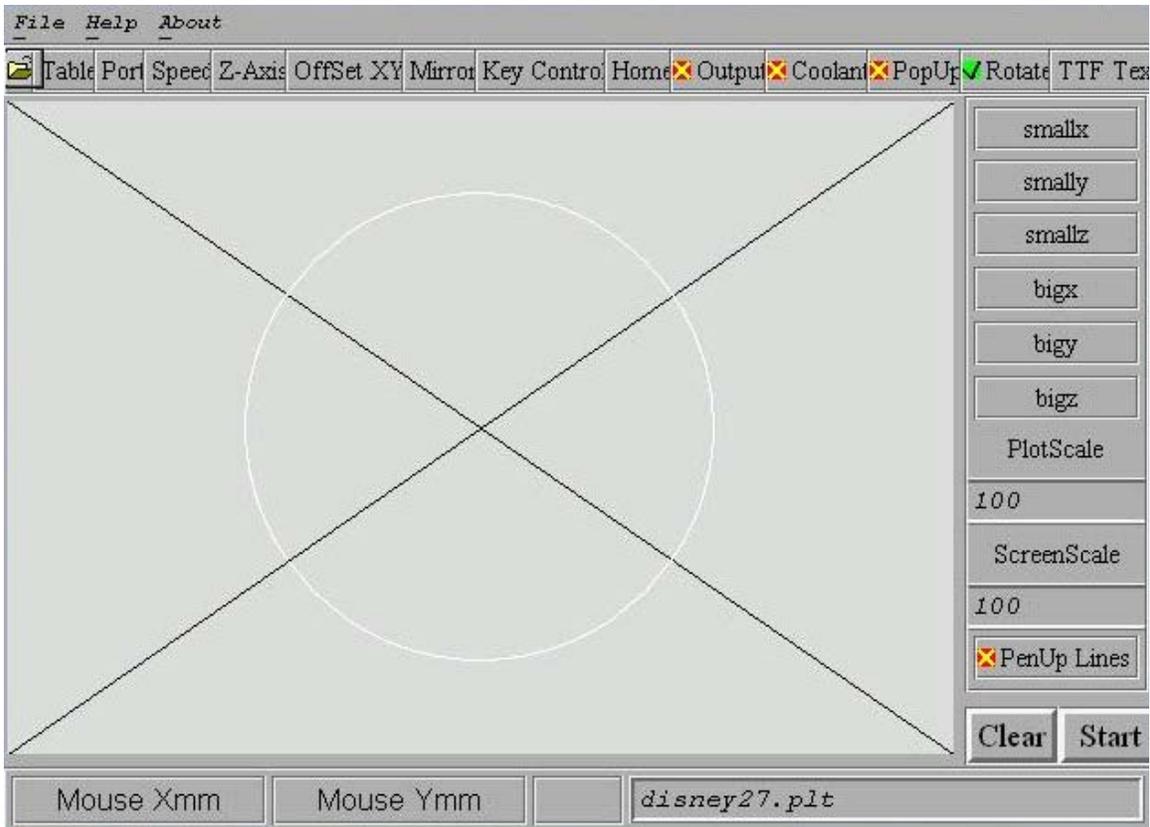


Figure 6. Main screen of Cstep

The right button of the mouse draws a line by clicking it and scrolling the mouse, the left button draws a dot that gives an offset to the plotter to where it is located. After making your drawing and configuring the port outputs on the **Port** menu, just check the **Output** button and click **Start** to send the output signals to the parallel port. The port output screen should be configured as shown in [Figure 7](#).

Output type

Conrad SMC
 Phase Control
 Pulse Dir.

Parallel Port Adres

LPT 278
 LPT 378
 LPT 3BC

Pulse en Dir. Port Settings

Pin 2 D0	<input type="text" value="X-Step"/>			Pin 1	<input type="text"/>
Pin 3 D1	<input type="text" value="X-Direction"/>			Pin 14	<input type="text"/>
Pin 4 D2	<input type="text" value="Y-Step"/>			Pin 31	<input type="text"/>
Pin 5 D3	<input type="text" value="Y-Direction"/>			Pin 36	<input type="text"/>
Pin 6 D4	<input type="text" value="Z-Step"/>			Pin 10	<input type="text"/>
Pin 7 D5	<input type="text" value="Z-Direction"/>			Pin 11	<input type="text"/>
Pin 8 D6	<input type="text"/>			Pin 12	<input type="text"/>
Pin 9 D7	<input type="text"/>			Pin 13	<input type="text"/>
				Pin 32	<input type="text"/>

Low to High
 High to Low

Figure 7. Port output configuration screen

The step signals look like this:



Figure 8. Output step signals

The separation between each step depends on the slope of the line you are drawing.

4.3.3 USING CSTEP IN WINDOWS XP

As Cstep is a program designed to be used in MS-DOS, it is necessary to use another program for it to run well in Windows XP. This program is called UserPort, it's freeware that can be downloaded from the following link: <http://www.embeddedtronics.com/design&ideas.html>

Once the download has finished, unzip the file and copy the files UserPort.exe and UserPort.sys to the directory c:\WINDOWS\system32. Open the Run command on the Start menu and type c:\WINDOWS\system32\UserPort.exe c:\WINDOWS\system32\UserPort.sys. Click OK and the following window will appear.

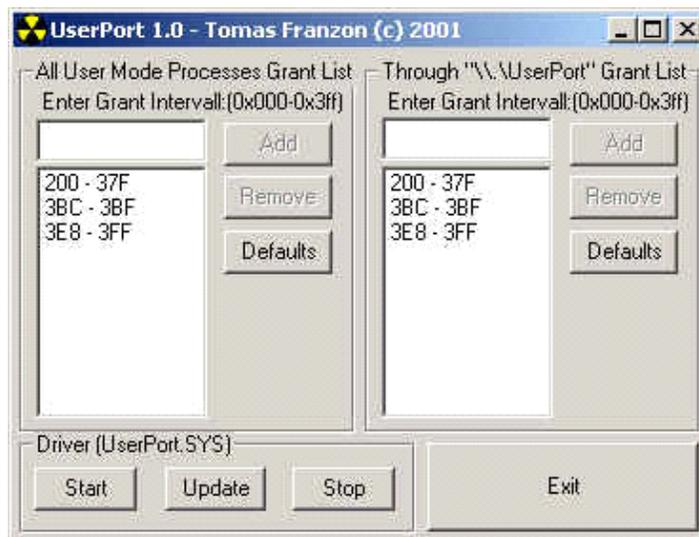


Figure 9. Main screen of UserPort

Remove all directions from the lists and then add the direction 0x378-0x37B to the list on the left.

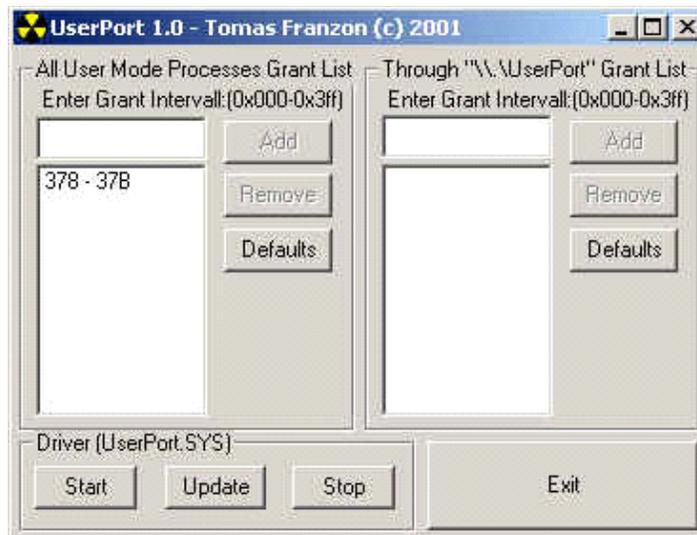


Figure 10. Parallel port direction on the left list

Click Update and Start.

Open Cstep from the Command Prompt (Cstep runs better in DOS mode, not DOS emulation), it must be in the directory c:\Mgui\Luberth\. Begin using it.

4.3.4 STEPPER MOTOR CONTROL WITH MPC17C724

The stepper motors used consist of two coils, and are controlled with the MC9S08QG8, that translates the parallel port signals shown in [Figure 8](#) according to the following graph.

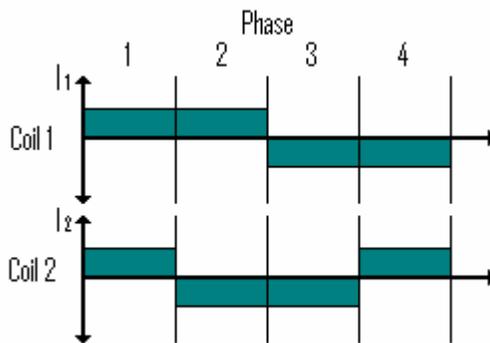


Figure 11. Motor coil current

[Figure 11](#) shows the current on the coils of the motor on each phase. This mode of control produces greater torque but less resolution on the motor steps.

So according to [Figure 3](#) and [Table 2](#), the MC9S08QG8 interrupts on each pulse of the parallel port and toggles the corresponding output port to produce a step on the motor.

4.4 The 17C724 EVM

Figure 12 shows the board of the 'QG8 development kit.

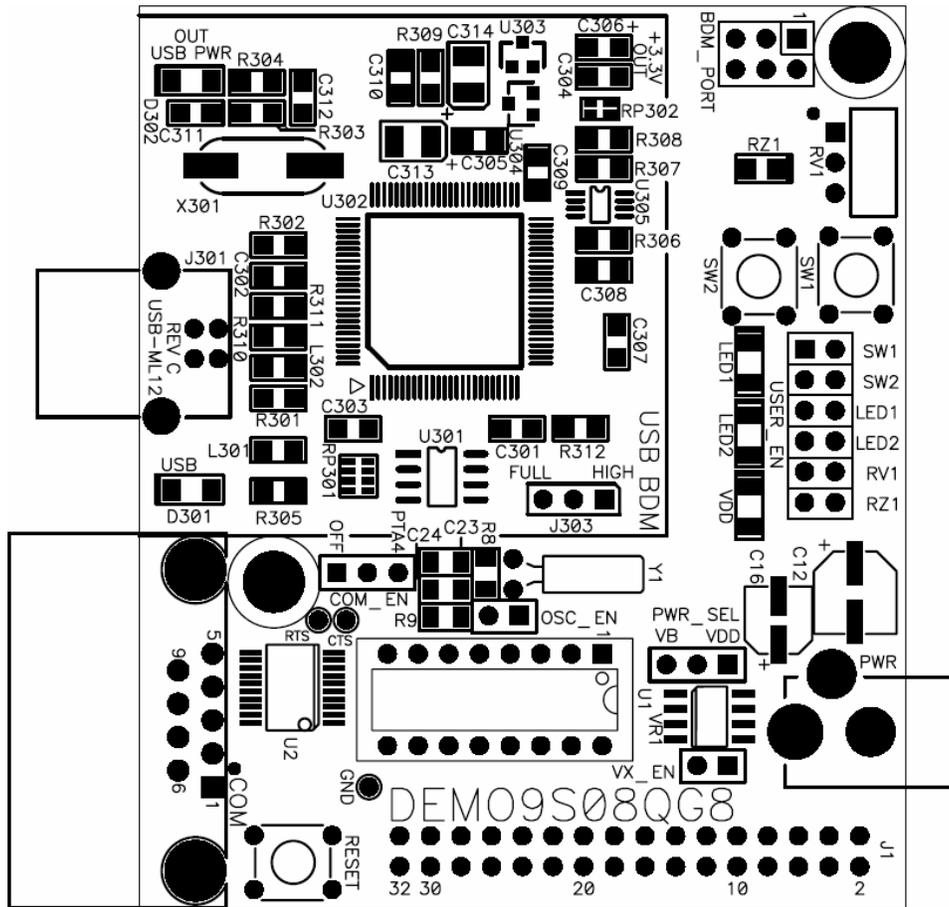


Figure 12. QG8 development kit board

Figure 13 shows the way jumpers must be connected to program the MC9S08QG8 when supplying the board via an USB cable. Once programmed, the USB cable must be disconnected to avoid noise on the background pin, and the board must be supplied by the PWR connector. To do this, change the PWR_SEL jumper, as shown in Figure 14.

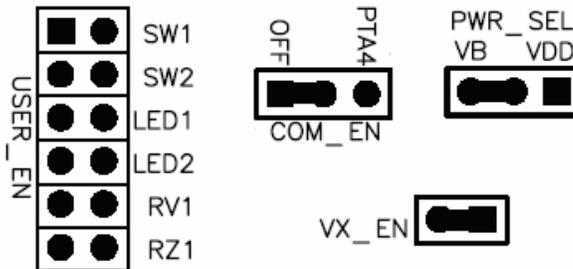


Figure 13. Jumpers in the 'QG8 board

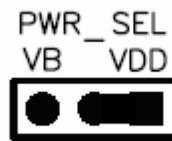


Figure 14. PWR_SEL jumper connection for PWR connector supplying

Figure 15 and Figure 16 show the schematic and the board of the MPC17C724 EVM respectively.

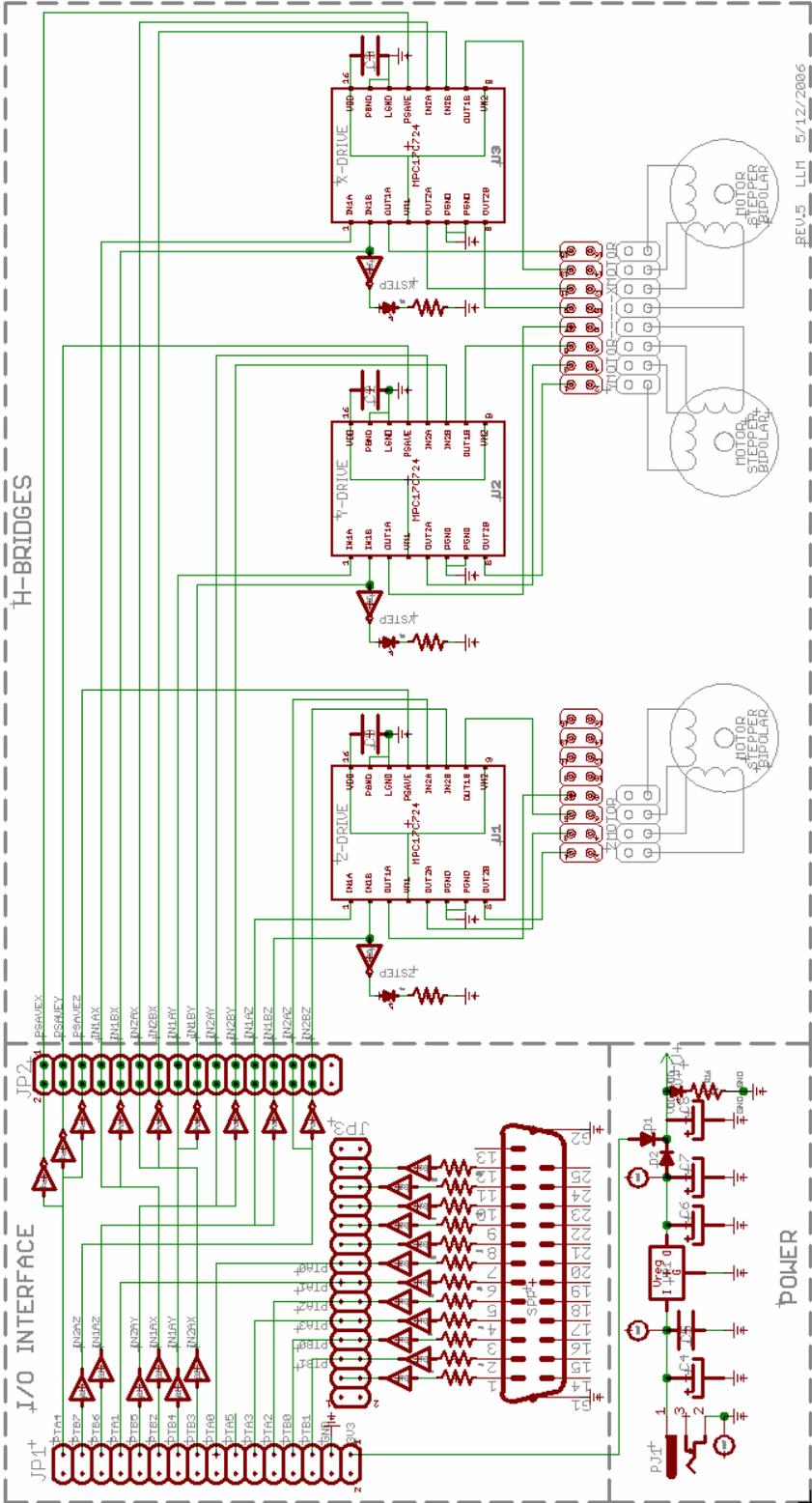


Figure 15. Schematic of the 17C724 EVM

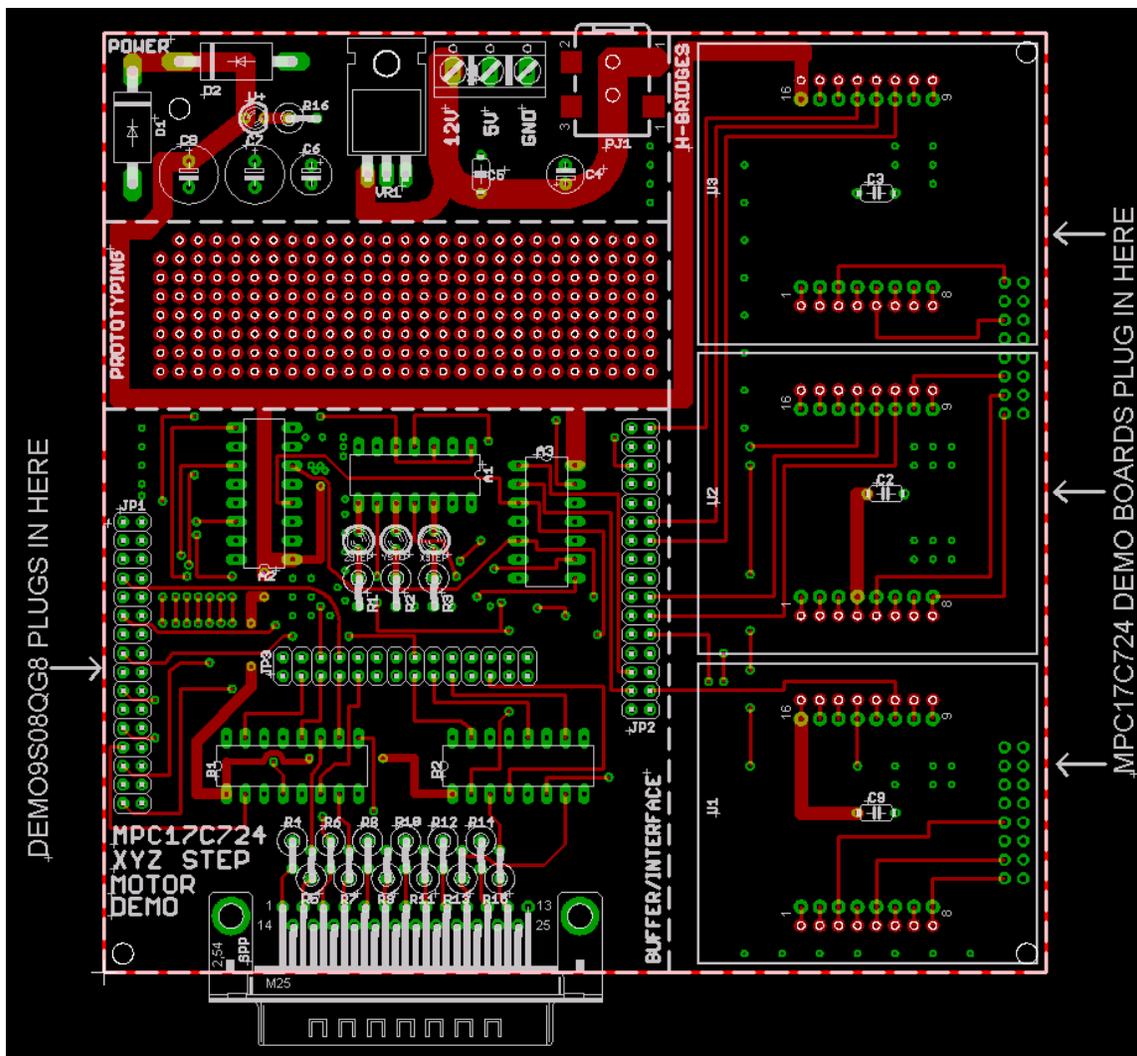


Figure 16. 17C724 EVM board

Figure 16 shows also where the MPC17C724 boards and the ‘QG8 development kit must be connected. Header JP1 must be connected to J1, on the ‘QG8 board, noting the pin 1 mark on top on the header. The MPC17C724 boards have similar pin 1 marks for its connectors.

The power supply of the board can be through connectors X1 or PJ1, with a 12 V source as indicated on the board.

Finally, to start using the MPC17C724 EVM, just program the ‘QG8, open Cstep, plug the parallel port cable to SPP connector, and connect the motors to the motor headers as indicated in Figure 17.

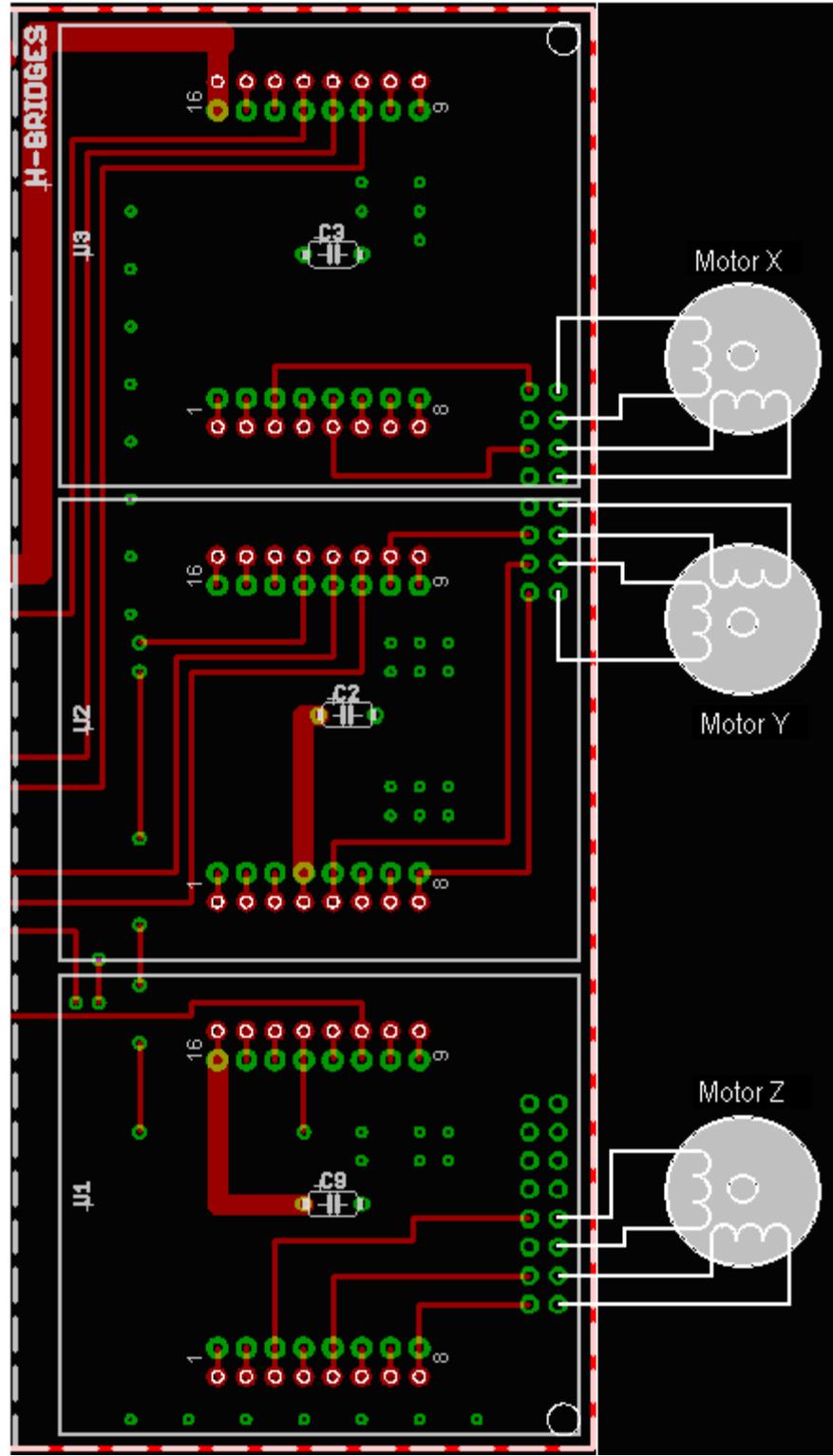


Figure 17. Motor connection

4.5 Source Code

The following part of the code must be in a file named “definitions.h” in the sources folder of the project.

```

/* Watchdog and Background pin configuration */
#define Disable_Watchdog() {SOPT1 = 0x01;}

/* MTim Configuration */
#define Configure_Timer() {MTIMCLK_PS = 8; MTIMCLK_CLKS = 0; MTIMMOD = 0; MTIMSC = 0x50;}
#define Reset_Timer() {MTIMSC_TRST = 1;}
#define Stop_Timer() {MTIMSC_TSTP = 1;}
#define Start_Timer() {MTIMSC = 0x60;}
#define Clear_Timer_Flag() {MTIMSC_TOF = 0;}

/* Keyboard interrupt definitions */
#define Configure_Keyboard_Interrupt() {KBISC_KBIMOD = 0; KBIES = 0x2A; KBIPE = 0x2A;}
#define Enable_Keyboard_Interrupt() {KBISC_KBIE = 1;}
#define ACKNOWLEDGE KBISC_KBACK

/* Motor X port definitions */
#define XSTEP PTBD_PTBD1
#define XSTEP_DD PTBDD_PTBD1
#define XDIR PTBD_PTBD0
#define XDIR_DD PTBDD_PTBD0
#define XSTEP_INT KBIPE_KBIPE5
#define XSTEP_PULLUP PTBPE_PTBP1
#define IN1X PTBD_PTBD2
#define IN1XDD PTBDD_PTBD2
#define IN2X PTBD_PTBD3
#define IN2XDD PTBDD_PTBD3

/*Motor Y port definitions */
#define YSTEP PTAD_PTAD3
#define YSTEP_DD PTADD_PTADD3
#define YDIR PTAD_PTAD2
#define YDIR_DD PTADD_PTADD2
#define YSTEP_INT KBIPE_KBIPE3
#define YSTEP_PULLUP PTAPE_PTAPE3
#define IN1Y PTBD_PTBD4
#define IN1YDD PTBDD_PTBD4
#define IN2Y PTBD_PTBD5
#define IN2YDD PTBDD_PTBD5

```

```

/*Motor Z port definitions */
#define ZSTEP PTAD_PTAD1
#define ZSTEP_DD PTADD_PTADD1
#define ZDIR PTAD_PTAD0
#define ZDIR_DD PTADD_PTADD0
#define ZSTEP_INT KBIPE_KBIPE1
#define ZSTEP_PULLUP PTAPE_PTAPE1
#define IN1Z PTBD_PTBD6
#define IN1ZDD PTBDD_PTBDD6
#define IN2Z PTBD_PTBD7
#define IN2ZDD PTBDD_PTBDD7

/* PSAVE port definition */
#define PSAVE PTAD_PTAD4

```

Now this part of the code must be in the main.c file of the project.

```

/**
 * Copyright (c) 2006, Freescale Semiconductor
 * Freescale Quick Reference
 *
 * File name : main.c
 * Project name: MPC17C724 EVM.
 *
 * Author : Jorge Ramírez Rivero.
 * Department : RTAC Americas.
 *
 * Description : QG8 program to control stepper motors with Cstep.
 *
 * History :
 * 06/28/2006 : First version finished. (B01882)
 * 07/17/2006 : Second version finished. (B01882)
 **/

#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */
#include "definitions.h"

unsigned char countx = 0, county = 0, countz = 0;
unsigned int timer_count = 0, dirx = 0, diry = 0, dirz = 0;
void main(void) {
Disable_Watchdog(); /* Disable background debug mode pin and */

```

Overview

```

/* watchdog */
EnableInterrupts; /* Enable interrupts */
/* Configure I/O ports */
/* X Control ports */
XSTEP = 0;
XSTEP_DD = 0;
XDIR = 0;
XDIR_DD = 0; /* XSTEP and XDIR are set as inputs */
IN1X = 0;
IN1XDD = 1;
IN2X = 0; /* IN1 and 2 of the motors are */
/* initially low */
IN2XDD = 1; /* IN1 and 2 are set as outputs */
/* Y Control ports */
YSTEP = 0;
YSTEP_DD = 0;
YDIR = 0;
YDIR_DD = 0;
IN1Y = 0;
IN1YDD = 1;
IN2Y = 0;
IN2YDD = 1;
/* Z Control ports */
ZSTEP = 0;
ZSTEP_DD = 0;
ZDIR = 0;
ZDIR_DD = 0;
IN1Z = 0;
IN1ZDD = 1;
IN2Z = 0;
IN2ZDD = 1;
/* PSAVE Initialization */
PSAVE = 0; /* Power save enabled */
/* MTim_setup */
Configure_Timer(); /* If there is no activity for a few */
/* seconds PSAVE activates */
/* Configure and enable KBI (Keyboard Interrupt) */
Configure_Keyboard_Interrupt();
ACKNOWLEDGE = 1; /* Clear Pending Keyboard Interrupts */
Enable_Keyboard_Interrupt(); /* Enable Keyboard Interrupts */
for (;;) {
} /* loop forever */

```

```

}
interrupt 18 void KeyBoardIsr(void) {
    unsigned int temp;
    Reset_Timer();
    Stop_Timer();
    PSAVE = 1; /* Power save disabled */
    if (XSTEP == 1) { /* Checks if the interruption was caused */
        /* by motor X control pin */
        if (dirx != XDIR) { /* Checks if there is a change in */
            /* the direction of the motor */
                countx = ~countx; /* Toggling count changes the direction */
            /* of the motor */
            dirx = XDIR;
        }
        if (countx == 0) { /* Count checks which port is to be */
            /* toggled */
                IN1X = ~IN1X;
                countx = 0xFF;
            }
        else if (countx == 0xFF) {
            IN2X = ~IN2X; /* Toggle ports */
            countx = 0;
        }
    }
    if (YSTEP == 1) { /* Checks if the interruption was caused */
        /* by motor Y control pin */
        if (diry != YDIR) {
            county = ~county;
            diry = YDIR;
        }
        if (county == 0) {
            IN1Y = ~IN1Y;
            county = 0xFF;
        }
        else if (county == 0xFF) {
            IN2Y = ~IN2Y;
            county = 0;
        }
    }
    if (ZSTEP == 1) { /* Checks if the interruption was caused */
        /* by motor Z control pin */
        if (dirz != ZDIR) {
    
```

Overview

```

        countz = ~countz;
        dirz = ZDIR;
    }
    if (countz == 0) {
        IN1Z = ~IN1Z;
        countz = 0xFF;
    }
    else if (countz == 0xFF) {
        IN2Z = ~IN2Z;
        countz = 0;
    }
}
for (temp = 0; temp <= 0x1FFE; temp++) {
}
/* Debouncer cycle */
ACKNOWLEDGE = 1;
/* Clear Pending Keyboard Interrupts */
/* Clear Keyboard Interrupt Flag */
Start_Timer();
/* Start timer */
}
interrupt 12 void MTimISR(void) {
    Clear_Timer_Flag();
    if (timer_count <= 100) {
        timer_count ++;
    }
    else {
        Stop_Timer();
        timer_count = 0;
        PSAVE = 0;
        /* Activate PSAVE */
    }
}

```

4.5.1 PROGRAMMING THE ‘QG8

To program the ‘QG8, the jumpers on the board must be connected as [Figure 12](#) shows.

Freescale CodeWarrior V3.1 (or newer) is needed. Once you have installed and registered it, follow the next steps to program it with the code provided above. These steps are based on version 5.0 of the CodeWarrior.

First, create a new project by selecting **New** from the **File** menu or by selecting **Create New Project** from the Startup menu (if it is configured to be displayed on startup). A window will appear.

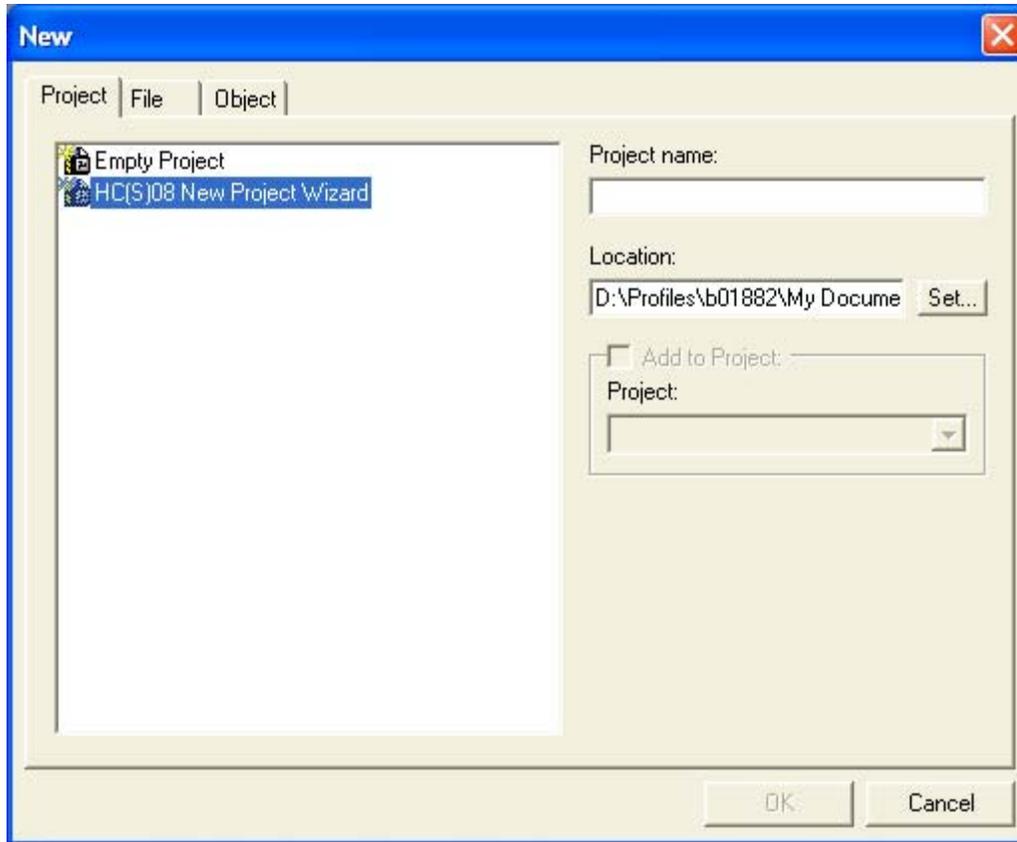


Figure 18. New project window

Note: If you selected the option from the startup menu, this window will not be shown, but the wizard will open automatically.

Select **HC(S)08 New Project Wizard**, name the project and set its location by clicking **Set**. Then click **OK**.

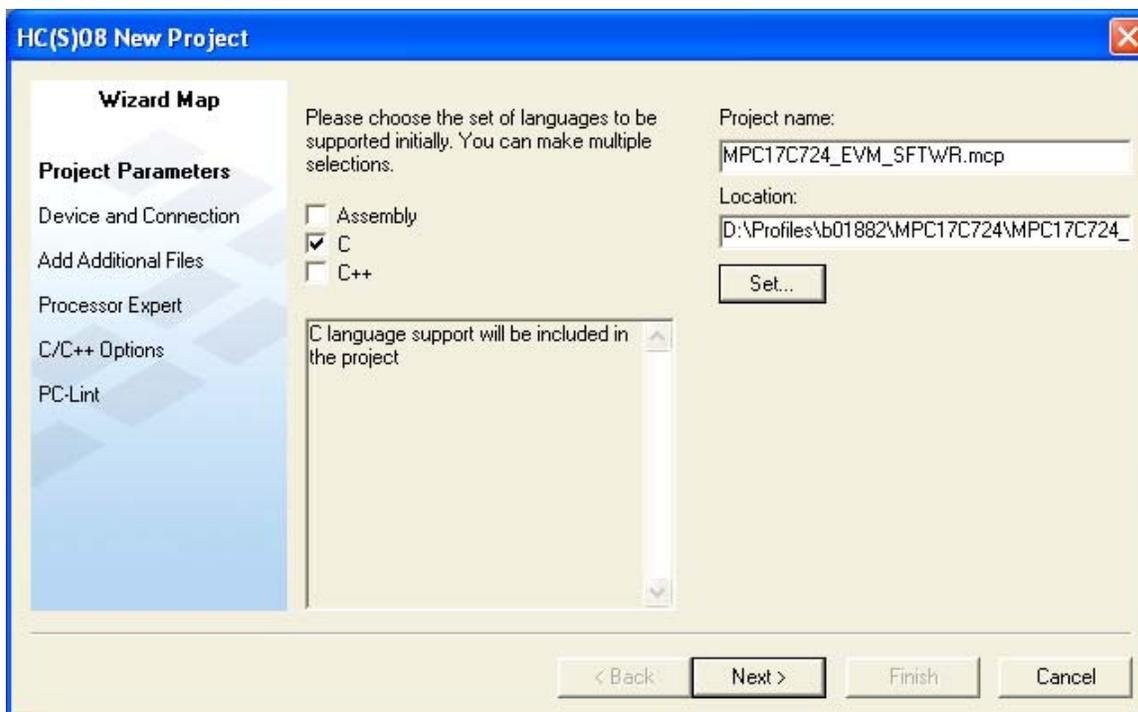


Figure 19. New project wizard window

The next screen is the Language Support. Select **C** and click next.

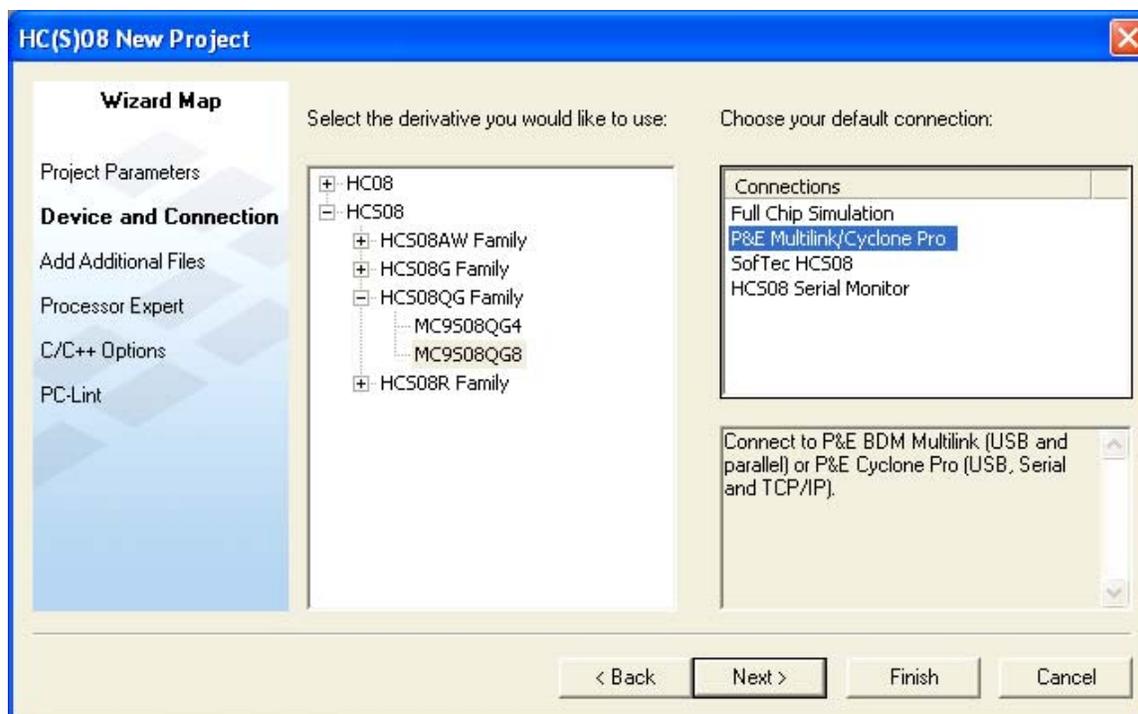


Figure 20. New project wizard window

Now select the **MC9S08QG8** from the **HCS08** family, and **P&E Multilink/Cyclone Pro** as your default connection.

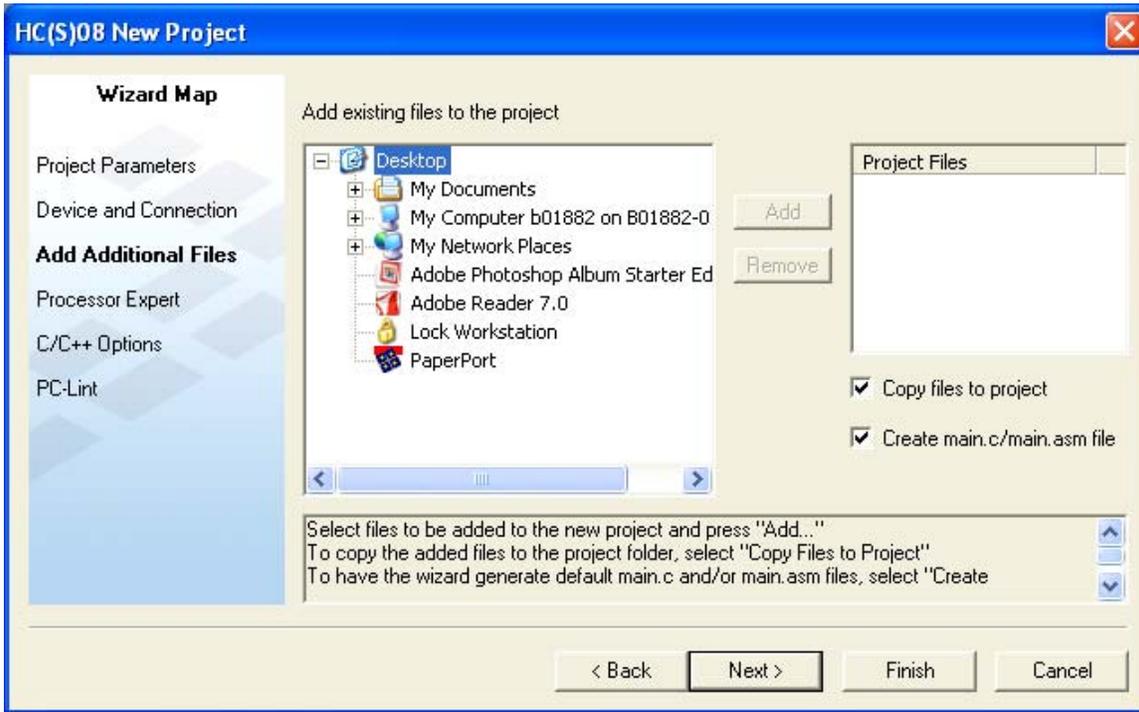


Figure 21. New project wizard window

No existing files are needed on the project. Click Next on this screen.

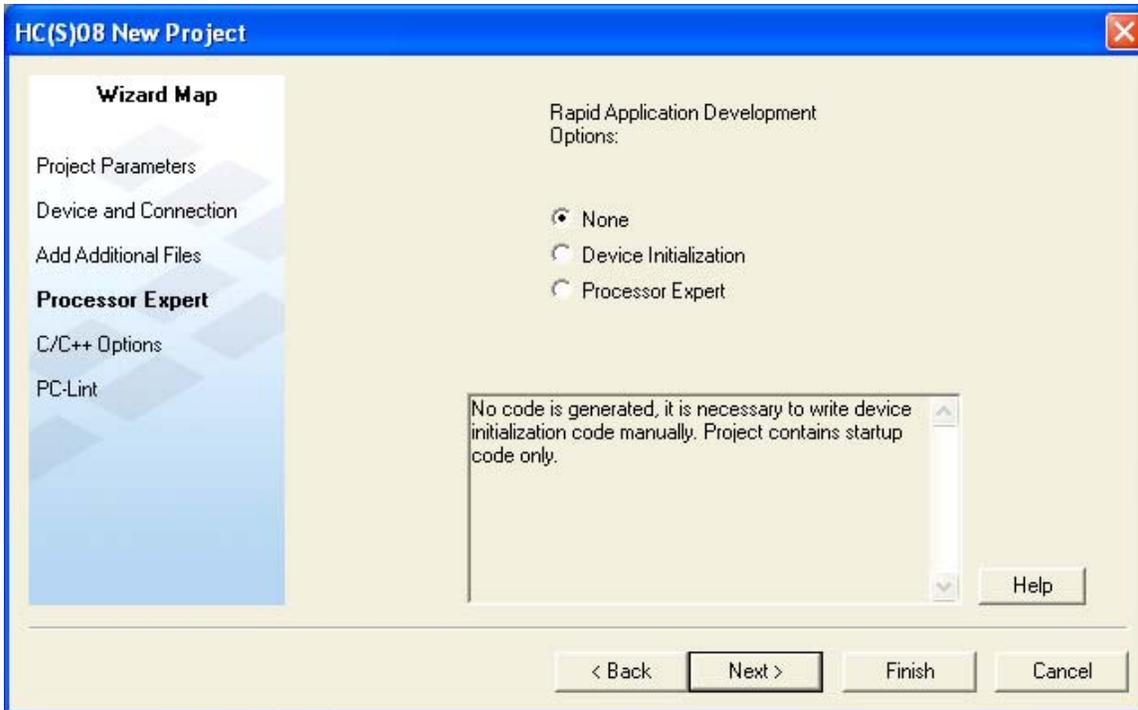


Figure 22. New project wizard window

Select **None** on the Rapid Application Development Options.

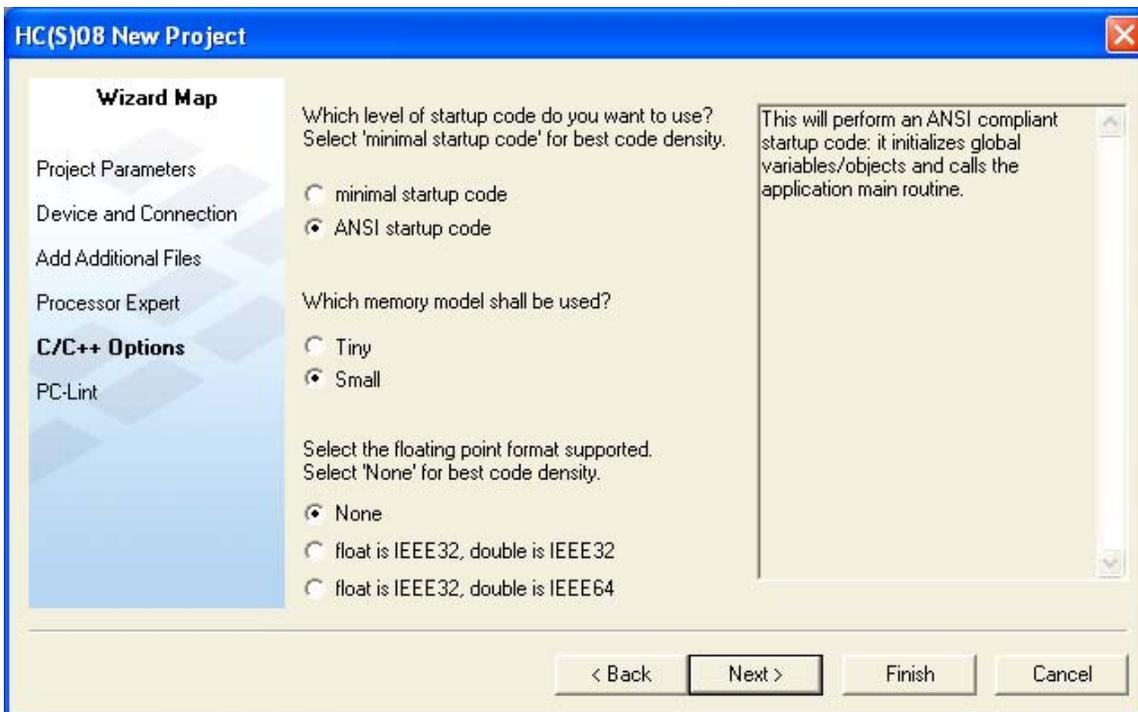


Figure 23. New project wizard window

Select **ANSI startup code** as the level of startup code, **Small** as the memory model, and **None** of the floating point formats supported.

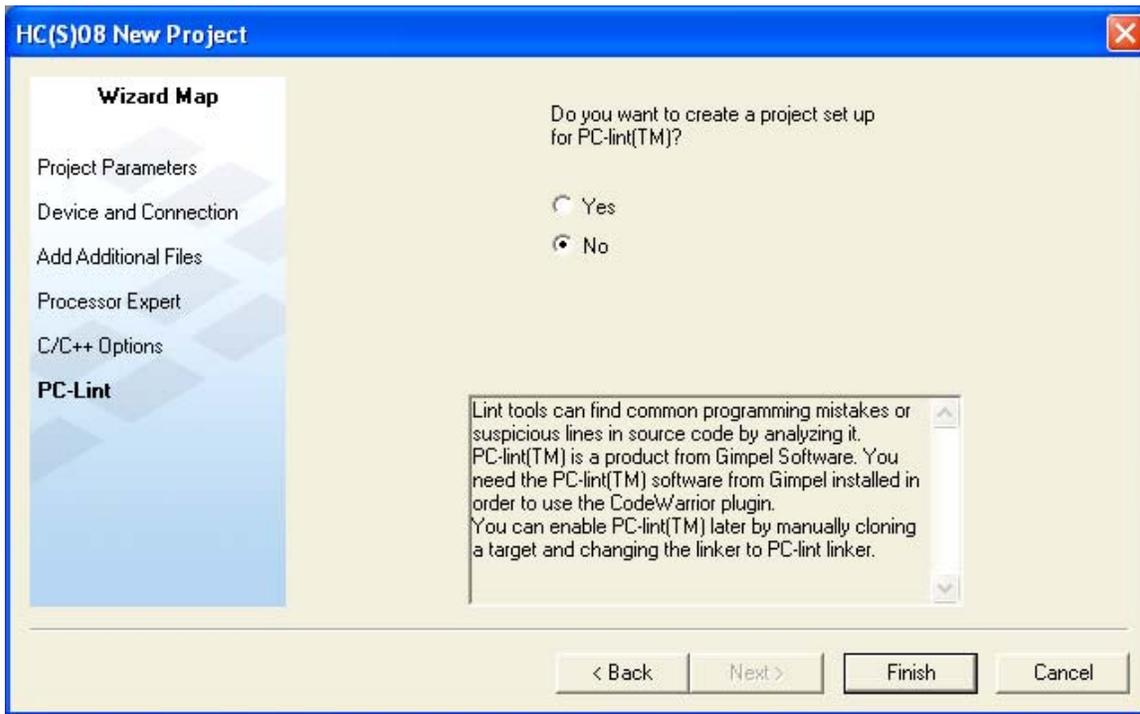


Figure 24. New project wizard window

PC-lint option will appear. Select **No** and click **Finish**.

Once the project wizard has finished, a project including a skeleton application will be created. The skeleton will look like [Figure 25](#).

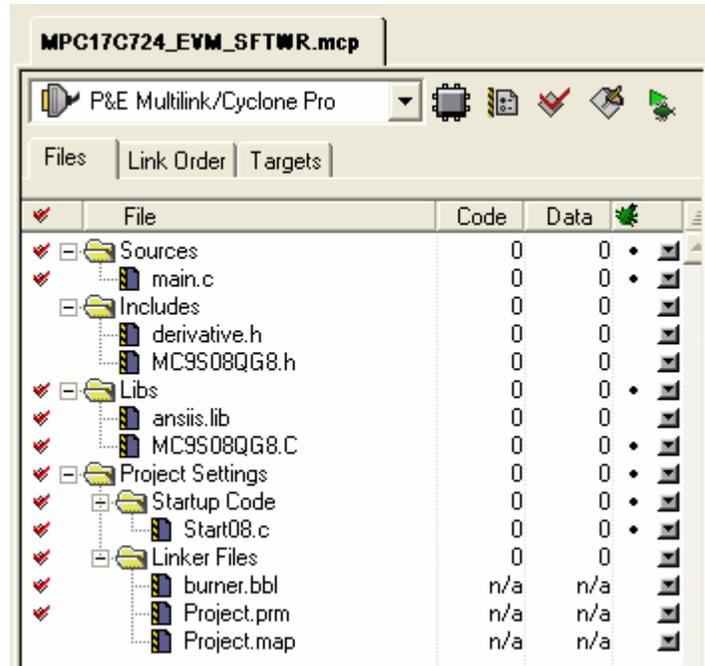


Figure 25. New project wizard window

Double click on the **main.c** icon on the sources folder, delete the code that appears, copy the **main.c** file code provided above and paste it on the **main.c** file.

Click **New Text File** on the **File** menu, and a new blank file will appear. Copy the code of the **“definitions.h”** file and paste it there. Now save the file under the name of **“definitions.h”** on the **Sources** folder of the project. Remember to save the whole project too.

Now your project is ready to be loaded on the ‘QG8. Click on the **Debug Green Arrow** on top of the skeleton window. Make sure that the default connection is set as **P&E Multilink/Cyclone Pro**, as shown on [Figure 25](#), and your ‘QG8 board is plugged into the USB port of your computer.

A new window will open.

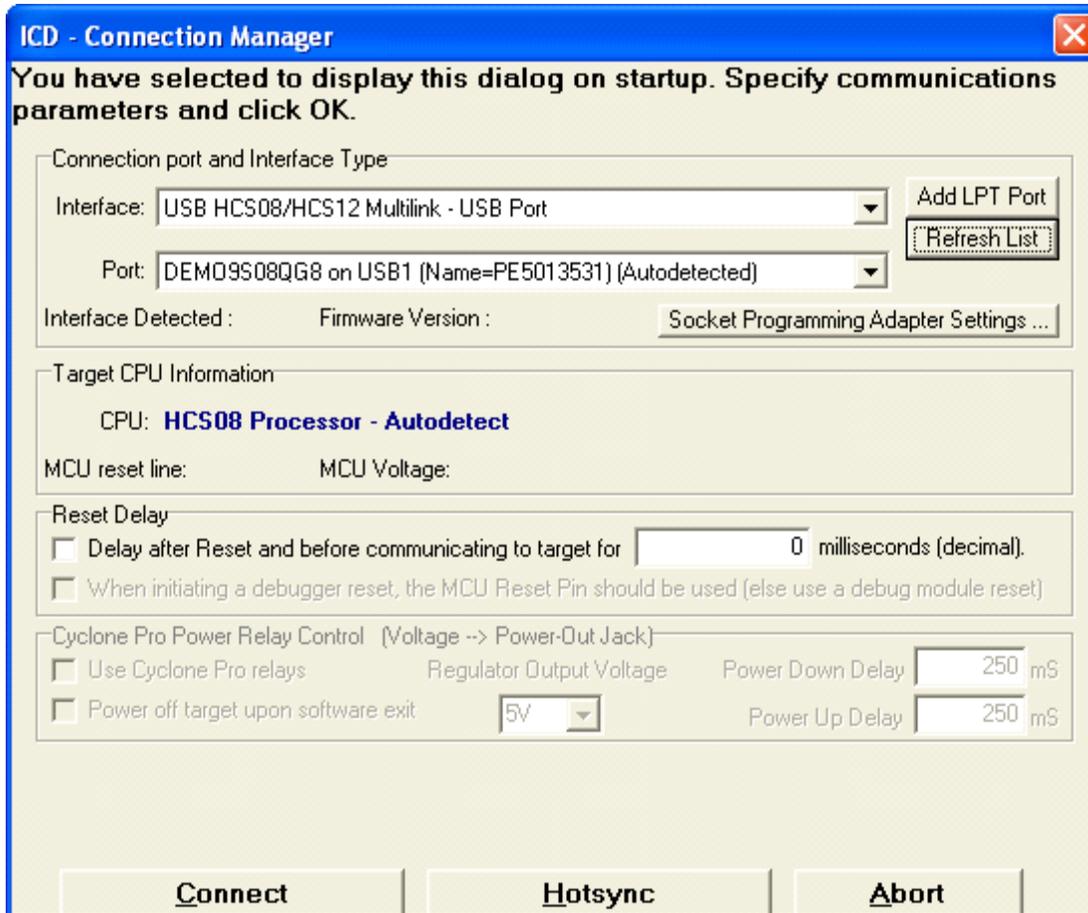


Figure 26. Connection Manager

The 'QG8 board connection will be auto detected. If not, make sure the board is well connected and installed, and then click **Refresh List**. Once the board is detected, click **Connect**.

A dialog box will appear.



Figure 27. Program Flash dialogue box

Click **Yes**.

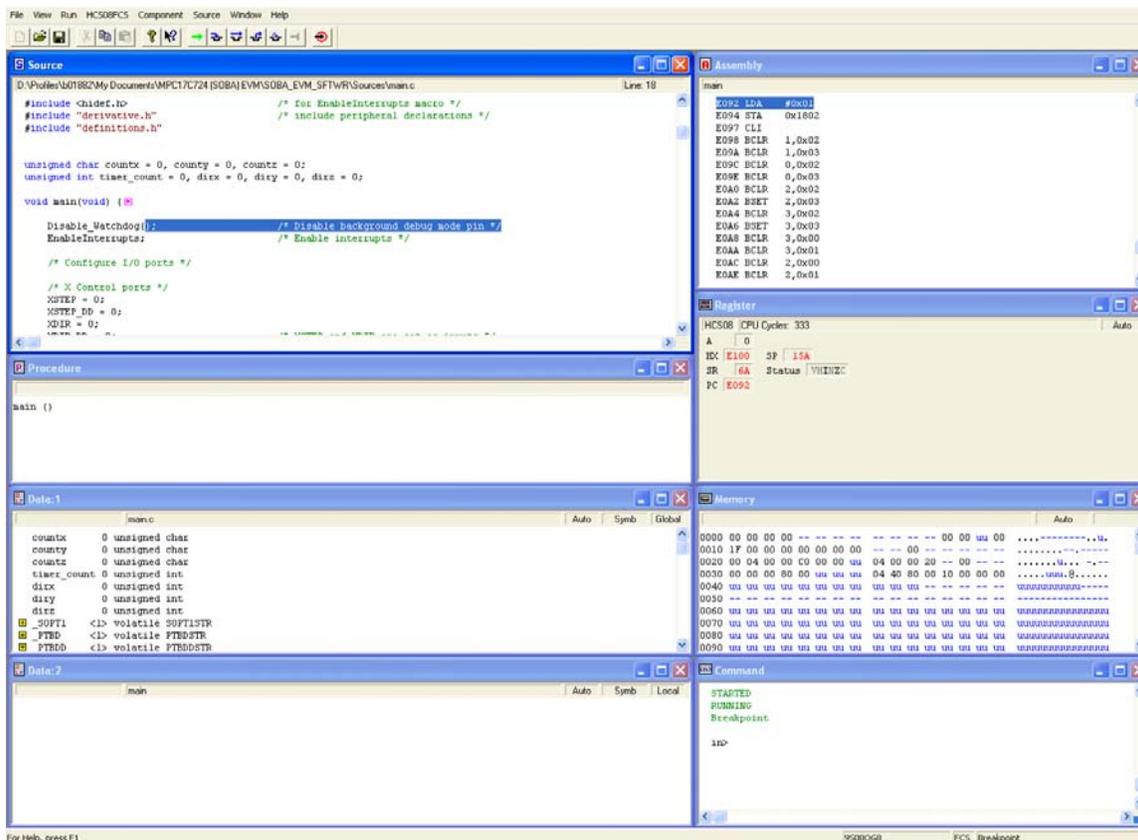


Figure 28. True Time Simulator

Finally, click on the **Start Green Arrow** on top of this screen (Figure 28), or hit **F5**, and the QG8 will be programmed.

To stop the program, click on the **Halt Red Button**, or hit **F6**.

Remember that once programmed, the PWR_SEL jumper on the QG8 board must be changed as shown in Figure 12, and the board must be supplied via the PWR connector for the hardware to work properly.

5 References

- MPC17C724 Datasheet
- MC9S08QG8 Datasheet
- MC9S08QG8 Application notes
- MC9S08QG8 Reference manual
- MC9S08QG8 Users Guides
- www.luberth.com/cstep
- www.writelog.com/support/lpt_port_support_on_windows_nt.htm

How to Reach Us:

Home Page:

www.freescale.com

E-mail:

support@freescale.com

USA/Europe or Locations Not Listed:

Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc., 2006. All rights reserved.