MOTOROLA
intelligence everywhere™

digital dna™

Accelerometer Reference Design Using the MC68HC908QY4

Designer Reference Manual

M68HC08
Microcontrollers

DRM054/D
Rev. 0
12/2003

MOTOROLA.COM/SEMICONDUCTORS

Accelerometer Reference Design Using the MC68HC908QY4

Reference Design

By: Rogelio González
Rebeca Delgado
Motorola Semiconductor Products Sector
Mexico

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:
http://motorola.com/semiconductors

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

Revision History

Revision History

Date	Revision Level	Description	Page Number(s)
December, 2003	N/A	Initial release	N/A

Table of Contents

Table of Contents

Section 5. Software Design Considerations

Section 6. Source Code

List of Figures and Tables

List of Figures and Tables

Section 1. Introduction and Setup

1.1 Introduction

Motorola's Low-Cost Accelerometer Reference Design was developed to demonstrate the capabilities of the accelerometers (MMA1220D and MMA1260D) in conjunction with the MC68HC908QY4 microcontroller unit (MCU). The demo board consists of hardware and software for this purpose.

Hardware consists of:

- 8-bit MCU
- Z-axis sensitivity accelerometers
- User interface: 16 x 2 character display and two push buttons
- On-board 5-Vdc power supply
- RS-232 interface and MON08 hardware interface for external microcontroller communication and for in-application programming

NOTE: *The board is powered by a 9-Vdc power supply*

The software consists of:

- Self-test verification on both accelerometers
- Acceleration display for the accelerometer's readings
- User interface control
- Over-acceleration warning

NOTE: *The application is fully written in C language for CodeWarrior HC08 V2.1.*

The 4K MCU FLASH memory can be reprogrammed through the user mode monitor program. However, the user monitor program itself can be programmed or erased on the board through the MON08 hardware interface.

1.2 MC68HC908QY4 Microcontroller Unit

The Motorola M68HC08 Family of 8-bit MCUs brings designers of consumer electronics, industrial, and automotive systems increased flexibility in the development and manufacturing processes, as well as reduced time-to-market and system cost. This development is based on a MC68HC908QY4 MCU, member of the series.

The M68HC908QY Family helps make it simple to incorporate the benefits of FLASH technology into your designs, helping to reduce overall system costs and speed your time-to-market.

Motorola's FLASH MCUs are in-application and in-circuit re-programmable, with very fast programming times (as fast as 32 microseconds/byte), block protection, and security features to help customers guard intellectual property contained in software code. They allow embedded system designers to program late in the manufacturing cycle, make upgrades remotely in the field, and to respond quickly to the changing needs of their customers and the market with more flexibility than one-time programmable and ROM-based MCUs.

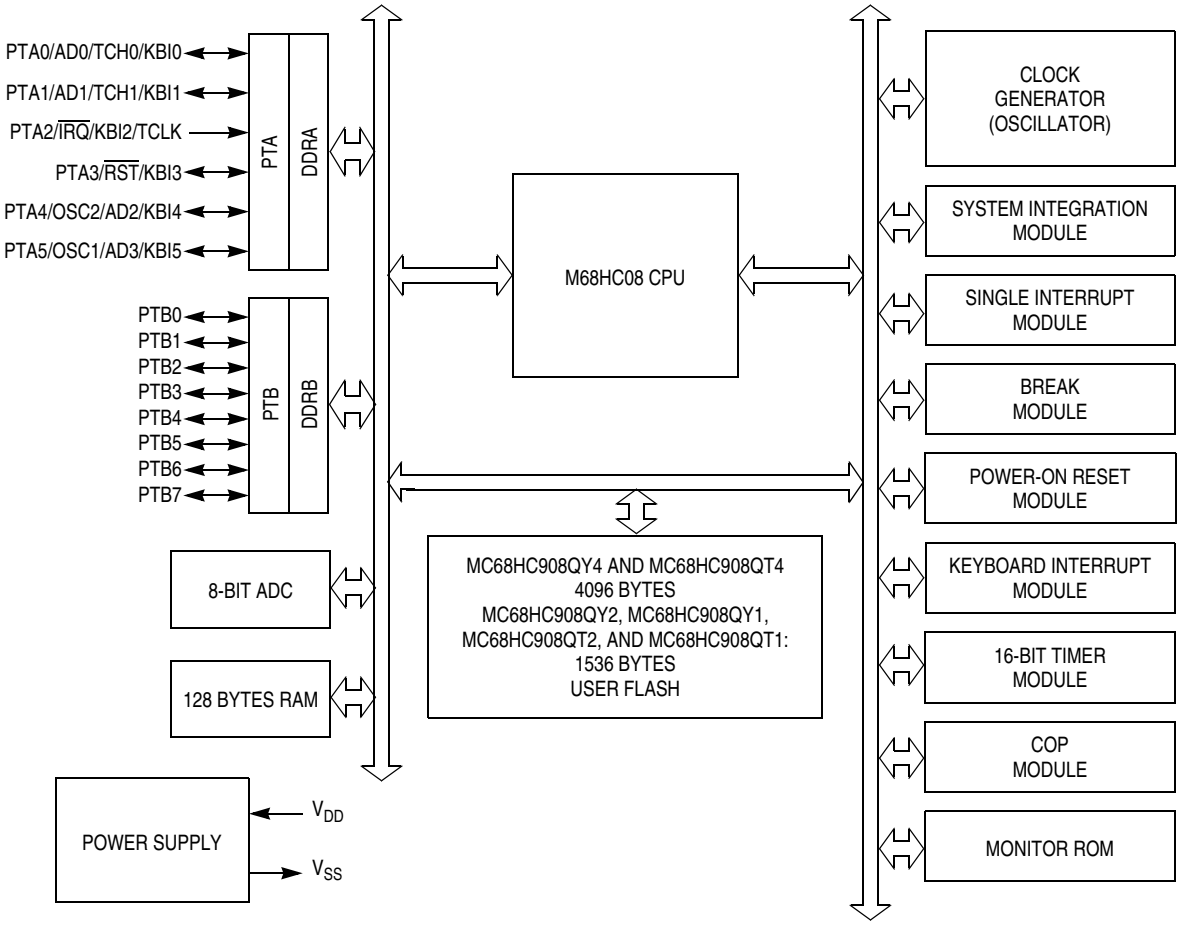
A block diagram of the MC68HC908QY4 is shown in **Figure 1-1**.

In addition, a variety of integrated peripherals make these MCUs versatile enough for a wide range of systems — from computer peripherals to automotive electronics. This family is ideally suited for applications such as discrete replacement, appliances, control systems, home and industrial security systems, fluorescent light ballasts, and electromechanical replacement. The Motorola M68HC908QY M68HC08 Family of 8-bit MCUs will impact consumer, automotive, and industrial products ranging from light dimmer switches to washing machines, all the way into very cost-critical applications.

The M68HC08 Family is a Complex Instruction Set Computer (CISC) with a Von Neumann architecture. All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08). The M68HC908QY Family offers a variety of low-pin-count, low-cost package options available with different configurations of modules, memory sizes, and types.

Table 1-1. MC68HC908QY4 Peripherals and Memory

Internal RAM (Bytes)	FLASH (Bytes)	Timer Interface Module		I/O Pins (Max)	A/D Converter Module		Pin Count	Supply Voltage (V)	Max Bus Frequency (MHz)	PWM Module	
		Channels	Bits		Channels	Bits				Channels	Bits
128	4096	2	16	14	4, 2	4, 8	16	5, 3	8, 3	2, 8	16



RST, IRQ: Pins have internal (about 30K Ohms) pull up
PTA[0:5]: High current sink and source capability
PTA[0:5]: Pins have programmable keyboard interrupt and pull up
PTB[0:7]: Not available on 8-pin devices – MC68HC908QT1, MC68HC908QT2, and MC68HC908QT4
ADC: Not available on the MC68HC908QY1 and MC68HC908QT1

Figure 1-1. MC68HC908QY4 Block Diagram

Features of the MC6868HC908QY4 include:

- High-performance M68HC08 CPU core
- Fully upward-compatible object code with M68HC05 Family
- 5-V and 3-V operating voltages (V_{DD})
- 8-MHz internal bus operation at 5 V, 4-MHz at 3 V
- Trimmable internal oscillator
  - 3.2 MHz internal bus operation
  - 8-bit trim capability
  - ± 25% untrimmed
  - ± 5% trimmed

- Auto wakeup from STOP capability
- Configuration (CONFIG) register for MCU configuration options, including:
  - Low-voltage inhibit (LVI) trip point
- In-system FLASH programming
- FLASH security(1)
- 4096 bytes of on-chip in-application programmable FLASH memory (with internal program/erase voltage generation)
- 128 bytes of on-chip random-access memory (RAM)
- 2-channel, 16-bit timer interface module (TIM)
- 4-channel, 8-bit analog-to-digital converter (ADC)
- 5 or 13 bidirectional input/output (I/O) lines and one input only:
  - Six shared with keyboard interrupt function and ADC
  - Two shared with timer channels
  - One shared with external interrupt (IRQ)
  - Eight extra I/O lines on 16-pin package only
  - High current sink/source capability on all port pins
  - Selectable pullups on all ports, selectable on an individual bit basis
  - Three-state ability on all port pins
- 6-bit keyboard interrupt with wakeup feature (KBI)
- Low-voltage inhibit (LVI) module features:
  - Software selectable trip point in CONFIG register
- System protection features:
  - Computer operating properly (COP) watchdog
  - Low-voltage detection with reset
  - Illegal opcode detection with reset
  - Illegal address detection with reset
- External asynchronous interrupt pin with internal pullup ($\overline{\text{IRQ}}$) shared with general-purpose input pin
- Master asynchronous reset pin ($\overline{\text{RST}}$) shared with general-purpose input/output (I/O) pin
- Power-on reset
- Internal pullups on $\overline{\text{IRQ}}$ and $\overline{\text{RST}}$ to reduce external components
- Memory mapped I/O registers

1. No security feature is absolutely secure. However, Motorola's strategy is to make reading or copying the FLASH difficult for unauthorized users.

- Power saving stop and wait modes
- Available in these packages:
  - 16-pin plastic dual in-line package (PDIP)
  - 16-pin small outline integrated circuit (SOIC) package
  - 16-pin thin shrink small outline package (TSSOP)

Features of the CPU08 include the following:

- Enhanced HC05 programming model
- Extensive loop control functions
- 16 addressing modes (eight more than the HC05)
- 16-bit index register and stack pointer
- Memory-to-memory data transfers
- Fast 8 × 8 multiply instruction
- Fast 16/8 divide instruction
- Binary-coded decimal (BCD) instructions
- Optimization for controller applications
- Efficient C language support

1.3 MMA1220 and MMA1260 Accelerometers

The MMA series of silicon capacitive, micro-machined accelerometers features signal conditioning, a 2-pole low-pass filter, and temperature compensation. Zero-g offset full scale span and filter cut-off are factory set and require no external devices. A full system self-test capability verifies system functionality.

Features of the MMA1220 and MMA1260 include:

- Integral signal conditioning
- Linear output
- Ratiometric performance (MMA1220)
- Bessel filters:
  - MMA1220 — 4th order Bessel filter preserves pulse shape integrity
  - MMA1260 — 2nd order bessel filter
- Calibrated self-test
- MMA1220 — low-voltage detect clock monitor
- EPROM parity check status
- Transducer hermetically sealed at wafer level for superior reliability
- Robust design, high shock survivability

Typical applications include:

- Vibration monitoring and recording
- Appliance control
- Mechanical bearing monitoring
- Computer hard drive protection
- Computer mouse and joysticks
- Virtual reality input devices
- Sports diagnostic devices and systems

Table 1-2. Accelerometer Operating Characteristics

Accelerometer	Pin Count	Zero-g (V)	Sensitivity (mV/g)	Acceleration Range (g)	Supply Voltage (V)
MMA1220	16	2.5	250	± 8	5
MMA1260	16	2.5	1200	± 1.5	5

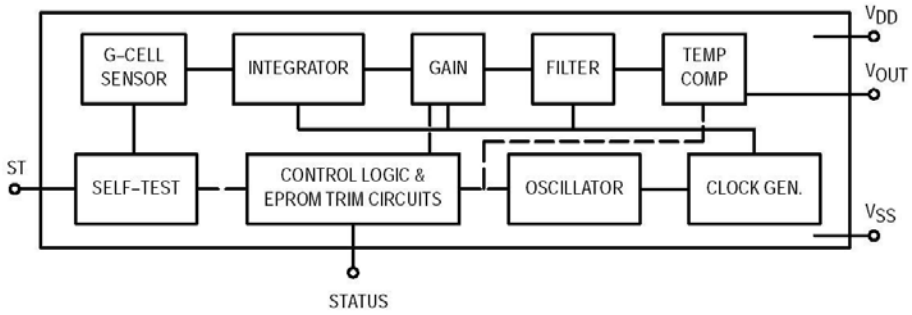


Figure 1-2. Simplified MMA1220 Block Diagram

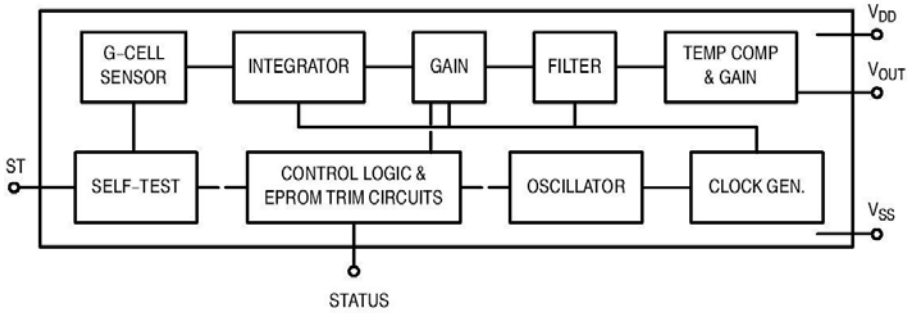


Figure 1-3. Simplified MMA1260 Block Diagram

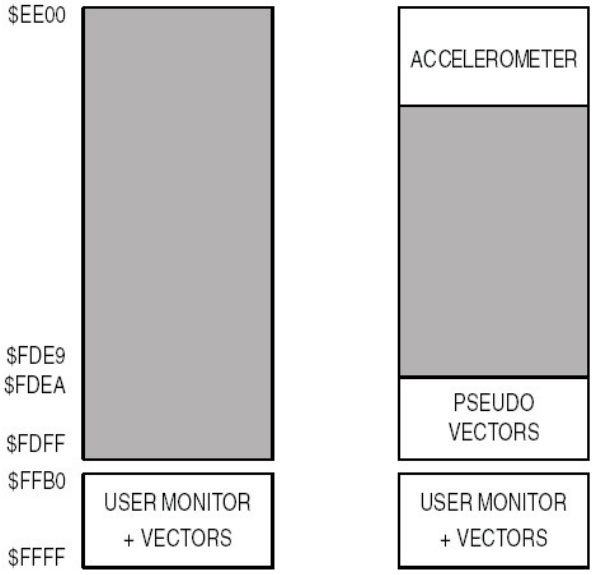
1.4 Setup Guide

This board operates in two different modes: programming mode and running mode. Programming mode allows downloading code to the MCU either by using a user monitor or a MON08 programmer. In running mode, the MCU executes the downloaded code.

NOTE: *Out of the box conditions suppose the board is programmed with "QY4 Accelerometer Code V1".*

1.4.1 Reprogramming the User FLASH Using User Monitor

The accelerometer demo board comes with extra (test) code in the FLASH. The following is a sample session that uses the CodeWarrior IDE to erase the user contents of the 4K FLASH module (see **Figure 1-4 (a)**) and reprogram with the accelerometer sample project using the user monitor mode (see **Figure 1-4 (b)**). The only hardware interface needed to do this is a serial cable connected between the board and your PC. Turn off power to the board at this time.



(a) User Monitor Erase **(b) Reprogram**

Figure 1-4. Sample Session

1. Launch the CodeWarrior IDE. (CW08 V2.1 or later).
2. Under File Menu select Open.
3. Find and Select QY4_Acc_Board folder and click Open (see **Figure 1-5**).

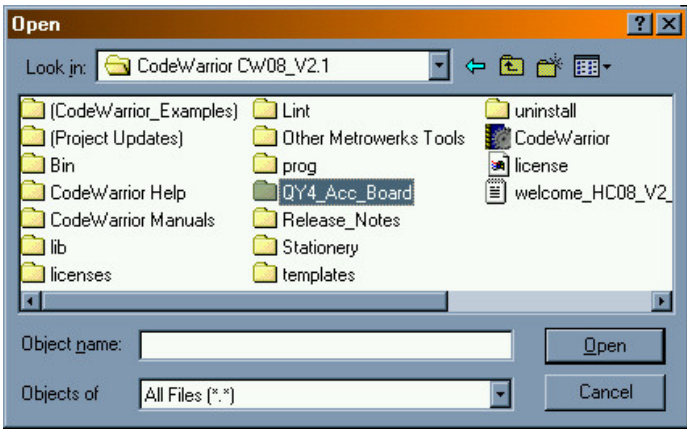


Figure 1-5. Find the Demo Project

4. Double-click on the QY4_Accelerometer.mcp file name. The CodeWarrior project will open (see **Figure 1-6**).

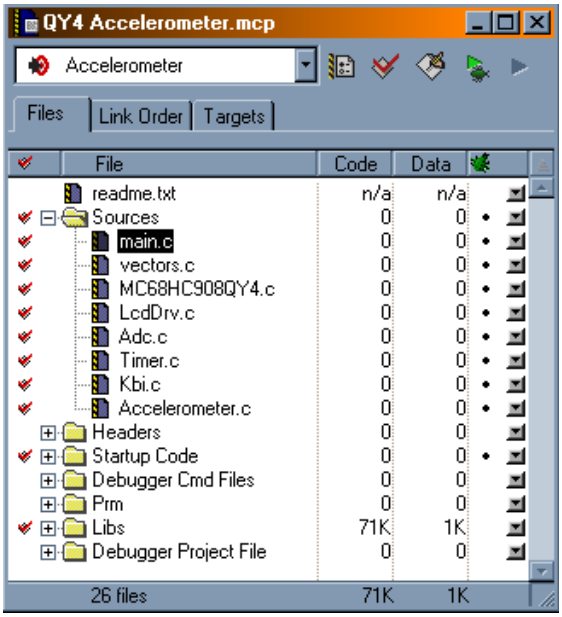


Figure 1-6. Project Window for Demo Software

5. Click the Debug icon (green arrow) in either the CodeWarrior menu bar or in the project window to start the debugger. The True-Time Simulator and Real-Time Debugger Window opens up. See **Figure 1-7**.

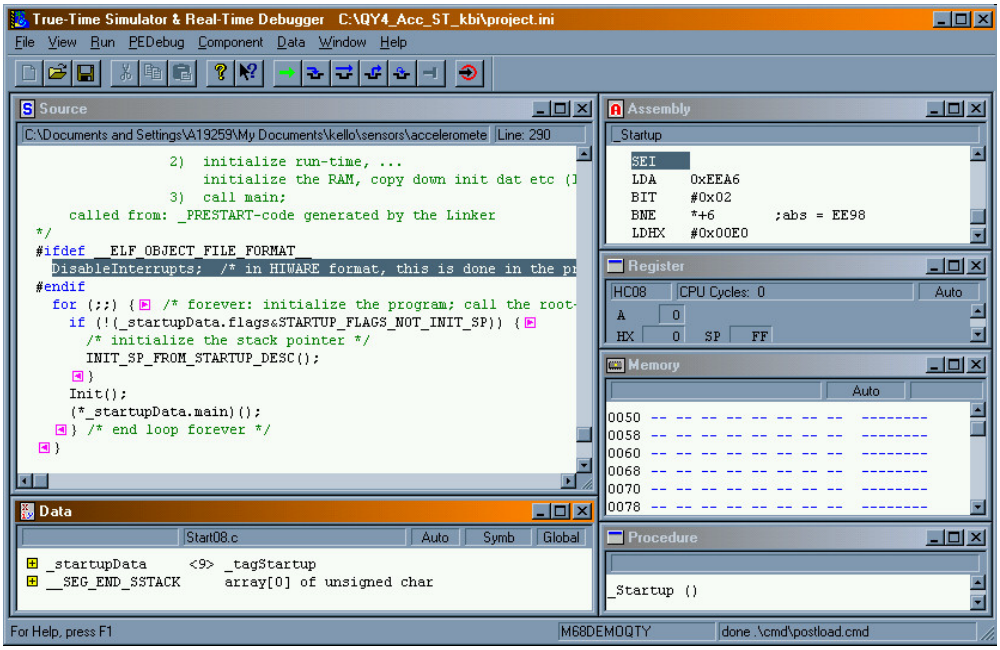


Figure 1-7. True-Time Simulator and Real-Time Debugger Window

6. From the PEDebug pulldown menu select Device: M68DEMOQTY. (Device:HC908QT4 –> 68HC08 –> QT/QY Family –> M68DEMOQTY). See **Figure 1-8**)

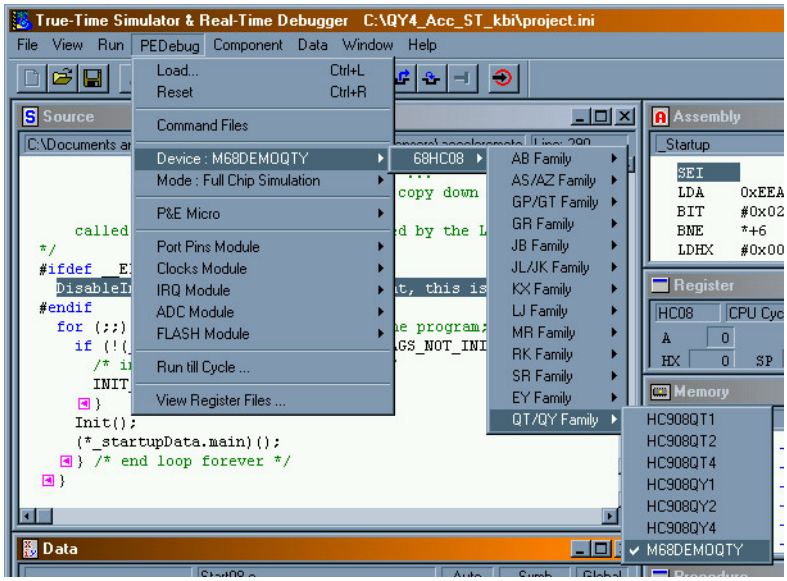


Figure 1-8. PEDebug Pulldown Menu for Device and Mode Selection

7. Likewise, select Mode: In-Circuit Debug/Programming. The PROG08SZ programmer launches and the M68DEMOQTY Board Reset Sequence window appears (see **Figure 1-9**).

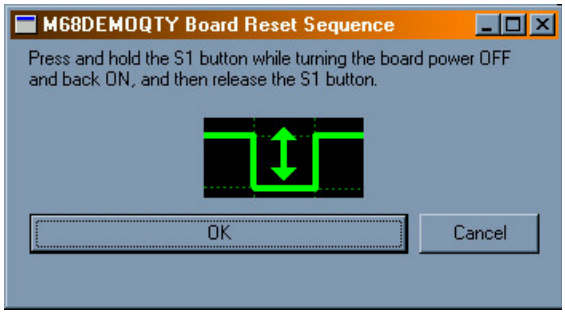


Figure 1-9. M68DEMOQTY Board Reset Sequence Window

If the Attempting to contact target window appears, select Class III, serial port, 9600 baud, IGNORE security failure, and click Contact target. See **Figure 1-10**.

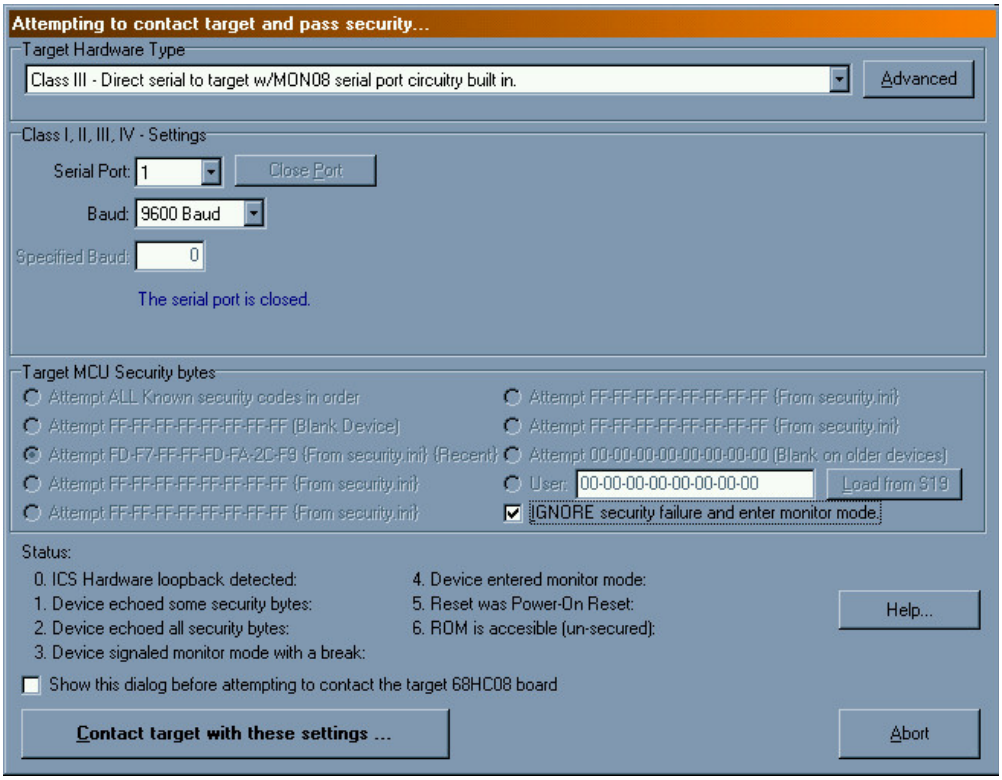


Figure 1-10. Attempting to Contact Target Window

8. Press and hold the S1 push button switch on the demo board while sliding S2 to the ON position.

NOTE: *The LCD will not reset if you did not hold the push button down while turning power on.*

9. Click OK.
The programmer Confirm window appears (see **Figure 1-11**).

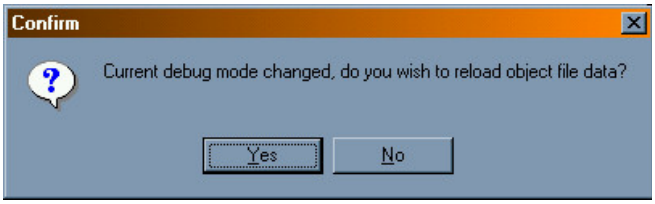


Figure 1-11. Programmer Confirm Window

10. Click Yes.
The Erase and Program FLASH window appears (see **Figure 1-12**).

Figure 1-12. Programmer Erase and Program FLASH Window

11. Click Yes.
The programmer begins an automated sequence that erases the FLASH then reprograms it with the project object file see **Figure 1-13**.

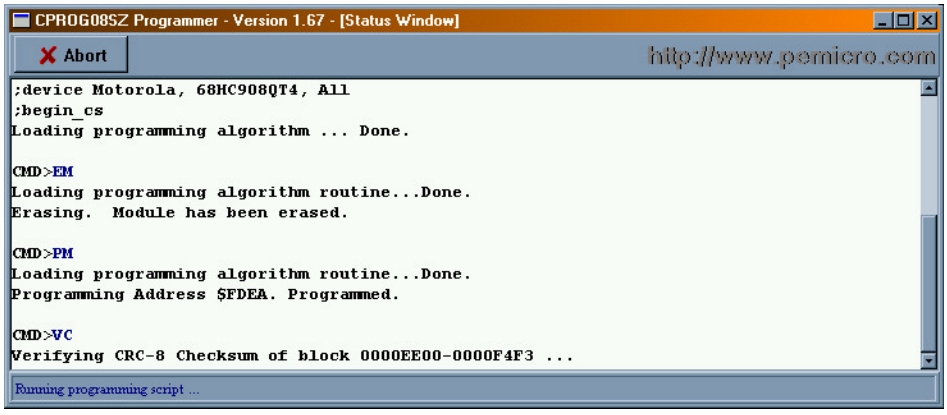


Figure 1-13. CPROG08SZ Programmer Window

The M68DEMOQTY Board Reset Sequence window appears twice more.

12. Turn off switch S2.
13. Hold S1 depressed while turning S1 back on.
14. Click OK.
15. Repeat steps 13–15 as prompted. The Debugger window reappears.
16. Right-click in the Memory window.
17. Select Address in menu.
18. Type in address EF82 and click OK. The new memory contents are shown. Contents will be FF if not programmed. See **Figure 1-14**.

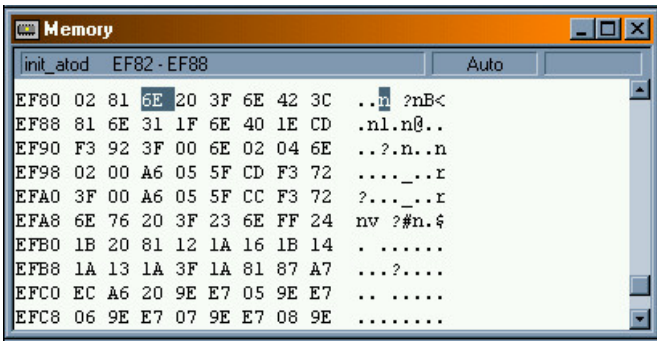


Figure 1-14. Verify New Code in FLASH

19. Close the Debugger and CodeWarrior windows.
20. Power down the demo board.
21. Disconnect the serial cable from the demo board.
22. Power up the demo board. The LCD should display the acceleration changes as the board is moved in its z-axis.

1.4.2 Reprogram the FLASH

To reprogram the entire FLASH for the demo board normal monitor mode will be needed. Other than the CodeWarrior tool, the following are required:

- Normal monitor mode interface
  - External
  - On-board
- Internal oscillator trim value
- Software

1.4.2.1 Normal Monitor Mode Interface

The normal monitor mode interface must be used to reprogram the entire FLASH memory. The interface brings serial communications, high voltage, an external clock, and mode settings to the board.

External Requirements

Refer to the application note entitled *Low-Cost Programming and Debugging Options for M68HC08 MCUs,* Motorola document order number AN2317/D, for one of several monitor mode circuits that can serve as the interface between the PC and the board. Any of the three circuits, or either a MON08-Cyclone or a MON08-Multilink from P&E Microcomputer Systems can be used.

On-Board Requirements

The demo board comes with an on-board MON08 interface. In order to use normal monitor mode, jumpers JP3 and JP4 should be removed. The board also comes with a 16-pin header (JP5) to directly plug the MON08 interface to program the MCU.

Connect the following signals from the monitor mode interface to the JP5 header on the demo board.

1. Connect the ground line to pin 2 (V_{SS}).
2. Connect the V_{TST} signal to pin 6 (PTA2/$\overline{IRQ}$).
3. Connect the COM signal to pin 8 (PTA0).
4. Connect the MOD1 signal to pin 10 (PTA4).
5. Connect the MOD0 signal to pin 12 (PTA1).
6. Connect the OSC signal to pin 13 (OSC1).
7. Connect the V_{DD} line to pin 15 (V_{DD}).

NOTE: *All remaining pins should be left unconnected.*

1.4.2.2 Internal Oscillator Trim Value

To find the internal oscillator trim value please refer to application note entitled *MC68HC908QY4 Internal Oscillator Usage Notes*, Motorola document order number AN2312/D, which provides some methods for finding this value.

1.4.2.3 Software

Program code to load into the Accelerometer Demo Board.

1.4.3 Reprogramming the FLASH Through a MON08 Interface

This procedure is, for the most part, for loading a completely new program into the FLASH memory on the demo board. This is an example using CodeWarrior and a MON08-Cyclone interface:

1. Launch CodeWarrior IDE (CW08 V2.1 or later).
2. Under File menu select New.
3. Select HC08 Stationery (**Figure 1-15**).

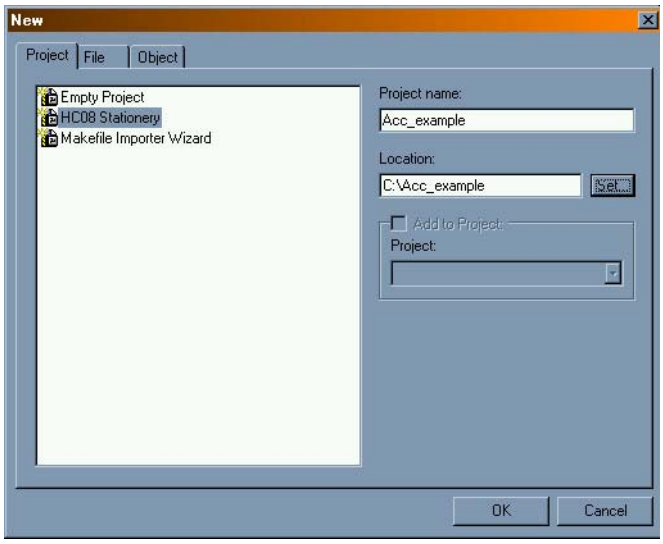


Figure 1-15. CodeWarrior New Project Window

4. Type in project name. Click OK.
5. In the New Project Stationery Window, click on + by QT_QY to expand selections. See **Figure 1-16**.

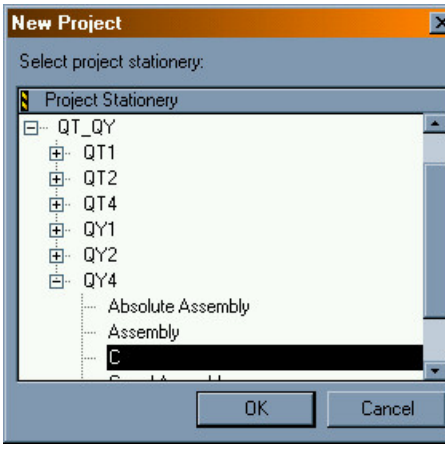


Figure 1-16. New Project Stationery Window

6. Click on + by QY4 to expand the selections.
7. Click on C. Click OK.
8. The project window is displayed. See **Figure 1-17**.
9. Click on the + to expand the Sources folder.
10. Click on the Sources folder to highlight it.
11. To create new files click on the New Text File icon.
12. Save document (place in project folder Sources or Headers). Name file and click on Save.

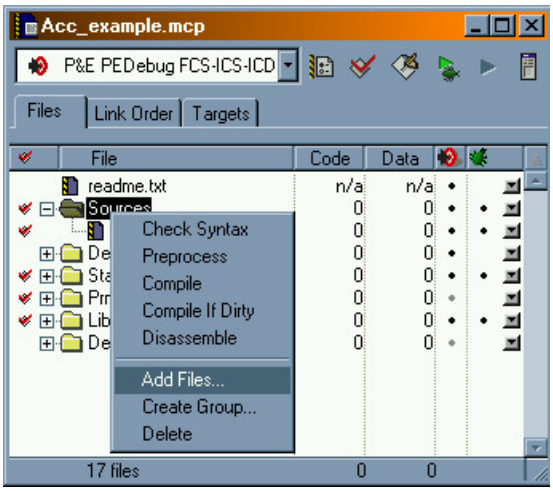


Figure 1-17. New Project Window – Add Files

13. To add files to the project right-click on the Sources folder and select Add Files… The “Select files to add…” window appears.
14. Navigate to the file to add and select it.
15. Click Open. The Add Files window appears. See **Figure 1-18**.
16. Uncheck P&E PEDebug FCS-ICS-ICD and MMDS-MMEVS. Click OK.

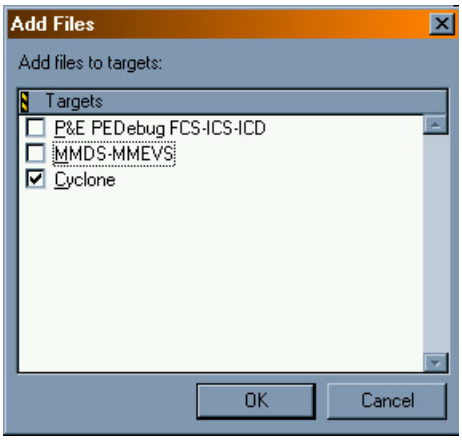


Figure 1-18. Add Files to Targets Window

17. Close the Project Messages window if it appears.
18. Repeat steps 13 thorough 18 to keep adding new files.
19. Set Trim Value (as explained in **1.4.2.2 Internal Oscillator Trim Value**).
20. Click on the Targets tab and select Cyclone. See **Figure 1-19**.
21. Click the Debug icon in the CodeWarrior tool bar. The True-Time Simulator & Real-Time Debugger launches.

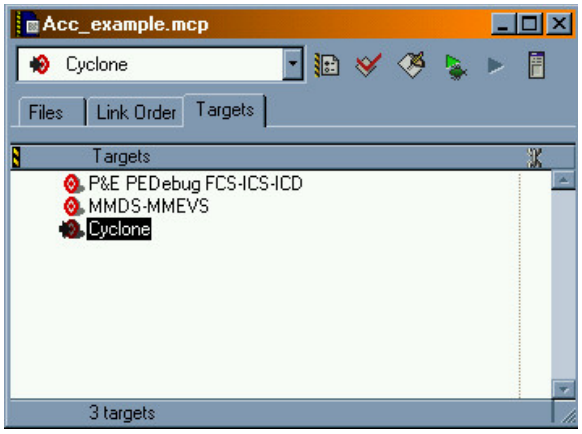


Figure 1-19. New Project Window – Set Targets

22. If the Attempting to contact target window (see **Figure 1-20**) does not appear, from the PEDebug pulldown menu select Device: HC908QY4. (Device: -> 68HC08 -> QT/QY Family -> HC908QY4).

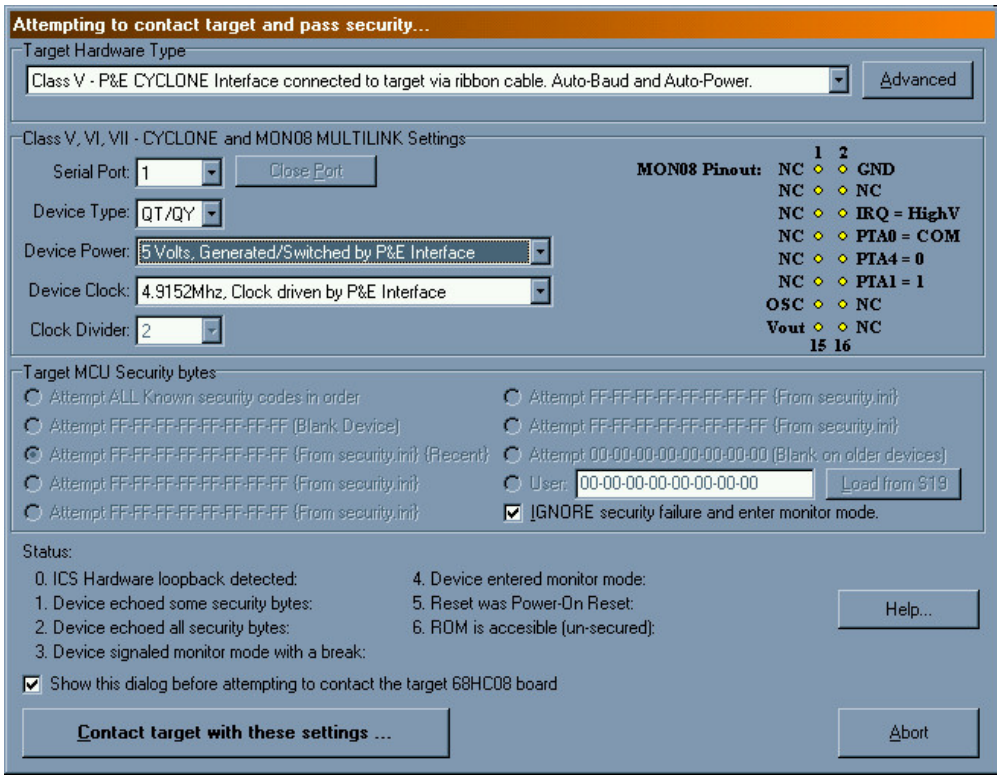


Figure 1-20. Attempting to Contact Target Window

23. Likewise, select Mode: In-Circuit Debug/Programming. The PROG08SZ programmer launches and the Attempting to contact target window appears.
24. Select Class V in Target Hardware Type. Also select serial port, QT/QY, 5 volts and 4.9152-MHz clock.
25. Check the IGNORE security failure box.
26. Click Contact target with these setting.
27. If the Confirm window appears click Yes to reload object data. See **Figure 1-21**.

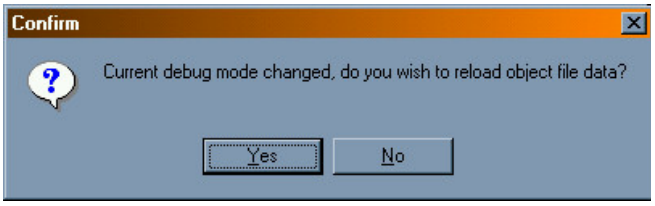


Figure 1-21. Programmer Confirm Window

28. Click Yes in the Erase and Program FLASH window. See **Figure 1-22**.

Figure 1-22. Programmer Erase and Program FLASH Window

29. The CPROG08S7 Programmer window appears as the programming script erases and reprograms the FLASH memory. Follow the instructions in the window and click OK as requested. The Command window in the True-Time Debugger indicates that the FLASH programming was successful. See **Figure 1-23**.
30. Close the Debugger and CodeWarrior windows.
31. Power down the demo board.
32. Disconnect the MON08 interface from the demo board. The new program is now loaded into the board.

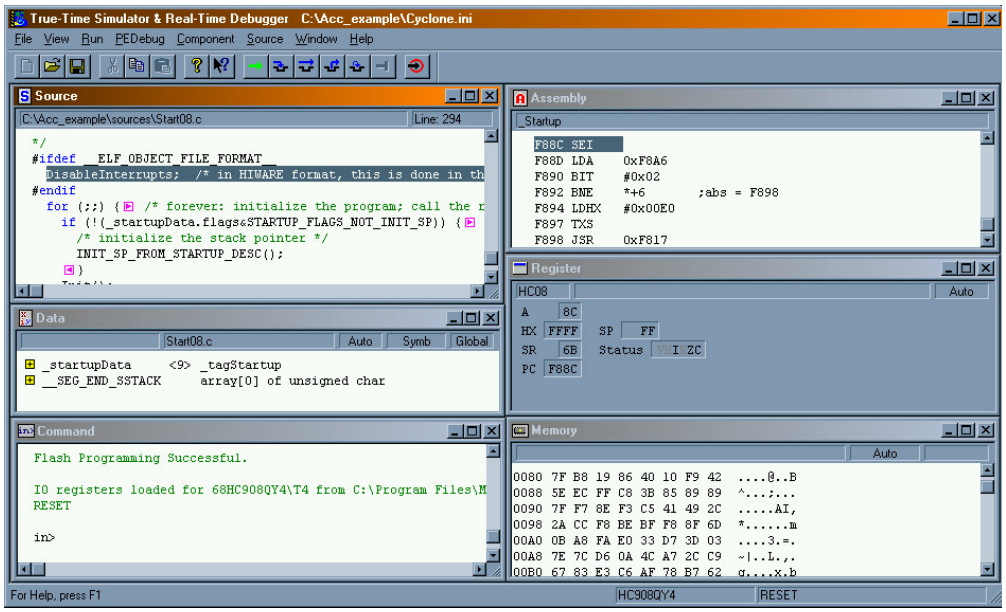


Figure 1-23. True-Time Debugger Window — Successful Programming

The Accelerometer Demo Board comes with a software application that runs using User Monitor. In the event that you accidentally erase the User Monitor from the demo board, the User Monitor must be reprogrammed for this application to run. Refer to the application note entitled *Reprogramming the M68DEMO908QT4*, Motorola document order number AN2322/D, for how to reprogram the User Monitor. Once the User Monitor is reprogrammed the software from the demo board can be programmed as shown in **1.4.1 Reprogramming the User FLASH Using User Monitor**.

Section 2. Operational Description

2.1 Introduction

Motorola's Low-Cost Accelerometer Reference Design is designed to provide an easy way to test and evaluate sensor functionality. The board includes two Motorola accelerometers with z-axis sensitivity, designed to sense acceleration changes and provide voltage proportional measurements for various applications. Board user interface includes a 16 x 2 line character LCD and two push buttons.

2.2 Electrical Characteristics

The electrical characteristics in **Table 2-1** apply to operation of the reference board at 25°C.

Table 2-1. Board Electrical Characteristics

Inputs	Min	Typ	Max	Unit
DC input voltage	7.5	9	10.5	V
DC input current	—	—	150	mA
Minimum logic 1 input voltage	3.5	—	—	V
Maximum logic 0 input voltage	—	—	1.5	V
RS-232 connection speed	9504	9600	9696	Baud

2.3 User Interfaces

The Accelerometer Board user interface (**Figure 2-1**) consists of:

- 16 x 2 line character LCD
- Trimmer potentiometer
- Power-on switch
- Jumper
- Two push buttons
- LED indicator
- MON08 interface
- RS0232 interface

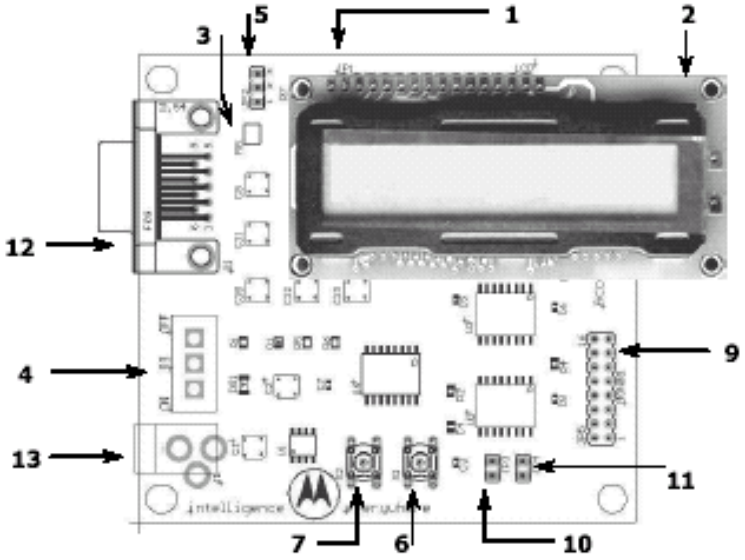


Figure 2-1. Accelerometer Reference Design Layout

Table 2-2. User Interfaces Reference Table

Num.	ID	Description
1	JP1	14-pin liquid crystal display (LCD) interface
2	LCD	16 characters per two lines LCD
3	R8	Trimmer to control the LCD's contrast
4	S3	Power-on switch
5	JP2	3-position jumper header: When shorted between position 1 and 2 it enables the LCD Placing it in position 2 and 3 will disable the LCD
6	S1	Push button resets current accelerometer's higher value detected by the software
7	S2	Push button toggles data acquisition between both accelerometers
8	DS1	Illuminates when power is applied to the board
9	JP5	16-pin MON08 interface to reprogram the board
10	JP3	2-position jumper header Removing it is necessary to reprogram MCU using MON08.
11	JP5	2-position jumper header Removing it is necessary to reprogram MCU using MON08.
12	X1	RS-232 interface, for monitor mode communication with a host computer
13	J1	DC jack to plug in 9-DC power adapter

2.4 Functional Description

Motorola's Low-Cost Accelerometer Reference Design comes programmed with code "QY4 Accelerometer Code V1". This program uses all features from the MMA1220 and MMA1260 accelerometers.

To turn on the board plug the DC adapter into the jack and turn the S3 power-on switch to its ON position. After turning on the board the first message displayed on the LCD is: "ACCELEROMETER EVALUATION BOARD". See **Figure 2-2**.

Figure 2-2. Accelerometer Board Welcome Message

2.4.1 Self-Test

When the welcome message disappears, "Running SelfTest on Demo Board" will be displayed on the LCD and the board will initiate a self-test routine on both accelerometers. This procedure allows verification of the mechanical and electrical integrity of the accelerometers.

The board will start testing the MMA1220. After testing procedure is finished on that accelerometer, the MMA1260 will initiate self-test.

CAUTION: *It is very important not to move the board at this moment as this may affect testing results.*

There are two possible messages to be displayed depending on self-test's results:

1. If testing results are successful, message "Self-Test OK…" will be displayed. See **Figure 2-3 (a)**.
2. The LCD will display "Self-Test Failed" (see **Figure 2-3 (b)**) if after five retries the accelerometer gives out a bad result on self-test. Reasons for this message to appear may be:
   a. Board was in motion when self-test was running.
   b. Selected accelerometer is not working properly and should be changed.

(a) Successful Self-Test Message

(b) Fault Self-Test Message

Figure 2-3. Self-Test Messages

2.4.2 Normal Mode

After the self-test procedure is finished, the application will begin running in normal mode. As both accelerometers have z-axis sensitivity, the board must be moved on its z-axis in order for the LCD to display any acceleration changes.

At start up, the MMA1220 will be the initial accelerometer being read (**Figure 2-4 (a)**). Pressing push button S2 will alternate between both accelerometers (**Figure 2-4 (b)** shows the data screen for the MMA1260 accelerometer). The information displayed in the data screen is:

- Accelerometer being read
- Current acceleration value sensed in g units (1g = 9.81m/s^2)
- Maximum acceleration value reported
- An over-acceleration warning

The information will display in the LCD as shown in **Figure 2-4**.

Acceleration data is displayed on the LCD as the board is moved on its z-axis. The top row in the LCD displays accelerations changes due to the board's z-axis acceleration changes and is being updated as data varies. As the readings in the accelerometer vary, the maximum value sensed to that point will be displayed in the bottom row of the LCD screen. To reset this value, push button S1 must be pressed.

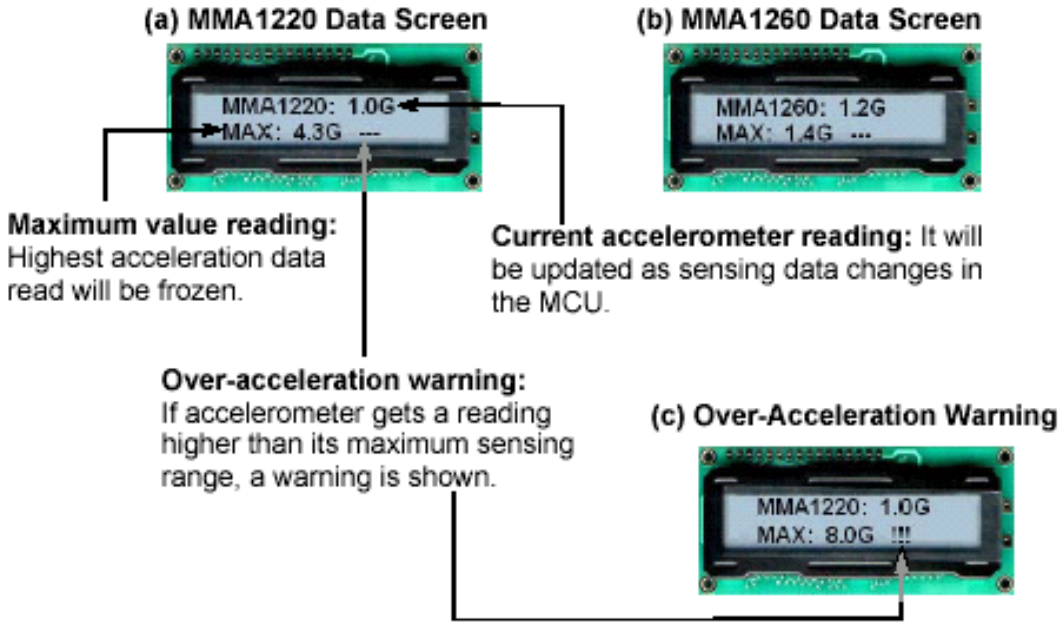


Figure 2-4. Data Screens

2.4.3 Over Acceleration

In case the accelerometer readings exceed those in its specification (± 8g for MMA1220, ± 1.5 for MMA1260), an over-acceleration warning is displayed (see **Figure 2-4 (c)**). For normal acceleration, reading "---" will be displayed. In case the acceleration reading is higher than the maximum value previously described, "!!!" will be displayed. To reset this condition from the LCD press S1.

NOTE: *It is very important to understand that although the accelerometers might be able to read acceleration changes above their specification limits, Motorola only guarantees accurate readings to those limits.*

2.4.4 Getting Started

The Accelerometer Board uses a 9-Vdc adapter plugged into J1 to power-on. A serial cable will only be needed to reprogram the board using monitor mode.

The following accessories are not included with the Accelerometer Board:

- DC adapter
- Serial cable
- MON08 programmer

Operational Description

Section 3. Schematics and Parts List

3.1 Introduction

Schematic and parts list detail are documented in this section.

3.2 Schematics

A schematic for the Low-Cost Accelerometer Reference Design appears in **Figure 3-1**. Interrupted lines coded with the same letters are electrically connected.

3.3 Part List

A detailed parts list for the Low-Cost Accelerometer Reference Design is shown in **Table 3-1**.

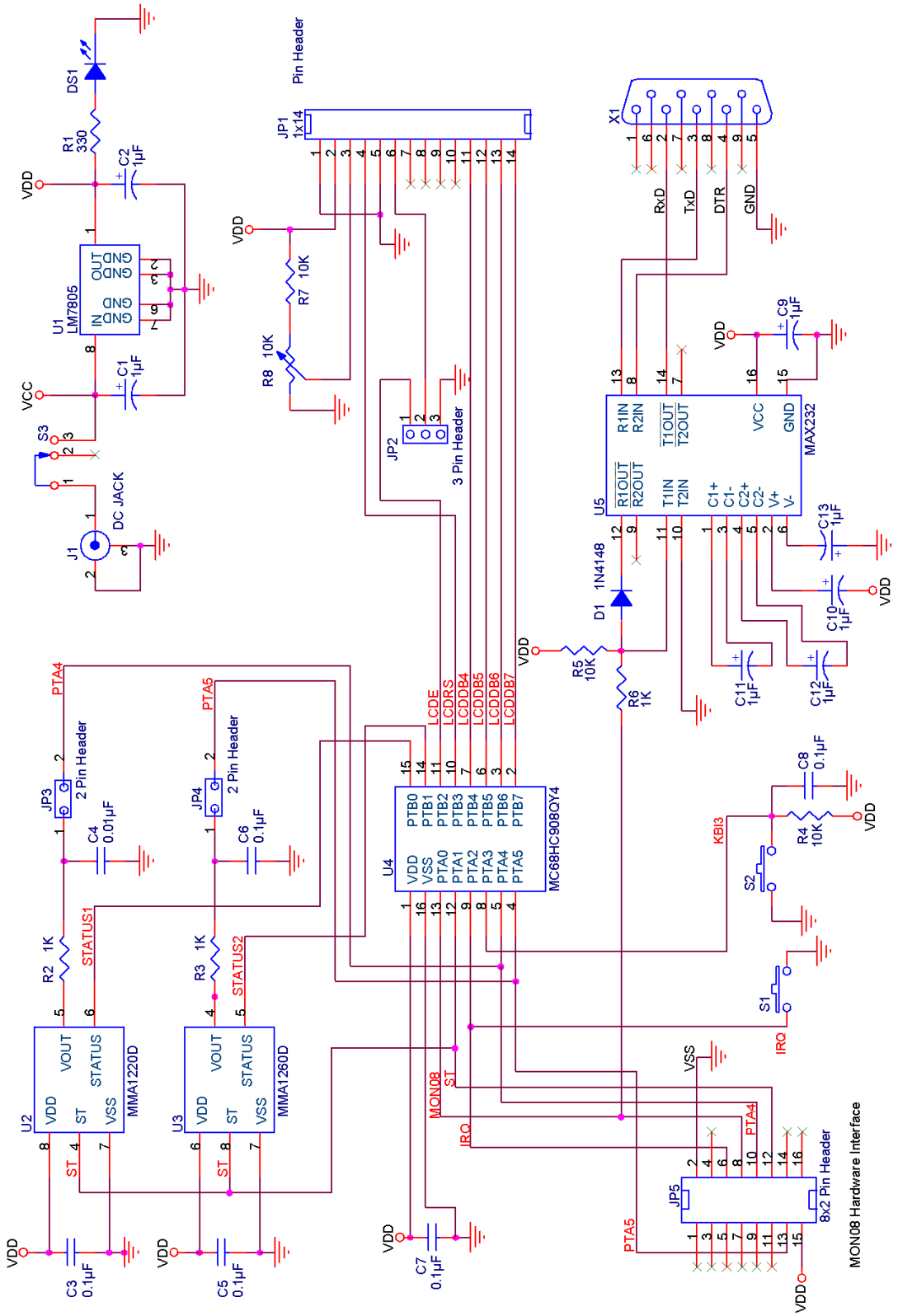


Figure 3-1. Accelerometer Board Schematic

Table 3-1. Bill of Materials

Qty	Value	Description	Label	Manufacturer	Part Number	Distributor	Distributor Part Number
Capacitors							
7	1uF	Electrolytic Capacitor	C1, C2, C9, C10 C11, C12, C13	United Chemi-Con	MVK50VC1R0MD55TP	Newark	16F9832
5	.1uF	Capacitor (0805)	C3, C5, C6, C7, C8	Kemet	C0805C104K5RACTR	Newark	96F8740
1	.01uF	Capacitor (0805)	C4	Kemet	C0805C103K5RACTR	Newark	93F2330
				Diodes			
1	1N4148	Switching Diode 1n4148	D1	On Semiconductors	MMSD4148T1	Newark	92B1276
1	Red	Red LED (1206)	DS1	Chicago Miniature Lamp, Inc.	CMD15-21VRC/TR8	Newark	93B3495
Integrated Circuits							
1	LM7805	Voltage Regulator (5V, 100mA)	U1	On Semi	MC78L05ABDR2	Newark	20C1133
1	MMA1220	Accelerometer (8Gs)	U2	Motorola	MMA1220D		
1	MMA1260	Accelerometer (1.5Gs)	U3	Motorola	MMA1260D		
1	MAX232	MAX232	U5	TI	MAX232D	Newark	91F3642
Connectors							
1	DC Jack	DC Jack 2.00 mm	J1	SPC Connectors	SPC4077	Newark	84N1161
1	DB9	DB9 (Female)	P1	SPC Connectors	DE-9S-PCB	Newark	44N8882
1	1x14 pins	Pin Receptacle		Tyco	1-534237-2	Newark	52F3009
Jumpers							
2	1x2 pins	Jumper	JP3, JP4				
Headers							
1	1x14 pins	Pin Header	JP1				
1	1x3 pins	Pin Header	JP2				
1	2x8 pins	Header MON08	JP5				
LCD							
1		2 x 16 Characters LCD	LCD	Tianma	TM162AAA6-2		

Table 3-1. Bill of Materials (Continued)

Qty	Value	Description	Label	Manufacturer	Part Number	Distributor	Distributor Part Number
				Microcontroller			
1	QY4	Microcontroller	U4	Motorola	MC68HC908QY4CDW		
				Resistors			
1	330, 1/4W	Resistor (0805)	R1	Vishay	CRCW0805331JRT1	Newark	66F9045
3	1K, 1/10W	Resistor (0805)	R2, R3, R6	Vishay	CRCW08051001FRT1	Newark	05F1507
3	10K, 1/10W	Resistor (0805)	R4, R5, R7	Vishay	CRCW08051002FRT1	Newark	05F1511
				Potentiometer			
1	10K	Trimmer (3313)	R8	Bourns, Inc.	3313J-1-103E	Newark	04F8036
				Switches			
2		Push Button	S1, S2	Omron Components USA	B3F-1000	Newark	52F3545
1		Slide Switch	S3				

Section 4. Hardware Design Considerations

4.1 Introduction

The board developed for the reference design includes the hardware required to run the application along with the interface to download new code to the MCU allowing the design engineer to tailor the application. The FLASH on the MCU can be reprogrammed through RS-232 interface or using a MON08 interface.

4.2 MC68HC908QY4 Microcontroller Unit (MCU)

The board comes with an 8-bit microcontroller (see **Figure 4-1**) preprogrammed to demonstrate the capabilities of the accelerometers MMA1220D and MMA1260D.

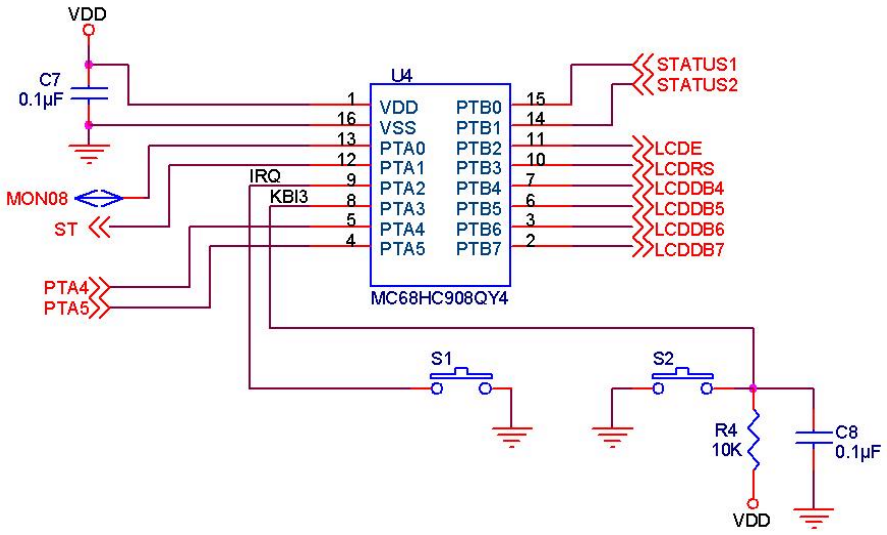


Figure 4-1. HC908QY4 MCU

4.3 DC Power Supply

The main power input to the board is through a power jack (**J1**). From the line input jack J1, with switch **S3** in its On position, there is a derivation that generates the low voltage power supply (5 Vdc). This power supply is generated using voltage regulator LM78L05A (**U1**). Input bypass capacitor **C1** has been added to the regulator's input to provide good high–frequency characteristics to insure stable operation. An output bypass capacitor (**C2**) has also been added for improved transient response. A red light-emitting diode (LED) (**DS1**) was included to show proper +5 Vdc power supply operation. **R1** will ensure proper current feed into DS1. See **Figure 4-2**.

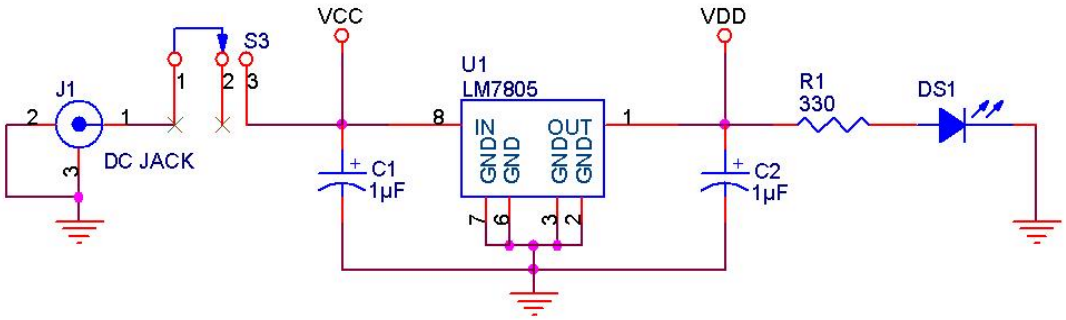


Figure 4-2. DC Power Supply

4.4 RS-232 Interface and MON08 Hardware Interface

These board interfaces (see **Figure 4-3**) allow the user to program the FLASH in the MCU (MON08). Also, communication via the RS-232 interface when operating in run mode.

- The board provides an RS-232 interface by the use of a MAX232 (**U5**). **U5** is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels.
- Capacitor **C9** is used on V_{DD} to decouple the power source. Capacitors **C10, C11, C12**, and **C13** are typical MAX232 operation.
- **R5** is a pull-up resistor. **R6** and **D1** are part of the monitor mode circuit implementation. **D1** is a fast switching diode to avoid bus conflict.
- JP5 is an 8 x 2 pin header connected to the MCU's ports. The pin layout for JP5 is arranged to interface with a MON08 programmer.

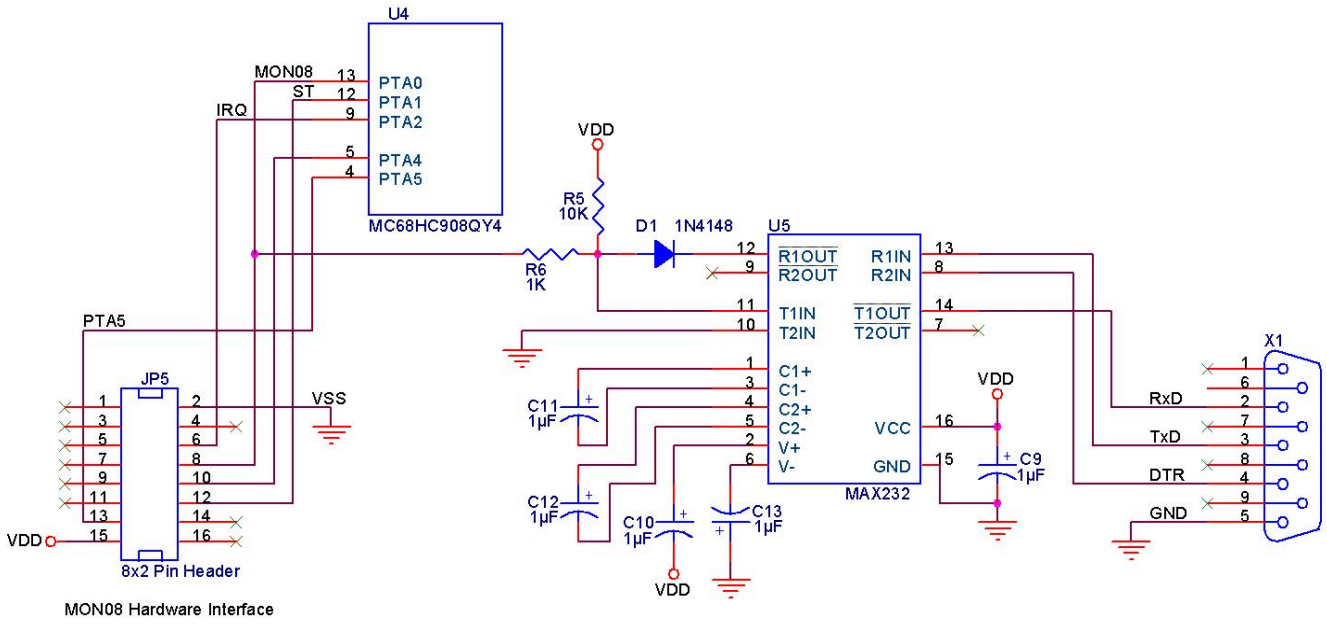


Figure 4-3. RS-232 and MON08 Interfaces

4.5 MMA1220 and MMA1260 Accelerometers

The board contains two accelerometers (**U2** and **U3**) connected to the MCU. Capacitors **C3** and **C5** are used on V_{DD} to decouple the power source. An RC filter (**R2** and **C4** for U2, **R3** and **C6** for U3) is placed on both accelerometers' outputs to minimize clock noise. See **Figure 4-4**.

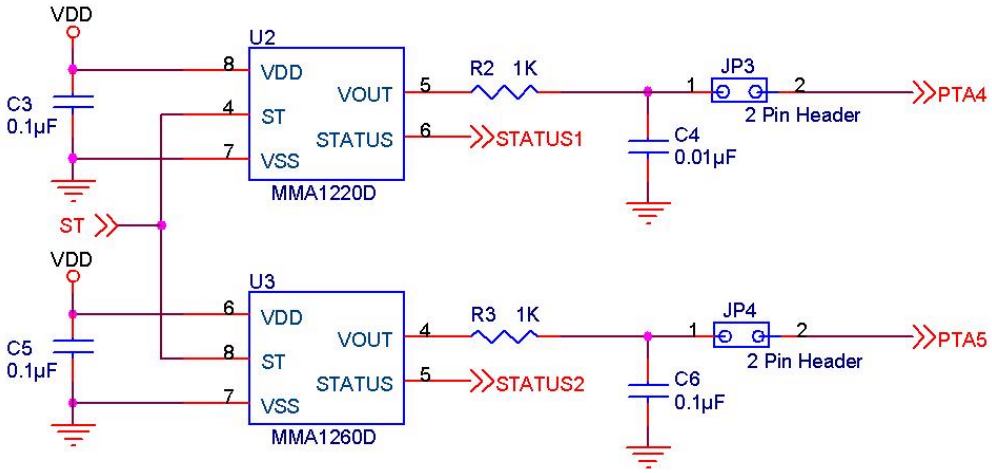


Figure 4-4. MMA1220 and MMA1260 Accelerometers

4.6 LCD Interface

The board contains an LCD as the main user interface feedback. The LCD contains an internal driver. The display is controlled and managed through the microcontroller's port signals. Resistor **R7** and trimmer **R8** control the LCD display's contrast. **Figure 4-5** shows the hardware interface.

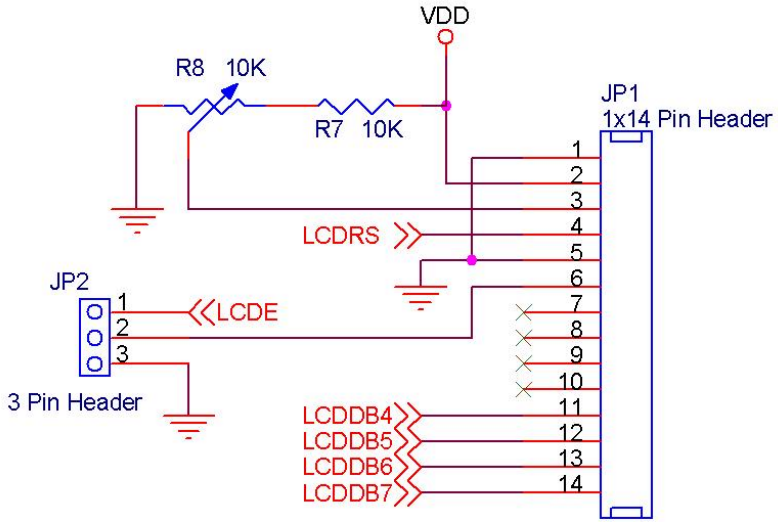


Figure 4-5. LCD Interface

Section 5. Software Design Considerations

5.1 Introduction

The code for the Low-Cost Accelerometer Reference Design is an endless loop that refreshes the LCD with the newest top accelerometer value. Interrupts from the timer and the push button will:

- Serve as debouncing (for the push buttons),
- Select the accelerometer to be read, or
- Reset the top value.

5.2 Application State Diagram

As **Figure 5-1** shows, the application state consists of the hardware initialization routine, hardware testing, followed by main loop with background tasks. Push button service and debouncing is managed by the interrupt routines.

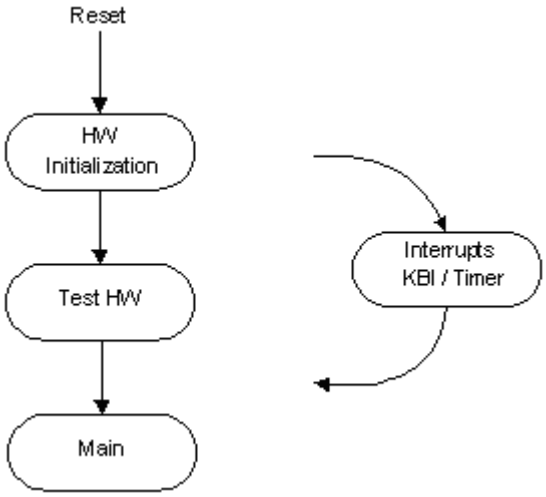


Figure 5-1. Application State Diagram

Software Design Considerations

The brief description of the application for the Accelerometer Reference Design follows:

- Hardware initialization routine:
  - Configuration register setup
  - Liquid crystal display (LCD) initialization
  - Port A (PTA) setup
  - Analog-to-digial (ADC) initialization
  - Timer initialization
  - Keyboard interface (KBI) initialization
- Hardware testing routine:
  - Test accelerometer status/self test
  - Display self test results on LCD
- Main loop:
  - Get new value
  - Format data
  - Check for new top value
  - Refresh LCD data
- Timer overflow interrupt handler:
  - Clear overflow flag
  - Push button debounce
  - If S2 is pressed switch accumulator read
  - If S1 is pressed reset top value
  - Acknowledge pending interrupts
  - Re-enables keyboard interrupts
- KBI interrupt handler:
  - Debouncing count variable initialization
  - Acknowledge KB Interrupts
  - Mask KB interrupts during 51.2 ms

5.3 Data Flow

Figure 5-2 shows the main loop flowchart.

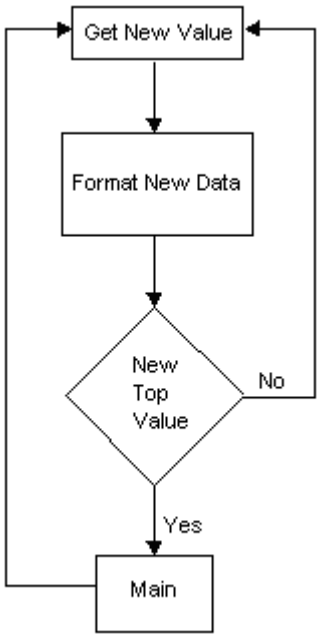


Figure 5-2. Main Loop Flowchart

5.4 Routines Description

This subsection describes the various routines.

5.4.1 Main(void)

This routine configures the MCU and its peripherals, tests the accelerometers' status. It also includes the main endless loop that is constantly refreshing the LCD with the newest top accelerometer value.

5.4.2 InitHardware(void)

Subroutine that initializes CONFIG registers, LCD, PTA, and accelerometers' status latch.

5.4.3 Wait1ms(void)

Fixed delay that waits for 1 millisecond.

5.4.4 WaitNms(UINT8 n)

This delay function waits for the amount of milliseconds specified by the parameter n.

Parameters:

n is an 8-bit unsigned value that specifies the amount of milliseconds that the delay function is going to wait.

5.4.5 Toggle(void)

This function toggles the E line on the LCD. The E line is connected to PTB2.

5.4.6 LcdCommand8(UINT8 u8Command)

Subroutine for sending control bytes to the LCD. This routine is used with the LCD in 8-bit mode. It is used to set the LCD to 4-bit mode.

Parameters:

u8Command is an 8-bit value for control commands.

5.4.7 LcdCommand(UINT8 u8Command)

Subroutine for sending control bytes to the LCD. This routine sends the 8-bit value in two parts, since the LCD is operating in 4-bit mode.

Parameters:

u8Command is an 8-bit value for different control commands.

5.4.8 InitLcd(void)

This subroutine is used to initialize the LCD to 4-bit mode and general settings.

5.4.9 DisplayString(UINT8 *str)

This function displays a string in the LCD at the current cursor position by sending the characters in the string until a NULL character is found.

Parameters:

*str is a pointer to the string to be displayed in the LCD.

5.4.10 InitAdc(void)

This routine will setup the ADC, configure the ADC prescaler, select single conversion mode and channel. The ADC will read the input from on-board accelerometers. Channel 2 corresponds for MMA1220 accelerometer and channel 3 for MMA1260.

5.4.11 AdcGetValue(UINT8 u8AccChannel)

This function generates a single conversion and waits until it is completed. The result is returned as an 8-bit variable. This loop is ended in case the ADC takes more than 10 attempts to obtain the converted value.

Parameters:

u8AccChannel is an 8-bit variable that indicates the channel to be measured.

5.4.12 InitTimer(void)

This function turns on timer channel 1 such that it counts between 0 and $FF. An overflow interrupt occurs every 256*64 = 16384 bus cycles (5.12 ms @3.2-MHZ bus).

5.4.13 _TOF_Interrupt(void)

This is the timer overflow interrupt service routine. The timer rolls over every 256T counts of the timer (TMOD = $FF). This overflow routine period is used as a time-base for the debounce of the push buttons on the board.

5.4.14 InitKbi(void)

This function configures the KBI module to accept interrupts on PTA2 and PTA3 that are connected to the S1 and S2 push buttons on the board.

5.4.15 _KBD_Interrupt(void)

This is the keyboard interrupt service routine. This routine sets the debounce time to 51.2 ms and masks the keyboard interrupts.

5.4.16 AccDataConvGs(UINT8 u8RawValue)

This subroutine converts the acquired data from the accelerometers to g units. This value is obtained from the actual accelerometer selected. An 8-bit value containing the data formatted in gs is returned.

Parameters:

u8RawValue is an 8-bit value that contains the data to be converted to gs.

5.4.17 AccTest(UINT8 u8AccUnderTest)

Subroutine that tests the accelerometers' status. This routine sets the self-test pin to high to test the accelerometers' functionality to be within specifications. This test is run a maximum amount of five times or until the first testing is passed. The output of the testing is reflected to the LCD.

Parameters:

U8accundertest is an 8-bit value that specifies the accelerometer to be used.

5.4.18 AccDataFormat(UINT8 u8AccData)

This function formats the data acquired from the ADC to the LCD message buffer. The data is formatted to g units and positioned in the message buffer to be displayed in the LCD. A Boolean variable is returned reflecting if a new top value is found.

Parameters:

u8AccData is an 8-bit value with the data acquired from the ADC.

5.5 MCU Usage

Table 5-1 shows how much memory is needed to program the MCU with the Low-Cost Accelerometer Reference Design application.

Table 5-1. RAM and FLASH Memory Usage

Memory (in 8-Bit Words)	Available (MC68HC908QY4)	Used (Application + Stack)
Program FLASH	4096	1442

NOTE: *Due of the application's total number of bytes, a MC68HC908QY1 may also be used on this reference design.*

Section 6. Source Code

6.1 Introduction

This section contains the source code for:

- Include files:
  - MC68HC908QY4.H
  - NITRON_MASKS.H
  - TYPES.H
  - LCDDRV.H
  - ADC.H
  - TIMER.H
  - KBI.H
  - ACCELEROMETER.H
- Source files:
  - MC68HC908QY4.C
  - VECTORS.C
  - MAIN.C
  - LCDDRV.C
  - ADC.C
  - TIMER.C
  - KBI.C
  - ACCELEROMETER.C

6.2 Include Files

6.2.1 MC68HC908QY4.H

```
/*
** ###################################################################
**
**     THIS BEAN MODULE IS GENERATED BY THE TOOL. DO NOT MODIFY IT.
**
**     Filename  : MC68HC908QY4.H
**
**     Processor : MC68HC908QY4CP
**
**     Version   : Driver 01.02
**
**     Compiler  : Metrowerks HC08 C Compiler V-5.0.13
**
**     Date/Time : 05.08.2002, 05:32
**
**     Abstract  :
**
**         This bean "IO_Map" implements an IO devices mapping.
**
**     Settings  :
**
**
**
**     Contents  :
**
**         No public methods
**
**
**     (c) Copyright UNIS, spol. s r.o. 1997-2002
**
**     UNIS, spol. s r.o.
**     Jundrovska 33
**     624 00 Brno
**     Czech Republic
**
**     http      : www.processorexpert.com
**     mail      : info@processorexpert.com
**
** ###################################################################
*/
/*Types definition*/
#ifndef _MC68HC908QY4_H
#define _MC68HC908QY4_H

typedef unsigned char bool;
typedef unsigned char byte;
typedef unsigned int word;
typedef unsigned long dword;
typedef unsigned long dlong[2];
#define __RESET_WATCHDOG() {asm sta COPCTL;} /* just write a byte to feed the dog */
```

```
#pragma MESSAGE DISABLE C1106 /* WARNING C1106: Non-standard bitfield type */
/* Based on CPU DB MC68HC908QT4, version 2.87.081 */

/*** PTA - Port A Data Register ***/
typedef union {
  byte Byte;
  struct {
    byte PTA0    :1;           /* Port A Data Bit 0, Keyboard interrupt pin 0 */
    byte PTA1    :1;           /* Port A Data Bit 1, Keyboard interrupt pin 1 */
    byte PTA2    :1;           /* Port A Data Bit 2, Keyboard interrupt pin 2 */
    byte PTA3    :1;           /* Port A Data Bit 3, Keyboard interrupt pin 3 */
    byte PTA4    :1;           /* Port A Data Bit 4, Keyboard interrupt pin 4 */
    byte PTA5    :1;           /* Port A Data Bit 5, Keyboard interrupt pin 5 */
    byte AWUL    :1;           /* Auto Wake-up Latch Data Bit */
    byte         :1;
  } Bits;
  struct {
    byte PTA     :6;
    byte AWUL    :1;
    byte         :1;
  } MergedBits;
} PTASTR;
extern volatile PTASTR _PTA @0x00000000;
#define PTA _PTA.Byte
#define PTA_PTA0 _PTA.Bits.PTA0
#define PTA_PTA1 _PTA.Bits.PTA1
#define PTA_PTA2 _PTA.Bits.PTA2
#define PTA_PTA3 _PTA.Bits.PTA3
#define PTA_PTA4 _PTA.Bits.PTA4
#define PTA_PTA5 _PTA.Bits.PTA5
#define PTA_AWUL _PTA.Bits.AWUL
#define PTA_PTA _PTA.MergedBits.PTA

/*** DDRA - Data Direction Register A ***/
typedef union {
  byte Byte;
  struct {
    byte DDRA0   :1;           /* Data Direction Register A Bit 0 */
    byte DDRA1   :1;           /* Data Direction Register A Bit 1 */
    byte         :1;
    byte DDRA3   :1;           /* Data Direction Register A Bit 3 */
    byte DDRA4   :1;           /* Data Direction Register A Bit 4 */
    byte DDRA5   :1;           /* Data Direction Register A Bit 5 */
    byte         :1;
    byte         :1;
  } Bits;
  struct {
    byte DDRA    :2;
    byte         :1;
    byte DDRA_3  :3;
    byte         :1;
    byte         :1;
  } MergedBits;
} DDRASTR;
```

```
extern volatile DDRASTR _DDRA @0x00000004;
#define DDRA _DDRA.Byte
#define DDRA_DDRA0 _DDRA.Bits.DDRA0
#define DDRA_DDRA1 _DDRA.Bits.DDRA1
#define DDRA_DDRA3 _DDRA.Bits.DDRA3
#define DDRA_DDRA4 _DDRA.Bits.DDRA4
#define DDRA_DDRA5 _DDRA.Bits.DDRA5
#define DDRA_DDRA _DDRA.MergedBits.DDRA
#define DDRA_DDRA_3 _DDRA.MergedBits.DDRA_3


/*** PTB - Port B Data Register ***/
typedef union {
  byte Byte;
  struct {
    byte PTB0    :1;           /* Port B Data Bit 0 */
    byte PTB1    :1;           /* Port B Data Bit 1 */
    byte PTB2    :1;           /* Port B Data Bit 2 */
    byte PTB3    :1;           /* Port B Data Bit 3 */
    byte PTB4    :1;           /* Port B Data Bit 4 */
    byte PTB5    :1;           /* Port B Data Bit 5 */
    byte PTB6    :1;           /* Port B Data Bit 6 */
    byte PTB7    :1;           /* Port B Data Bit 7 */
  } Bits;
} PTBSTR;
extern volatile PTBSTR _PTB @0x00000001;
#define PTB _PTB.Byte
#define PTB_PTB0 _PTB.Bits.PTB0
#define PTB_PTB1 _PTB.Bits.PTB1
#define PTB_PTB2 _PTB.Bits.PTB2
#define PTB_PTB3 _PTB.Bits.PTB3
#define PTB_PTB4 _PTB.Bits.PTB4
#define PTB_PTB5 _PTB.Bits.PTB5
#define PTB_PTB6 _PTB.Bits.PTB6
#define PTB_PTB7 _PTB.Bits.PTB7


/*** DDRB - Data Direction Register B ***/
typedef union {
  byte Byte;
  struct {
    byte DDRB0   :1;           /* Data Direction Register B Bit 0 */
    byte DDRB1   :1;           /* Data Direction Register B Bit 1 */
    byte DDRB2   :1;             /* Data Direction Register B Bit 1 */
    byte DDRB3   :1;           /* Data Direction Register B Bit 3 */
    byte DDRB4   :1;           /* Data Direction Register B Bit 4 */
    byte DDRB5   :1;           /* Data Direction Register B Bit 5 */
    byte DDRB6   :1;           /* Data Direction Register B Bit 6 */
    byte DDRB7   :1;           /* Data Direction Register B Bit 7 */
  } Bits;
} DDRBSTR;
extern volatile DDRBSTR _DDRB @0x00000005;
#define DDRB _DDRB.Byte
#define DDRB_DDRB0 _DDRB.Bits.DDRB0
#define DDRB_DDRB1 _DDRB.Bits.DDRB1
```

```
#define DDRB_DDRB2 _DDRB.Bits.DDRB2
#define DDRB_DDRB3 _DDRB.Bits.DDRB3
#define DDRB_DDRB4 _DDRB.Bits.DDRB4
#define DDRB_DDRB5 _DDRB.Bits.DDRB5
#define DDRB_DDRB6 _DDRB.Bits.DDRB6
#define DDRB_DDRB7 _DDRB.Bits.DDRB7


/*** KBSCR - Keyboard Status and Control Register ***/
typedef union {
  byte Byte;
  struct {
    byte MODEK   :1;           /* Keyboard Triggering Sensitivity Bit */
    byte IMASKK  :1;           /* Keyboard Interrupt Mask Bit */
    byte ACKK    :1;           /* Keyboard Acknowledge Bit */
    byte KEYF    :1;           /* Keyboard Flag Bit */
    byte         :1;
    byte         :1;
    byte         :1;
    byte         :1;
  } Bits;
} KBSCRSTR;
extern volatile KBSCRSTR _KBSCR @0x0000001A;
#define KBSCR _KBSCR.Byte
#define KBSCR_MODEK _KBSCR.Bits.MODEK
#define KBSCR_IMASKK _KBSCR.Bits.IMASKK
#define KBSCR_ACKK _KBSCR.Bits.ACKK
#define KBSCR_KEYF _KBSCR.Bits.KEYF

/*** KBIER - Keyboard Interrrupt Enable Register KBIER ***/
typedef union {
  byte Byte;
  struct {
    byte KBIE0   :1;           /* Keyboard Interrupt Enable Bit 0 */
    byte KBIE1   :1;           /* Keyboard Interrupt Enable Bit 1 */
    byte KBIE2   :1;           /* Keyboard Interrupt Enable Bit 2 */
    byte KBIE3   :1;           /* Keyboard Interrupt Enable Bit 3 */
    byte KBIE4   :1;           /* Keyboard Interrupt Enable Bit 4 */
    byte KBIE5   :1;           /* Keyboard Interrupt Enable Bit 5 */
    byte AWUIE   :1;           /* Auto Wake-up Interrupt Enable Bit */
    byte         :1;
  } Bits;
  struct {
    byte KBIE    :6;
    byte AWUIE   :1;
    byte         :1;
  } MergedBits;
} KBIERSTR;
extern volatile KBIERSTR _KBIER @0x0000001B;
#define KBIER _KBIER.Byte
#define KBIER_KBIE0 _KBIER.Bits.KBIE0
#define KBIER_KBIE1 _KBIER.Bits.KBIE1
#define KBIER_KBIE2 _KBIER.Bits.KBIE2
#define KBIER_KBIE3 _KBIER.Bits.KBIE3
#define KBIER_KBIE4 _KBIER.Bits.KBIE4
```

Source Code

```
#define KBIER_KBIE5 _KBIER.Bits.KBIE5
#define KBIER_AWUIE _KBIER.Bits.AWUIE
#define KBIER_KBIE _KBIER.MergedBits.KBIE


/*** CONFIG2 - Configuration Register 2 ***/
typedef union {
  byte Byte;
  struct {
    byte RSTEN   :1;           /* RST Pin Function Selection */
    byte         :1;
    byte         :1;
    byte OSCOPT0 :1;           /* Selection Bits for Oscillator Option */
    byte OSCOPT1 :1;           /* Selection Bits for Oscillator Option */
    byte         :1;
    byte IRQEN   :1;           /* IRQ Pin Function Selection Bit */
    byte IRQPUD  :1;           /* IRQ Pin Pullup Control Bit */
  } Bits;
  struct {
    byte RSTEN   :1;
    byte         :1;
    byte         :1;
    byte OSCOPT  :2;
    byte         :1;
    byte IRQEN   :1;
    byte IRQPUD  :1;
  } MergedBits;
} CONFIG2STR;
extern volatile CONFIG2STR _CONFIG2 @0x0000001E;
#define CONFIG2 _CONFIG2.Byte
#define CONFIG2_RSTEN _CONFIG2.Bits.RSTEN
#define CONFIG2_OSCOPT0 _CONFIG2.Bits.OSCOPT0
#define CONFIG2_OSCOPT1 _CONFIG2.Bits.OSCOPT1
#define CONFIG2_IRQEN _CONFIG2.Bits.IRQEN
#define CONFIG2_IRQPUD _CONFIG2.Bits.IRQPUD
#define CONFIG2_OSCOPT _CONFIG2.MergedBits.OSCOPT

/*** CONFIG1 - Configuration Register 1 ***/
typedef union {
  byte Byte;
  struct {
    byte COPD    :1;           /* COP Disable Bit */
    byte STOP    :1;           /* STOP Instruction Enable Bit */
    byte SSREC   :1;           /* Short Stop Recovery Bit */
    byte LVI5OR3 :1;           /* LVI 5-V or 3-V Operating Mode Bit */
    byte LVIPWRD :1;           /* Low Voltage Inhibit Power Disable Bit */
    byte LVIRSTD :1;           /* Low Voltage Inhibit Reset Disable Bit */
    byte LVISTOP :1;           /* LVI Enable in Stop Mode Bit */
    byte COPRS   :1;           /* COP Reset Period Selection Bit */
  } Bits;
} CONFIG1STR;
extern volatile CONFIG1STR _CONFIG1 @0x0000001F;
#define CONFIG1 _CONFIG1.Byte
#define CONFIG1_COPD _CONFIG1.Bits.COPD
#define CONFIG1_STOP _CONFIG1.Bits.STOP
```

```
#define CONFIG1_SSREC _CONFIG1.Bits.SSREC
#define CONFIG1_LVI5OR3 _CONFIG1.Bits.LVI5OR3
#define CONFIG1_LVIPWRD _CONFIG1.Bits.LVIPWRD
#define CONFIG1_LVIRSTD _CONFIG1.Bits.LVIRSTD
#define CONFIG1_LVISTOP _CONFIG1.Bits.LVISTOP
#define CONFIG1_COPRS _CONFIG1.Bits.COPRS

/*** TSC - TIM Status and Control Register TSC ***/
typedef union {
  byte Byte;
  struct {
    byte PS0      :1;           /* Prescaler Select Bit */
    byte PS1      :1;           /* Prescaler Select Bit */
    byte PS2      :1;           /* Prescaler Select Bit */
    byte          :1;
    byte TRST     :1;           /* TIM Reset Bit */
    byte TSTOP    :1;           /* TIM Stop Bit */
    byte TOIE     :1;           /* TIM Overflow Interrupt Enable Bit */
    byte TOF      :1;           /* TIM Overflow Flag Bit */
  } Bits;
  struct {
    byte PS       :3;
    byte          :1;
    byte TRST     :1;
    byte TSTOP    :1;
    byte TOIE     :1;
    byte TOF      :1;
  } MergedBits;
} TSCSTR;
extern volatile TSCSTR _TSC @0x00000020;
#define TSC _TSC.Byte
#define TSC_PS0 _TSC.Bits.PS0
#define TSC_PS1 _TSC.Bits.PS1
#define TSC_PS2 _TSC.Bits.PS2
#define TSC_TRST _TSC.Bits.TRST
#define TSC_TSTOP _TSC.Bits.TSTOP
#define TSC_TOIE _TSC.Bits.TOIE
#define TSC_TOF _TSC.Bits.TOF
#define TSC_PS _TSC.MergedBits.PS

/*** TMODH - TIM Counter Modulo Register High ***/
typedef union {
  byte Byte;
  struct {
    byte BIT8     :1;           /* TIM Counter Modulo Bit */
    byte BIT9     :1;           /* TIM Counter Modulo Bit */
    byte BIT10    :1;           /* TIM Counter Modulo Bit */
    byte BIT11    :1;           /* TIM Counter Modulo Bit */
    byte BIT12    :1;           /* TIM Counter Modulo Bit */
    byte BIT13    :1;           /* TIM Counter Modulo Bit */
    byte BIT14    :1;           /* TIM Counter Modulo Bit */
    byte BIT15    :1;           /* TIM Counter Modulo Bit */
  } Bits;
```

```
  struct {
    byte BIT_8    :8;
  } MergedBits;
} TMODHSTR;
extern volatile TMODHSTR _TMODH @0x00000023;
#define TMODH _TMODH.Byte
#define TMODH_BIT8 _TMODH.Bits.BIT8
#define TMODH_BIT9 _TMODH.Bits.BIT9
#define TMODH_BIT10 _TMODH.Bits.BIT10
#define TMODH_BIT11 _TMODH.Bits.BIT11
#define TMODH_BIT12 _TMODH.Bits.BIT12
#define TMODH_BIT13 _TMODH.Bits.BIT13
#define TMODH_BIT14 _TMODH.Bits.BIT14
#define TMODH_BIT15 _TMODH.Bits.BIT15
#define TMODH_BIT_8 _TMODH.MergedBits.BIT_8

/*** TMODL - TIM Counter Modulo Register Low ***/
typedef union {
  byte Byte;
  struct {
    byte BIT0     :1;           /* TIM Counter Modulo Bit */
    byte BIT1     :1;           /* TIM Counter Modulo Bit */
    byte BIT2     :1;           /* TIM Counter Modulo Bit */
    byte BIT3     :1;           /* TIM Counter Modulo Bit */
    byte BIT4     :1;           /* TIM Counter Modulo Bit */
    byte BIT5     :1;           /* TIM Counter Modulo Bit */
    byte BIT6     :1;           /* TIM Counter Modulo Bit */
    byte BIT7     :1;           /* TIM Counter Modulo Bit */
  } Bits;
  struct {
    byte BIT      :8;
  } MergedBits;
} TMODLSTR;
extern volatile TMODLSTR _TMODL @0x00000024;
#define TMODL _TMODL.Byte
#define TMODL_BIT0 _TMODL.Bits.BIT0
#define TMODL_BIT1 _TMODL.Bits.BIT1
#define TMODL_BIT2 _TMODL.Bits.BIT2
#define TMODL_BIT3 _TMODL.Bits.BIT3
#define TMODL_BIT4 _TMODL.Bits.BIT4
#define TMODL_BIT5 _TMODL.Bits.BIT5
#define TMODL_BIT6 _TMODL.Bits.BIT6
#define TMODL_BIT7 _TMODL.Bits.BIT7
#define TMODL_BIT _TMODL.MergedBits.BIT

/*** ADSCR - ADC Status and Control Register ***/
typedef union {
  byte Byte;
  struct {
    byte CH0      :1;           /* ADC Channel Select Bit 0 */
    byte CH1      :1;           /* ADC Channel Select Bit 1 */
    byte CH2      :1;           /* ADC Channel Select Bit 2 */
    byte CH3      :1;           /* ADC Channel Select Bit 3 */
    byte CH4      :1;           /* ADC Channel Select Bit 4 */
    byte ADCO     :1;           /* ADC Continuous Conversion Bit */
```

```
    byte AIEN    :1;           /* ADC Interrupt Enable Bit */
    byte COCO    :1;           /* Conversions Complete Bit */
  } Bits;
  struct {
    byte CH      :5;
    byte ADCO    :1;
    byte AIEN    :1;
    byte COCO    :1;
  } MergedBits;
} ADSCRSTR;

extern volatile ADSCRSTR _ADSCR @0x0000003C;
#define ADSCR _ADSCR.Byte
#define ADSCR_CH0 _ADSCR.Bits.CH0
#define ADSCR_CH1 _ADSCR.Bits.CH1
#define ADSCR_CH2 _ADSCR.Bits.CH2
#define ADSCR_CH3 _ADSCR.Bits.CH3
#define ADSCR_CH4 _ADSCR.Bits.CH4
#define ADSCR_ADCO _ADSCR.Bits.ADCO
#define ADSCR_AIEN _ADSCR.Bits.AIEN
#define ADSCR_COCO _ADSCR.Bits.COCO
#define ADSCR_CH _ADSCR.MergedBits.ADV

/*** ADR - ADC Data Register ***/
typedef union {
  byte Byte;
  struct {
    byte AD0     :1;           /* ADC Data Bit 0 */
    byte AD1     :1;           /* ADC Data Bit 1 */
    byte AD2     :1;           /* ADC Data Bit 2 */
    byte AD3     :1;           /* ADC Data Bit 3 */
    byte AD4     :1;           /* ADC Data Bit 4 */
    byte AD5     :1;           /* ADC Data Bit 5 */
    byte AD6     :1;           /* ADC Data Bit 6 */
    byte AD7     :1;           /* ADC Data Bit 7 */
  } Bits;
  struct {
    byte AD      :8;
  } MergedBits;
} ADRSTR;
extern volatile ADRSTR _ADR @0x0000003E;
#define ADR _ADR.Byte
#define ADR_AD0 _ADR.Bits.AD0
#define ADR_AD1 _ADR.Bits.AD1
#define ADR_AD2 _ADR.Bits.AD2
#define ADR_AD3 _ADR.Bits.AD3
#define ADR_AD4 _ADR.Bits.AD4
#define ADR_AD5 _ADR.Bits.AD5
#define ADR_AD6 _ADR.Bits.AD6
#define ADR_AD7 _ADR.Bits.AD7
#define ADR_AD _ADR.MergedBits.AD
```

Source Code

```
/*** ADICLK - ADC Input Clock Register ***/
typedef union {
  byte Byte;
  struct {
    byte         :1;
    byte         :1;
    byte         :1;
    byte         :1;
    byte         :1;
    byte ADIV0   :1;           /* ADC Clock Prescaler Bit 0 */
    byte ADIV1   :1;           /* ADC Clock Prescaler Bit 1 */
    byte ADIV2   :1;           /* ADC Clock Prescaler Bit 2 */
  } Bits;
  struct {
    byte         :1;
    byte         :1;
    byte         :1;
    byte         :1;
    byte         :1;
    byte ADIV    :3;
  } MergedBits;
} ADICLKSTR;
extern volatile ADICLKSTR _ADICLK @0x0000003F;
#define ADICLK _ADICLK.Byte
#define ADICLK_ADIV0 _ADICLK.Bits.ADIV0
#define ADICLK_ADIV1 _ADICLK.Bits.ADIV1
#define ADICLK_ADIV2 _ADICLK.Bits.ADIV2
#define ADICLK_ADIV _ADICLK.MergedBits.ADIV


/*** TCNT - TIM Counter Register ***/
typedef union {
  word Byte;
  struct {
    byte BIT0    :1;           /* TIM Counter Bit */
    byte BIT1    :1;           /* TIM Counter Bit */
    byte BIT2    :1;           /* TIM Counter Bit */
    byte BIT3    :1;           /* TIM Counter Bit */
    byte BIT4    :1;           /* TIM Counter Bit */
    byte BIT5    :1;           /* TIM Counter Bit */
    byte BIT6    :1;           /* TIM Counter Bit */
    byte BIT7    :1;           /* TIM Counter Bit */
    byte BIT8    :1;           /* TIM Counter Bit */
    byte BIT9    :1;           /* TIM Counter Bit */
    byte BIT10   :1;           /* TIM Counter Bit */
    byte BIT11   :1;           /* TIM Counter Bit */
    byte BIT12   :1;           /* TIM Counter Bit */
    byte BIT13   :1;           /* TIM Counter Bit */
    byte BIT14   :1;           /* TIM Counter Bit */
    byte BIT15   :1;           /* TIM Counter Bit */
  } Bits;
  struct {
    word BIT     :16;
  } MergedBits;
```

```
} TCNTSTR;
extern volatile TCNTSTR _TCNT @0x00000021;
#define TCNT _TCNT.Byte
#define TCNT_BIT0 _TCNT.Bits.BIT0
#define TCNT_BIT1 _TCNT.Bits.BIT1
#define TCNT_BIT2 _TCNT.Bits.BIT2
#define TCNT_BIT3 _TCNT.Bits.BIT3
#define TCNT_BIT4 _TCNT.Bits.BIT4
#define TCNT_BIT5 _TCNT.Bits.BIT5
#define TCNT_BIT6 _TCNT.Bits.BIT6
#define TCNT_BIT7 _TCNT.Bits.BIT7
#define TCNT_BIT8 _TCNT.Bits.BIT8
#define TCNT_BIT9 _TCNT.Bits.BIT9
#define TCNT_BIT10 _TCNT.Bits.BIT10
#define TCNT_BIT11 _TCNT.Bits.BIT11
#define TCNT_BIT12 _TCNT.Bits.BIT12
#define TCNT_BIT13 _TCNT.Bits.BIT13
#define TCNT_BIT14 _TCNT.Bits.BIT14
#define TCNT_BIT15 _TCNT.Bits.BIT15
#define TCNT_BIT _TCNT.MergedBits.BIT

/*** TMOD - TIM Counter Modulo Register ***/
typedef union {
  word Byte;
  struct {
    byte BIT0    :1;           /* TIM Counter Modulo Bit */
    byte BIT1    :1;           /* TIM Counter Modulo Bit */
    byte BIT2    :1;           /* TIM Counter Modulo Bit */
    byte BIT3    :1;           /* TIM Counter Modulo Bit */
    byte BIT4    :1;           /* TIM Counter Modulo Bit */
    byte BIT5    :1;           /* TIM Counter Modulo Bit */
    byte BIT6    :1;           /* TIM Counter Modulo Bit */
    byte BIT7    :1;           /* TIM Counter Modulo Bit */
    byte BIT8    :1;           /* TIM Counter Modulo Bit */
    byte BIT9    :1;           /* TIM Counter Modulo Bit */
    byte BIT10   :1;           /* TIM Counter Modulo Bit */
    byte BIT11   :1;           /* TIM Counter Modulo Bit */
    byte BIT12   :1;           /* TIM Counter Modulo Bit */
    byte BIT13   :1;           /* TIM Counter Modulo Bit */
    byte BIT14   :1;           /* TIM Counter Modulo Bit */
    byte BIT15   :1;           /* TIM Counter Modulo Bit */
  } Bits;
  struct {
    word BIT     :16;
  } MergedBits;
} TMODSTR;
extern volatile TMODSTR _TMOD @0x00000023;
#define TMOD _TMOD.Byte
#define TMOD_BIT0 _TMOD.Bits.BIT0
#define TMOD_BIT1 _TMOD.Bits.BIT1
#define TMOD_BIT2 _TMOD.Bits.BIT2
#define TMOD_BIT3 _TMOD.Bits.BIT3
#define TMOD_BIT4 _TMOD.Bits.BIT4
#define TMOD_BIT5 _TMOD.Bits.BIT5
#define TMOD_BIT6 _TMOD.Bits.BIT6
```

Source Code

```
#define TMOD_BIT7 _TMOD.Bits.BIT7
#define TMOD_BIT8 _TMOD.Bits.BIT8
#define TMOD_BIT9 _TMOD.Bits.BIT9
#define TMOD_BIT10 _TMOD.Bits.BIT10
#define TMOD_BIT11 _TMOD.Bits.BIT11
#define TMOD_BIT12 _TMOD.Bits.BIT12
#define TMOD_BIT13 _TMOD.Bits.BIT13
#define TMOD_BIT14 _TMOD.Bits.BIT14
#define TMOD_BIT15 _TMOD.Bits.BIT15
#define TMOD_BIT _TMOD.MergedBits.BIT


/*
-------------------------------------------------------------------------------
Exceptions in bit names of timer status and control registers (TASC, TSC, TBSC) for
every channel
due to backward compatibility with HC08 AZx versions  */
/*#define TSC0_CHxMAX _TSC0.Bits.CH0MAX
#define TSC0_TOVx _TSC0.Bits.TOV0
#define TSC0_ELSxA _TSC0.Bits.ELS0A
#define TSC0_ELSxB _TSC0.Bits.ELS0B
#define TSC0_MSxA _TSC0.Bits.MS0A
#define TSC0_MSxB _TSC0.Bits.MS0B
#define TSC0_CHxIE _TSC0.Bits.CH0IE
#define TSC0_CHxF _TSC0.Bits.CH0F

#define TSC1_CHxMAX _TSC1.Bits.CH1MAX
#define TSC1_TOVx _TSC1.Bits.TOV1
#define TSC1_ELSxA _TSC1.Bits.ELS1A
#define TSC1_ELSxB _TSC1.Bits.ELS1B
#define TSC1_MSxA _TSC1.Bits.MS1A
#define TSC1_CHxIE _TSC1.Bits.CH1IE
#define TSC1_CHxF _TSC1.Bits.CH1F*/
#endif

/*
** ###################################################################
**
**     This file was created by UNIS Processor Expert 02.90 for
**     the Motorola HC08 series of microcontrollers.
**
** ###################################################################
*/
```

6.2.2 NITRON_MASKS.H

```
/* PortA Register */

enum PTA_MASKS{
PTA0 = 0x01,
PTA1 = 0x02,
PTA2 = 0x04,
PTA3 = 0x08,
PTA4 = 0x10,
PTA5 = 0x20,
AWUL = 0x40
};

/* PortB Register */
enum PTB_MASKS {
PTB0 = 0x01,
PTB1 = 0x02,
PTB2 = 0x04,
PTB3 = 0x08,
PTB4 = 0x10,
PTB5 = 0x20,
PTB6 = 0x40,
PTB7 = 0x80
};

/* Data Direction Register PortA */
enum DDRA_MASKS {
DDRA0 = 0x01,
DDRA1 = 0x02,
DDRA2 = 0x04,
DDRA3 = 0x08,
DDRA4 = 0x10,
DDRA5 = 0x20
};

/* Data Direction Register PortB */
enum DDRB_MASKS{
DDRB0 = 0x01,
DDRB1 = 0x02,
DDRB2 = 0x04,
DDRB3 = 0x08,
DDRB4 = 0x10,
DDRB5 = 0x20,
DDRB6 = 0x40,
DDRB7 = 0x80
};

/* Keyboard Status and Control Register */
enum KBSCR_MASKS{
MODEK       = 0x01,
IMASKK      = 0x02,
ACKK  = 0x04,
KEYF  = 0x08
};
```

Source Code

```
/* Keyboard Interrupt Enable Register */
enum KBIER_MASKS{
KBIE0_ = 0x01,
KBIE1_ = 0x02,
KBIE2_ = 0x04,
KBIE3_ = 0x08,
KBIE4_ = 0x10,
KBIE5_ = 0x20,
AWUIE_ = 0x40
};

/* Configuration Register 2 */
enum CONFIG2_MASKS{
RSTEN       = 0x01,
OSCOPT0 = 0x08,
OSCOPT1 = 0x10,
IRQEN       = 0x40,
IRQPUD      = 0x80
};


/* Configuration Register 1 */
enum CONFIG1_MASKS {
COPD  = 0x01,
STOP  = 0x02,
SSREC       = 0x04,
LVI5OR3 = 0x08,
LVIPWRD = 0x10,
LVIRSTD = 0x20,
LVISTOP = 0x40,
COPRS       = 0x80
};

/* TIM Status and Control Register */
enum TSC_MASKS {
PS0   = 0x01,
PS1   = 0x02,
PS2   = 0x04,
TRST  = 0x10,
TSTOP       = 0x20,
TOIE  = 0x40,
TOF   = 0x80
};

// Timer Prescaler Selection (PS2:PS1:PS0)
enum TIMPrescaler {
Prescaler_by_1  = 0x00,
Prescaler_by_2  = 0x01,
Prescaler_by_4  = 0x02,
Prescaler_by_8    = 0x03,
Prescaler_by_16   = 0x04,
Prescaler_by_32   = 0x05,
Prescaler_by_64   = 0x06
};
```

```
/* ADC Status and Control Register */
enum ADSCR_MASKS{
CH0   = 0x00,
CH1   = 0x01,
CH2   = 0x02,
CH3   = 0x03,
ADCO  = 0x20,
AIEN  = 0x40,
COCO  = 0x80
};

/* ADC Input Clock Register */
enum ADICLK_MASKS {
ADIV0 = 0x20,
ADIV1 = 0x40,
ADIV2 = 0x80
};

// ADC Clock Prescaler Bits (ADV2:ADIV1:ADIV0)
enum ADCPrescaler {
ADCPrescaler_by_1  = 0x00,
ADCPrescaler_by_2  = 0x20,
ADCPrescaler_by_4  = 0x40,
ADCPrescaler_by_8  = 0x60,
ADCPrescaler_by_16 = 0x80
};
```

6.2.3 TYPES.H

```
/*****************************************************************************
                    * Copyright (c) 2003, Motorola Inc.                       *
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :     types.h                                                   *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   In this file, types definitions, defines and macros are   *
*                 implemented.                                              *
*****************************************************************************/

/* New Data type definitions */
typedef unsigned short int  UINT16;       // 16 bit unsigned integer (0, 65535)
typedef signed short int    SINT16;       // 16 bit signed integer   (-32768, 32767)
typedef unsigned char       UINT8;        // 8 bit unsigned byte   (0, 255)
typedef signed char         INT8;         // 8 bit signed byte   (-128, 127)

/* Accelerometer-Adc correspondance */
#define MMA1220                     CH2
#define MMA1260                     CH3

/* Lcd Commands */
#define CURSOR_HOME                 0x02
#define CURSOR_2ND_LINE             0xC0
#define CLEAR_LCD                   0x01

/* Status definition */
#define OK                          TRUE
#define FAIL                        FALSE

/* Min and Max values according to Accelerometers Electric Specs*/
#define MMA1220_MIN_DELTA           51
#define MMA1220_MAX_DELTA           77
#define MMA1260_MIN_DELTA           15
#define MMA1260_MAX_DELTA           46

/* Constants for the data conversion to Gs */
#define MMA1220_CONST               128
#define MMA1260_CONST               61

// General Boolean defines
#define TRUE  1
#define FALSE 0

// Macro definitions
#define EnableInterrupts()          {__asm CLI;}
#define Forever()                   while(1)

/*****************************************************************************
* End types.h                                                                *
*****************************************************************************/
```

6.2.4 LCDDRV.H

```
/*****************************************************************************
* Copyright (c) 2003, Motorola Inc.
*
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :        LcdDrv.h                                               *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   In this file, the LcdDrv prototype functions are          *
*                 implemented.                                              *
*****************************************************************************/

/* Function Prototypes */
void Wait1ms(void);
void Toggle(void);
void WaitNms(UINT8 n);
void InitLcd(void);
void LcdCommand(UINT8 u8Command);
void DisplayString(UINT8 *str);

/*****************************************************************************
* End LcdDrv.h
*
*****************************************************************************/
```

6.2.5 ADC.H

```
/*****************************************************************************
* Copyright (c) 2003, Motorola Inc.
*
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :     Adc.h                                                     *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   In this file, the Adc prototype functions are             *
*                 implemented. Also the global variables are declared.      *
*****************************************************************************/

/* Global Variables */
extern UINT8 near gu8TopAccValue;
extern bool  near gbResetTop;
extern bool  near gbOveraccState;
extern UINT8 near gau8LcdFirstLine[];
extern UINT8 near gu8AccSelect;
extern UINT8 near gu8Wait5msCount;

/* Function Prototypes */
void InitAdc(void);
UINT8 AdcGetValue(UINT8 u8AccChannel);

/*****************************************************************************
* End Adc.h                                                                  *
*****************************************************************************/
```

6.2.6 TIMER.H

```
/*****************************************************************************
                    * Copyright (c) 2003, Motorola Inc.                      *
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :     Timer.h                                                   *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   In this file, the Timer prototype functions are           *
*                 implemented. Also the global variables are declared.      *
*****************************************************************************/

/* Global Variables */
extern UINT8 near gu8TopAccValue;
extern UINT8 near gau8LcdFirstLine[];
extern UINT8 near gu8AccSelect;
extern UINT8 near gu8Wait5msCount;
extern bool  near gbResetTop;
extern bool  near gbOveraccState;

/* Function Prototypes */
void InitTimer(void);

/*****************************************************************************
* End Timer.h                                                                *
*****************************************************************************/
```

6.2.7 KBI.H

```
/*****************************************************************************
                    * Copyright (c) 2003, Motorola Inc.                      *
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :     Kbi.h                                                     *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   In this file, the Kbi prototype functions are             *
*                 implemented. Also the global variables are declared.      *
*****************************************************************************/
/* Global Variables */
extern UINT8 near gu8TopAccValue;
extern bool  near gbResetTop;
extern bool  near gbOveraccState;
extern UINT8 near gau8LcdFirstLine[];
extern UINT8 near gu8AccSelect;
extern UINT8 near gu8Wait5msCount;

/* Function Prototypes */
void InitKbi(void);

/*****************************************************************************
* End Kbi.h                                                                  *
*****************************************************************************/
```

6.2.8 ACCELEROMETER.H

```
/*****************************************************************************
* Copyright (c) 2003, Motorola Inc.
*
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :     Accelerometer.h                                           *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   In this file, the Acceleromoter prototype functions are   *
*                 implemented. Also the global variables are declared.      *
*****************************************************************************/

/* Global Variables */
extern UINT8 near gu8TopAccValue;
extern bool  near gbResetTop;
extern bool  near gbOveraccState;
extern UINT8 near gau8LcdFirstLine[];
extern UINT8 near gu8AccSelect;
extern UINT8 near gu8Wait5msCount;

/* Function Prototypes */
UINT8 AccDataConvGs(UINT8);
void AccTest(UINT8);
bool AccDataFormat(UINT8);

/*****************************************************************************
* End Accelerometer.h                                                       *
*****************************************************************************/
```

6.3 Source Files

6.3.1 MC68HC908QY4.C

```
/*
** ###################################################################
**
**     THIS BEAN MODULE IS GENERATED BY THE TOOL. DO NOT MODIFY IT.
**
**     Filename  : M68HC908QT4.C
**
**     Version   : Driver 01.02
**
**     Compiler  : Metrowerks HC08 C Compiler V-5.0.13
**
**     Date/Time : 05.08.2002, 05:32
**
**     Abstract  :
**
**         This implements an IO devices mapping.
**
**     Settings  :
**
**
**
**     Contents  :
**
**         No public methods
**
**
**     (c) Copyright UNIS, spol. s r.o. 1997-2002
**
**     UNIS, spol. s r.o.
**     Jundrovska 33
**     624 00 Brno
**     Czech Republic
**
**     http      : www.processorexpert.com
**     mail      : info@processorexpert.com
**
** ###################################################################
*/
/* Based on CPU DB MC68HC908QY4, version 2.87.081 */
#include "MC68HC908QY4.h"

volatile ADICLKSTR _ADICLK;    /* ADC Input Clock Register */
volatile ADRSTR _ADR;          /* ADC Data Register */
volatile ADSCRSTR _ADSCR;      /* ADC Status and Control Register */
volatile CONFIG1STR _CONFIG1;  /* Configuration Register 1 */
volatile CONFIG2STR _CONFIG2;  /* Configuration Register 2 */
```

```
volatile DDRASTR _DDRA;         /* Data Direction Register A */
volatile DDRBSTR _DDRB;         /* Data Direction Register B */
volatile KBIERSTR _KBIER;       /* Keyboard Interrrupt Enable Register KBIER */
volatile KBSCRSTR _KBSCR;       /* Keyboard Status and Control Register */
volatile PTASTR _PTA;           /* Port A Data Register */
volatile PTBSTR _PTB;           /* Port B Data Register */
volatile TMODHSTR _TMODH;       /* TIM Counter Modulo Register High */
volatile TMODLSTR _TMODL;       /* TIM Counter Modulo Register Low */
volatile TSCSTR _TSC;           /* TIM Status and Control Register TSC */
volatile TCNTSTR _TCNT;         /* TIM Counter Register */
volatile TMODSTR _TMOD;         /* TIM Counter Modulo Register */
/*
** ###################################################################
**
**     This file was created by UNIS Processor Expert 02.90 for
**     the Motorola HC08 series of microcontrollers.
**
** ###################################################################
*/
```

6.3.2 VECTORS.C

```
/*****************************************************************************
* Copyright (c) 2003, Motorola Inc.
*
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :     vectors.c                                                 *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   In this file is located the Pseudo-Vector table. This
*
* is used for the user mode monitor only.                                   *
*
*
* This table is valid if the user-mode monitor has been programmed into
*
* the DEMO board supplied by Motorola. For normal modes of the device
*
* a different table is used and can be seen in the standard QT/QY demo
*
* applications.
*
*
*
* This table allows the user to control three major aspects of the QY4:
*
*
*
* (1) The default value of the CONFIG1 register
*
* (2) The locations of all the interrupt routines.
*
* (2) The location of the routine to run upon reset.
*
*****************************************************************************/

void _ADC_Interrupt(void);
void _KBD_Interrupt(void);
void _TOF_Interrupt(void);
void _TCH1_Interrupt(void);
void _TCH0_Interrupt(void);
void _IRQ_Interrupt(void);
void _Startup(void);

#define CONFIG1_VAL  0x3D                /* user value for CONFIG1 */

#define CONFIG1_ADR  0xFDEA              /* address of user value for CONFIG1 */
#define JMP_TAB_ADR  0xFDEB              /* address of jump table */
#define JMP_Code     0xCC                /* opcode of JMP instruction */
```

```
typedef void (*tIntFunc)(void);

typedef struct jumpEntry {
   unsigned char jmpIstr;
   tIntFunc      intFunc;
} JumpEntry;

const unsigned char CONFIG1 @CONFIG1_ADR = CONFIG1_VAL;

/* table of JMP instructions to interrupt routines */
const JumpEntry IntJmpTable[] @JMP_TAB_ADR = {
           JMP_Code, _ADC_Interrupt,
           JMP_Code, _KBD_Interrupt,
           JMP_Code, _TOF_Interrupt,
           JMP_Code, _TCH1_Interrupt,
           JMP_Code, _TCH0_Interrupt,
           JMP_Code, _IRQ_Interrupt,
           JMP_Code, _Startup
};
```

6.3.3 MAIN.C

```
/*****************************************************************************
* Copyright (c) 2003, Motorola Inc.                                          *
                     * Motorola Confidential Proprietary
                     *
* ------------------------------------------------------------------------- *
* File name :         main.c                                                 *
* Project name:    Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :    In this file, the MCU configuration, HW and data         *
*                  initialization    and an endless loop is implemented.
*
*****************************************************************************/

#include <startup.h>
#include "types.h"
#include "MC68HC908QY4.h"
#include "Nitron_Masks.h"
#include "LcdDrv.h"
#include "Adc.h"
#include "Timer.h"
#include "Kbi.h"
#include "Accelerometer.h"

/* GLOBAL VARIABLES DEFINITION */
UINT8 near gu8TopAccValue = 0;
UINT8 near gau8LcdFirstLine[] = "MMA1220:    . G  ";
UINT8 near gu8AccSelect;
UINT8 near gu8Wait5msCount;
bool  near gbResetTop = TRUE;
bool  near gbOveraccState = FALSE;

/*****************************************************************************
* void InitHardware(void)                                                    *
*                   Subroutine to initialize CONFIG registers, LCD, PTA, and
*
*                   Accelerometers' Status Latch.                           *
*                                                                           *
* Parameters:      None.                                                    *
*                                                                           *
* Return:          None.                                                    *
*****************************************************************************/
void InitHardware(void) {

      /* Configuration register setup LVI disabled, COP disabled */
   CONFIG1 = LVIRSTD|LVIPWRD|COPD;

      /* Initial set-up of LCD */
   InitLcd();

      /* Port A initialization PTA1 as output for Self Test pin */
   PTA = 0x00;
   DDRA_DDRA1 = 1;

      /* Self-Test pin toggle to reset Status latch on the Accelerometers */
```

```
   PTA_PTA1 = 0;
   WaitNms(50);
      PTA_PTA1 = 1;
      WaitNms(50);
      PTA_PTA1 = 0;

      return;
}

/*************************************************************************
* void main(void)                                                        *
*               This function configures the MCU and its peripherals, tests   *
*               the Accelerometers' Status. It also includes the main endless *
*               loop that is constantly refreshing the LCD with the newest    *
*               top Accelerometer value.                                 *
*                                                                        *
* Parameters:   None.                                                    *
*                                                                        *
* Return:       None.                                                    *
*************************************************************************/
void main(void) {

      UINT8 au8Lcd2ndLine[]="MAX:    . G  --- ";
      UINT8 u8NewAdcValue;
      bool bNewTopValue;

      /* Read data from MMA1220 initially */
      gu8AccSelect = MMA1220;

      InitHardware();
      InitAdc();
      InitTimer();
      InitKbi();

      EnableInterrupts();

      /* Display Welcome message on LCD */
      LcdCommand(CLEAR_LCD);
      LcdCommand(CURSOR_HOME);
      DisplayString(" ACCELEROMETER  ");
      LcdCommand(CURSOR_2ND_LINE);
      DisplayString("EVALUATION BOARD");

      /* Wait for 0.75 seconds */
      WaitNms(250);
      WaitNms(250);
      WaitNms(250);

      /* Hardware test on the Accelerometers    */
      AccTest(MMA1220);
      WaitNms(250);

      AccTest(MMA1260);
      WaitNms(250);

      Forever(){  /*    Endless main loop   */

            /* Get new value from the Accelerometer selected. The data is
```

```
            formatted to Gs. The flag bNewTopValue indicates if there is a
            new top value. */

         u8NewAdcValue = AdcGetValue(gu8AccSelect);

         bNewTopValue = AccDataFormat(u8NewAdcValue);

         /* LCD first line is constantly refreshed with the newest ADC value */

         LcdCommand(CURSOR_HOME);
         DisplayString(gau8LcdFirstLine);

         /* Check if there is a new top value. */

         if(bNewTopValue){


               /* Copy the data already formatted from the first LCD line to
                  the second line. */

               au8Lcd2ndLine[5] = gau8LcdFirstLine[9];
               au8Lcd2ndLine[6] = gau8LcdFirstLine[10];
               au8Lcd2ndLine[8] = gau8LcdFirstLine[12];


               /* Display "!!!" warning in case the new reading is outside
                  spec range. */

               if(gbOveraccState){
                     au8Lcd2ndLine[12] = '!';
                     au8Lcd2ndLine[13] = '!';
                     au8Lcd2ndLine[14] = '!';
               }
               else{
                     au8Lcd2ndLine[12] = '-';
                     au8Lcd2ndLine[13] = '-';
                     au8Lcd2ndLine[14] = '-';
               }

               /* Refresh LCD second line with new value */

               LcdCommand(CURSOR_2ND_LINE);
               DisplayString(au8Lcd2ndLine);
         }
      } /* Forever loop end */

      return;
}

#pragma TRAP_PROC
void _IRQ_Interrupt(void) {
   // Implementation of the IRQ interrupt routine
   return;
}

/*****************************************************************************
* End main.c                                                                 *
*****************************************************************************/
```

6.3.4 LCDDRV.C

```
/*****************************************************************************
                    * Copyright (c) 2003, Motorola Inc.                       *
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :     LcdDrv.c                                                  *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   The LCD control and delay subroutines are implemented in  *
*                 this file.                                                *
*****************************************************************************/

#ifndef  _ACCELEROMETER_H
#define  _ACCELEROMETER_H
#include "types.h"
#include "MC68HC908QY4.h"
#include "Nitron_Masks.h"
#include "LcdDrv.h"
#endif

#define BUSFREQ 0x06

#pragma DATA_SEG SHORT _DATA_ZEROPAGE
UINT8 Counter;

/*****************************************************************************
* void Wait1ms(void)                                                         *
*                 This function waits for 1 ms.                             *
* Parameters:     None.                                                     *
*                                                                           *
* Return:         None.                                                     *
*****************************************************************************/
void Wait1ms(void) {
      asm {
            PSHA
            LDA #BUSFREQ
            DLLoop:
            DBNZA DLSub
            BRA DLDone
            DLSub:
            MOV  #0xFF,Counter
            here1:
            DBNZ Counter,here1
            BRA DLLoop
            DLDone:
            PULA
      }

      return;
}
```

```
/*************************************************************************
* void WaitNms(UINT8 n)                                                  *
*                 This delay function waits for the amout of milisenconds *
*                 specified by the parameter n.                          *
*                                                                        *
* Parameters:     n is an 8 bit unsigned value that specifies the amout of ms
*
*                 that the delay function is going to wait.              *
*                                                                        *
* Return:         None.                                                  *
*************************************************************************/
void WaitNms(UINT8 n) {

   UINT8 i;

   for(i=1;i<=n;i++) Wait1ms();

   return;
}

/*************************************************************************
* void Toggle(void)                                                      *
*                 This function toggles the E line on the LCD. The E line is
*
*                 connected to PTB2.                                     *
*                                                                        *
* Parameters:     None                                                   *
*                                                                        *
* Return:         None.                                                  *
*************************************************************************/
void Toggle(void) {

   PTB |= PTB2;
   PTB &= ~(PTB2);

   return;
}

/*************************************************************************
* void LcdCommand8(UINT8 u8Command)
*
*                Subroutine for sending control bytes to the LCD. This routine*
*                is used with the LCD in 8-bit mode. It is used to set the    *
*                LCD to 4-bit mode.                                      *
*                                                                        *
* Parameters:    u8Command is an 8 bit value for control commands.       *
*                                                                        *
* Return:        None.                                                   *
*************************************************************************/
void LcdCommand8(UINT8 u8Command) {

      PTB = u8Command|(PTB & 0x0F);
   Toggle();
   Wait1ms();

      return;
}
```

```
/*************************************************************************
* void LcdCommand(UINT8 u8Command)                                       *
*               Subroutine for sending control bytes to the LCD. This routine
*
*               sends the 8 bit value in two parts, since the LCD is operating*
*               in 4-bit mode.                                           *
*                                                                        *
* Parameters:   u8Command is an 8 bit value for different control commands.   *
*                                                                        *
* Return:       None.                                                    *
*************************************************************************/
void LcdCommand(UINT8 u8Command) {

      PTB = u8Command & 0xF0;             // MS Nibble
      Toggle();

      PTB = (u8Command & 0x0F)<<4;        // LS Nibble
      Toggle();
      Wait1ms();

      return;
}

/*************************************************************************
* void InitLcd(void)                                                     *
*                 Subroutine to initialize the LCD to 4-bit mode and general  *
*                 settings.                                              *
*                                                                        *
* Parameters:     None                                                   *
*                                                                        *
* Return:         None.                                                  *
*************************************************************************/
void InitLcd(void) {

   /* PTB7:2 = 0, PTB1:0 not part of LCD port, RS, E = 0 */
   PTB = 0x00;
   DDRB = 0xFC;

   /* Wait for Vdd to reach 4.5V */
   WaitNms(15);

   /* 8-bit format used */
   LcdCommand8(0x30);
   WaitNms(4);

   LcdCommand8(0x30);

   LcdCommand8(0x30);

   LcdCommand8(0x20);

   /* From here, 4-bit format used */
   LcdCommand(0x28);                   // 4-bit, 2 lines and font set.

   LcdCommand(0x0C);                   // Display on, cursor off, blinker off
```

```
   LcdCommand(0x06);                    // Cursor movement incremental, no shift

   LcdCommand(0x01);                    // Cursor to home, LCD cleared

   return;
}


/*************************************************************************
* void DisplayString(UINT8 *str)                                         *
*                  A function that displays a string in the LCD at the current *
*                  cursor position. The subroutine sends the characters in the *
*                  string until the NULL character is found.
*
*                  in 4-bit mode.                                        *
*                                                                        *
* Parameters:      *str is the Pointer to the string to be displayed in the LCD*
*                                                                        *
* Return:          None.                                                 *
*************************************************************************/
void DisplayString(UINT8 *str) {

      while ((*str) != '\0'){

            PTB = ((*str) & 0xF0)|(PTB & 0x0F);

            PTB |= 0x08;                              // Set RS Line
            Toggle();

            PTB = (((*str)<<4) & 0xF0)|(PTB & 0x0F);

            PTB |= 0x08;                              // Set RS Line
            Toggle();

            Wait1ms();
            str++;
      }

      return;

}

/*************************************************************************
* End LcdDrv.c                                                           *
*************************************************************************/
```

6.3.5 ADC.C

```
/*****************************************************************************
                    * Copyright (c) 2003, Motorola Inc.                       *
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :     Adc.c                                                     *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   Adc configuration and control routines are included       *
*                 in this file.                                             *
*                                                                           *
*****************************************************************************/

#ifndef  _ADC_H
#define  _ADC_H
#include "types.h"
#include "MC68HC908QY4.h"
#include "Nitron_Masks.h"
#include "Adc.h"
#endif

/*****************************************************************************
* void InitAdc(void)                                                         *
*                 ADC setup, configures the ADC prescaler, single conversion, *
*                 selects channel. The ADC will read the input from on board
*
*                 accelerometers. Channel 2 for MMA1220 accelerometer, channel* *
3 for MMA1260.                                                  *
*                                                                           *
* Parameters:     None.                                                     *
*                                                                           *
* Return:         None.                                                     *
*****************************************************************************/
void InitAdc(void) {
   ADICLK = ADCPrescaler_by_2;      // Configure the ADCLK Register
                                    // ADC input Clock/2 = 3.2 MHz/2

   ADSCR = CH2;                     // Configure the ADSCR Register
                                    // Select AD2 Channel (MMA1220)
                                    // One ADC Conversion
   return;

}
```

Source Code

```
/*************************************************************************
* UINT8 AdcGetValue(UINT8 u8AccChannel)                                  *
*               This function generates a single conversion and waits until   *
*               it is completed. The result is returned as an 8 bit variable.
*
*               This loop is ended in case the Adc takes more than 10 attempts*
*               to obtain the converted value.                           *
*                                                                        *
* Parameters:   u8AccChannel is an 8-bit variable that indicates the channel  *
*               to be measured Channel 2 for MMA1220 accelerometer, channel 3
*
*               for MMA1260 accelerometer.                               *
*                                                                        *
* Return:         The new 8-bit Adc value obtained.                      *
*************************************************************************/
UINT8 AdcGetValue(UINT8 u8AccChannel){

      UINT8 u8AttemptCount = 10;    // Limit amount of attempts.
      UINT8 u8AdcValue;

      ADSCR = u8AccChannel;         // Set Channel and generates new conversion

      while(!(ADSCR & COCO) && (u8AttemptCount>0)) //wait for conversion complete
            u8AttemptCount--;

      u8AdcValue = ADR;

      return u8AdcValue;
}


#pragma TRAP_PROC
void _ADC_Interrupt(void) {
   // Implementation of the ADC interrupt routine
   return;
}

/*************************************************************************
* End Adc.c                                                              *
*************************************************************************/
```

6.3.6 TIMER.C

```
/*****************************************************************************
* Copyright (c) 2003, Motorola Inc.
*
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :        Timer.c                                                *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   Timer configuration routines and Timer Overflow interrupt   *
*                 service routines are included in this file.               *
*****************************************************************************/

#ifndef  _TIMER_H
#define  _TIMER_H
#include "types.h"
#include "MC68HC908QY4.h"
#include "Nitron_Masks.h"
#include "Timer.h"
#endif

/*****************************************************************************
* void InitTimer(void)                                                       *
*               This function Turns on Timer Channel 1 such that it counts    *
*               between 0 and $FF.                                          *
*               An overflow interrupt occurs every 256 * 64 = 16384 bus cycles*
*               (5.12ms @3.2MHZ bus).                                       *
*                                                                           *
* Parameters:   None.                                                       *
*                                                                           *
* Return:       None.                                                       *
*****************************************************************************/
void InitTimer(void) {

      TSC = (TOIE | TRST | TSTOP | Prescaler_by_64);
                                    // Enable overflow interrupt
                                    // Timer 1 - Reset and Stopped.
                                    // Clicks once every 64 Bus Cycles

      TMODH = 0x00;                      // Timer counts from 0-$FF
      TMODL = 0xFF;                      // Overflows every 16384 Bus Cycles

      TSC_TSTOP = 0;                     // Start the timer

      return;

}

/*****************************************************************************
* void _TOF_Interrupt(void)                                                  *
*                 This is the Timer Overflow interrupt service routine. The   *
*                 timer rolls over every 256T counts of the timer (TMOD=$FF),
*
```

```
*                 which correspond to 256 * 64 = 16384 bus cycles
*
*                 (5.12ms @3.2MHZ bus).                                          *
*                 This overflow routine period is used as a time-base for the *
*                 debounce of the push-buttons on the board.                     *
*                                                                                *
* Parameters:     None.                                                          *
*                                                                                *
* Return:         None.                                                          *
*********************************************************************************/
#pragma TRAP_PROC
void _TOF_Interrupt(void) {

      UINT8 kbiportdata;

      TSC;                                        // Procedure to Acknowledge Inter-
rupt
      TSC_TOF = 0;                                // Read TSC and Clear Overflow flag

      // Button Debounce
      if(KBSCR_IMASKK == 1){                      // Keboard interrupts masked?
            if (--gu8Wait5msCount == 0 ){         // 51.2ms elapsed?
                  kbiportdata = (PTA & (KBIE2_|KBIE3_));

                  if(~kbiportdata == ~KBIE2_){  // If S2 is pressed switch Acc Read
                        if(gu8AccSelect == MMA1220) gu8AccSelect = MMA1260;
                        else  gu8AccSelect = MMA1220;
                        gbResetTop = TRUE;
                  }
                  else if(~kbiportdata == ~KBIE3_){
                        gu8TopAccValue = 0;     // If S1 is pressed reset TopValue
                  }

                  KBSCR_ACKK = 1;         // Acknowledge any pending interrupts
                  KBSCR_IMASKK = 0;       // Re-enables Keyboard Interrupts


            }

      }
      return;
}

#pragma TRAP_PROC
void _TCH1_Interrupt(void) {
   // Implementation of the TCH1 interrupt routine
   return;
}

#pragma TRAP_PROC
void _TCH0_Interrupt(void) {
   // Implementation of the TCH0 interrupt routine
   return;
}

/*********************************************************************************
* End Timer.c                                                                    *
*********************************************************************************/
```

6.3.7 KBI.C

```
/*****************************************************************************
* Copyright (c) 2003, Motorola Inc.
*
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :     Kbi.c                                                     *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   Kbi configuration and interrupt service routine are       *
*                 included in this file.                                    *
*                                                                           *
*****************************************************************************/

#ifndef  _KBI_H
#define  _KBI_H
#include "types.h"
#include "MC68HC908QY4.h"
#include "Nitron_Masks.h"
#include "Kbi.h"
#endif


/*****************************************************************************
* void InitKbi(void)                                                        *
*              This function  Configures the KBI Module to accept interrupts
*
*              on PTA2 and PT3 that are connected to S1 and S2 pushbuttons of
*
*              the board.                                                   *
*                                                                           *
* Parameters:  None.                                                        *
*                                                                           *
* Return:      None.                                                        *
*****************************************************************************/
void InitKbi(void) {

// The following procedure is to prevent False Interrupts at initialization

      KBSCR_IMASKK = 1;                   // Mask Keyboard interrupts
      KBIER  = KBIE2_|KBIE3_;             // Enables pin 2 and 3 of KBI
      KBSCR_ACKK   = 1;                   // Clear any false interrupts
      KBSCR_IMASKK = 0;                   // Unmask Keyboard interrupts

// END Avoidance of False Interrupts

      KBSCR = 0;              // Configures KBI Status & Control Register
                              // IMASKK=0: Clears KBI Mask Bit (Enable Ints)
                              // MODEK=0:  Interrupt requests on Falling Edge Only
      return;
}
```

Source Code

```
/*****************************************************************************
* void _KBD_Interrupt(void)                                                  *
*                 This is the Keyboard interrupt service routine. This routine*
*                 sets the debounce time to 51.2 ms and masks the keyboard    *
*                 interrupts.                                                *
*                                                                            *
* Parameters:     None.                                                      *
*                                                                            *
* Return:         None.                                                      *
*****************************************************************************/
#pragma TRAP_PROC
void _KBD_Interrupt(void) {

      gu8Wait5msCount = 10;         // Initializes gu8Wait5msCount for debouncing

   KBSCR |= (ACKK | IMASKK);        // Acknowledge KB Interrupts
                                    // Mask KB interrupts during 51.2ms (10 *
5.12ms)
      return;
}

/*****************************************************************************
* End Kbi.c                                                                  *
*****************************************************************************/
```

6.3.8 ACCELEROMETER.C

```
/*****************************************************************************
                    * Copyright (c) 2003, Motorola Inc.                       *
* Motorola Confidential Proprietary
*
* ------------------------------------------------------------------------- *
* File name :     Accelerometer.c                                           *
* Project name:   Low-Cost Accelerometer Evaluation Board                   *
* ------------------------------------------------------------------------- *
*                                                                           *
* Description :   In this file the Accelerometers' test function is         *
*                 implemented along the subroutines to format the data      *
*                 acquired from the accelerometers to be displayed in
*
*                 the LCD.                                                  *
*****************************************************************************/
#ifndef  _ACCELEROMETER_H
#define  _ACCELEROMETER_H
#include "types.h"
#include "MC68HC908QY4.h"
#include "Nitron_Masks.h"
#include "Accelerometer.h"
#include "LcdDrv.h"
#include "Adc.h"
#endif

/*****************************************************************************
* UINT8 AccDataConvGs(UINT8 u8RawValue)                                      *
*                 Subroutine convert the data acquired from the Accelerometers*
*                 to G units. This depends on the Accelerometer actually    *
*                 selected.                                                 *
*                                                                           *
* Parameters:     u8RawValue is an 8-bit value that contains the data to be *
*                 converted to Gs.                                          *
*                                                                           *
* Return:         8-bit value containing the data formatted to Gs.          *
*****************************************************************************/
UINT8 AccDataConvGs(UINT8 u8RawValue) {

      UINT16 u16FormattedValue;

      if(gu8AccSelect == MMA1220) {
            u16FormattedValue = u8RawValue*100;             //conversion to Gs for
MMA1220
            u16FormattedValue /= MMA1220_CONST;
            }
      else if(gu8AccSelect == MMA1260){
            u16FormattedValue = u8RawValue*10;        //conversion to Gs for MMA1260
            u16FormattedValue /= MMA1260_CONST;
            }
      return((UINT8)u16FormattedValue);
}

/*****************************************************************************
```

```
* void AccTest(UINT8 u8AccUnderTest)                                          *
*              Subroutine to test the Accelerometers status. The routine sets
*
*              the Self-test pin to high to test the Accelerometer function   *
*              to be within specs. This is tried a maximum amount of 5 times.
*
*              The subroutine reflects the output to the LCD.                 *
*                                                                             *
* Parameters:  u8AccUnderTest is an 8-bit value that specifies the            *
*              Accelerometer to be tested.                                    *
*                                                                             *
* Return:      None                                                           *
*******************************************************************************/
void AccTest(UINT8 u8AccUnderTest){

      bool bTestStatus = FAIL;
      UINT8 u8SelfTestCount = 0;
      UINT8 u8Value1 = 0;
      UINT8 u8Value2 = 0;

      LcdCommand(CLEAR_LCD);
      LcdCommand(CURSOR_HOME);

      DisplayString("Testing ");                        // Display acc under test

      if(u8AccUnderTest == MMA1220){
            DisplayString("MMA1220 ");
            }
      else if(u8AccUnderTest == MMA1260){
            DisplayString("MMA1260 ");
      }

      do {
         PTA_PTA1 = 0;                                                        //
Self-test pin low
         WaitNms(25);

            u8Value1 = AdcGetValue(u8AccUnderTest);    // Generate new ADC conversion
                                                       // Acc output without Self
Test

         PTA_PTA1 = 1;                                 // Self-test pin high
         WaitNms(25);

         u8Value2 = AdcGetValue(u8AccUnderTest);       // Generate new ADC conversion
                                                       // Acc output with Self Test

         if(u8Value2 > u8Value1){                      // Output change is positive?
            u8Value2 -= u8Value1;

            /* Check if change within accelerometer electrical specs */
            if((u8AccUnderTest == MMA1220) && (u8Value2 > MMA1220_MIN_DELTA) &&
  (u8Value2 < MMA1220_MAX_DELTA) ){
                        bTestStatus = OK;
                        break;
```

```
            }
            if((u8AccUnderTest == MMA1260) && (u8Value2 > MMA1260_MIN_DELTA) &&
  (u8Value2 < MMA1260_MAX_DELTA) ){
                        bTestStatus = OK;
                        break;
            }

         }

         u8SelfTestCount++;                        // Increase the testing counter

      }while(u8SelfTestCount<5);


      PTA_PTA1 = 0;
// Self test pin low

      LcdCommand(CURSOR_2ND_LINE);
      DisplayString("Self-Test ");
      if (bTestStatus == OK) {
         DisplayString("OK");                      // Self test successful
         }
      else {
         DisplayString("Failed");                  // Self-test unsuccessful
         }                                         // after 5 tries

      WaitNms(250);                                // Wait for 0.5 seconds
      WaitNms(250);

      return;
}

/*************************************************************************
* bool AccDataFormat(UINT8 u8AccData)                                    *
*               This function formats the data acquired from the Adc to the  *
*               the LCD Message Buffer. The data is formatted to G units and *
*               positioned in the Message Buffer to be displayed in the LCD
*
*                                                                        *
* Parameters:   u8AccData is an 8-bit value with the data acquired from the  *
*               Adc.                                                     *
*                                                                        *
* Return:       bNewTopFlag is a boolean variable that reflects if there is a*
*               new top value.                                           *
*************************************************************************/
bool AccDataFormat(UINT8 u8AccData){

      UINT8 u8BufferPosition;
      bool bNewTopFlag = FALSE;


// Format  ± 0.0 G
      if(u8AccData < 128){                         // Data is negative
         gau8LcdFirstLine[9] = '-';
         u8AccData = 128 - u8AccData;              // Remove offset
```

Source Code

```
        }
    else{                                       // Data is positive
        gau8LcdFirstLine[9] = ' ';
        u8AccData = u8AccData - 128;            // Remove offset
        }

    if((u8AccData > gu8TopAccValue) || (gbResetTop)){
        gu8TopAccValue = u8AccData;             // Update Top Value
        bNewTopFlag = TRUE;
        gbResetTop = FALSE;
        gbOveraccState = FALSE;
        }

    u8AccData = AccDataConvGs(u8AccData);       // Data converted to Gs

    if(gu8AccSelect == MMA1220){                // Change Acc indicator to MMA1220
        gau8LcdFirstLine[5] = '2';
        if(u8AccData>80) gbOveraccState = TRUE;
    }

    if(gu8AccSelect == MMA1260){                // Change Acc indicator to MMA1260
        gau8LcdFirstLine[5] = '6';
        if(u8AccData>15) gbOveraccState = TRUE;
        }

    if(u8AccData<10){
        gau8LcdFirstLine[10]='0';               // Add 0 in case of G<1
        }

    u8BufferPosition = 12;                      // BCD Converson to Message Buffer
    do{
        gau8LcdFirstLine[u8BufferPosition--]= (u8AccData%10) + '0';
        u8BufferPosition--;
        }while((u8AccData /= 10) > 0);

    return(bNewTopFlag);                        // Return Top Value Flag
}

/*****************************************************************************
* End Accelerometer.c                                                        *
*****************************************************************************/
```

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:
Motorola Literature Distribution
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

JAPAN:
Motorola Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

ASIA/PACIFIC:
Motorola Semiconductors H.K. Ltd.
Silicon Harbour Centre
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
852-26668334

HOME PAGE:
http://motorola.com/semiconductors

DRM054/D
Rev. 0
12/2003