

1 General

This document explains in detail the demonstration software example found on the starterkit disk in the directories `cd:\GZ16STK_CodeWarriorExamples` and `cd:\GZ16STK_CosmicExamples`.

This demonstration can be compiled with the CodeWarrior special edition rsp. The Cosmic Lite 4 K Version.

It demonstrates basic CAN I/O.

The SBC is configured for debug mode via a simple SPI driver.

The H-Bridge is used to control a small DC motor with various speed via PWM and changing rotational direction.

2 Open the Examples

Copy the complete demo directory files from CD to your harddisk. Un-writeprotect the files using the properties dialog in windows explorer.

2.1 Metrowerks

Navigate to the copied example and double-click on the *.mcp (Metrowerks project file) or open the Metrowerks IDE and navigate in the open dialog to the mcp-file.

2.2 Cosmic

Start the IDE and set the working directory (setup menu) to your copied files. Navigate to the copied example in the Project load dialog.

Change the name of the project working directory to the name of the example directory on your disk. If you encounter problems with this, see the IDE documentation, topic managing projects.

3 Simple CAN

This chapter describes the features of the uC software that is loaded to the boards in the delivery state. Loaded is the hex-file from the Metrowerks example. To example works the same with the Cosmic sources.

To reload this software (after testing other demos), use our tool HC08-ISP (included on the starterkit disk). The HEX-File for this application is included in the directory `GZ16STK_CodewarriorExamples\SimpleCAN\bin`.

The sources are also available.

3.1 Control of DC Motor via CAN

3.1.1 General

This setup demonstrates the control of a DC motor via CAN or SCI.

The DC motor is connected to the outputs of the H-bridge on board #2.

If you only have one board, a CAN analyzer tool with additional CAN Hardware (PC plug-in board or external device) can also be used. CAN Baud rate is 50kBaud.

Board #1 and #2 are connected via CAN bus.

Board #1 is additionally connected to the PC via SCI using the MONIF08-LC. The PC sends single character commands using a terminal program (e.g. Hyper Terminal). These commands will generate a single CAN message.

Commands defined are:

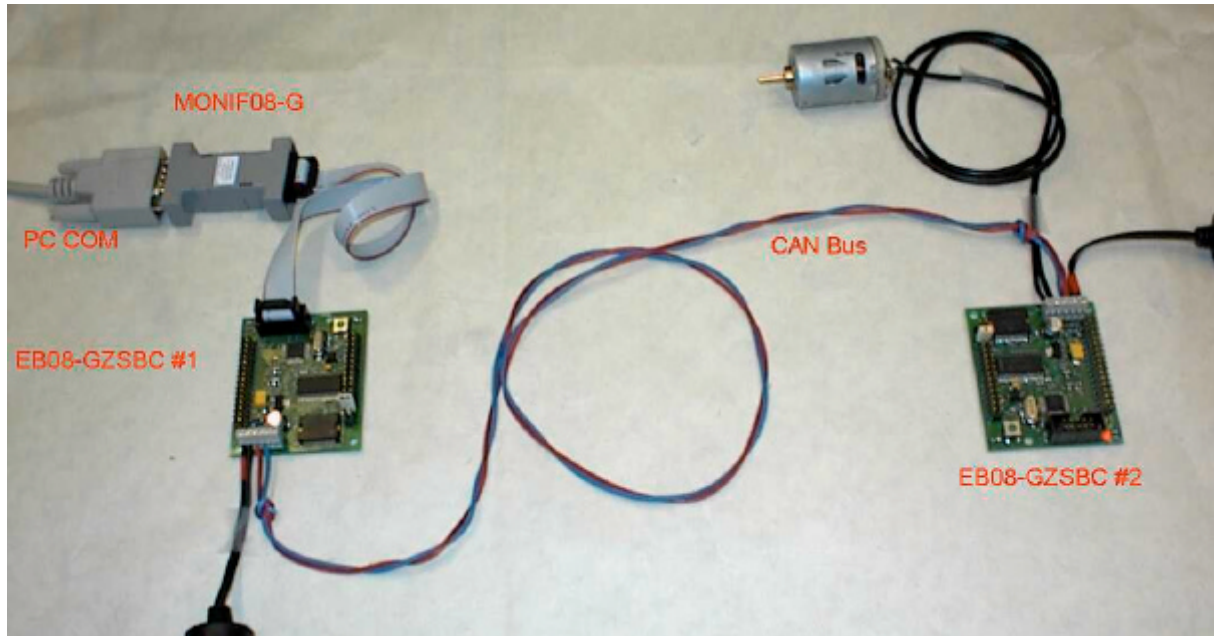
Shortcut Character	CAN Msg	Meaning
0	00000100:0:	Motor Stop
1	00000101:1:30	Motor forward, speed 0x30
2	00000101:1:E0	Motor forward, speed 0xE0
3	00000102:1:80	Motor reverse, speed 0x80
4	00000102:1:FF	Motor reverse, speed 0xFF (full speed)
6	00000120:0:	LED D3 off
7	00000121:0:	LED D3 on

On Board #2 the GZ16 software parses the incoming CAN messages and controls the board hardware accordingly.

Board #2 can additionally be connected to another PC COM interface with a second MONIF08 (or RS232-TLL Level Converter) . With this setup the incoming CAN messages can be monitored and messages can be send forth and back.

Alternatively an CAN analyser tool can be used to send the messages.

3.1.2 Hardware Setup



X9 Screw Terminal

Label	+ Bat	- Bat	CAN H	CAN L	HBR A	HBR B
Connect to	Supply plus 9..12V	Supply GND	CAN H	CAN L	H-Bridge Out, DC Motor	H-Bridge Out, DC Motor

Power Supply: brown wire is minus, white wire is plus

CAN Termination

To set the CAN termination connect X2 Pin 5 and Pin 6 with a jumper.

3.1.3 Software Setup

PC terminal program

Configure for the appropriate COM port, Baud rate 38400 Bd, 8 data bits, no parity, no protocol.

EB08-GZSBC #1

Metrowerks: load\SimpleCAN\bin\GZ16_ASM-C-Mixed.ABS.S19

Cosmic: Load simpleCAN.s19

EB08-GZSBC #2

Cosmic: Load simpleCAN.s19

Metrowerks: load\SimpleCAN\bin\GZ16_ASM-C-Mixed.ABS.S19

3.2 Control via SCI

With the same setup, but without CAN bus and only one board, the motor can be controlled via SCI.

Commands defined are:

Shortcut Character	Meaning
S or Space bar	Motor Stop
F	Motor forward at actual speed
R	Motor reverse at actual speed
+	speed up
-	Slow down

3.3 Debugging the application in Monitor Mode

Debugging is not so easy, since the PLL is used so the baud rate in Monitor changes. The PLL is necessary to have appropriate processing power for the timer interrupt routine to catch every CAN bus message at 50Kbaud. If the Timer interrupt is slowed down (routine TIMinit in TIM.c) , setting the PLL can be omitted (in smpleCAN.c, function PLLon(5) is to be left out) and debugging is possible. The SCI initialization (baud rate) is also to be changed.

3.4 Software inside Documentation

Features:

- monitoring the CAN bus via SCI
- generation messages on CAN as reaction on different events (push button, receipt on an other message, SCI single character command)
- Control of the DC motor

3.4.1 General

The demonstration application is located inside the main() routine.

After initialization of all necessary modules the program loops endlessly (main control loop) while the TBMisr does cyclic tasks.

3.4.2 Initialization

The PLL is set for factor 4.

The CAN is set for 50kbaud at 5MHz crystal clock.

The CAN ID acceptance filter is set to accept every message.

The SCI (ESCI on GZ16) is set to 38400Bd, 8 Bit, 1 Stop Bit , no Parity at 5 MHz bus clock (PLL dependent on GZ16 !).

The ports for the LEDs are set as outputs and the LEDs are switched on.

The Pull Up for the Push button is enabled.

The SPI is initialized and messages to configure an optional SBC-chip (MC33989) to debug mode are sent via SPI.

The PWM is initialized.

3.4.3 Main control loop

Inside this loop different tasks are performed:

Every second:

The state of a LED connected to PTC2 is changed.

If PTA2 is pulled low (e.g. with the Push-Button on the EB08-GZSBC) a predefined CAN message is send.

Upon receive of a CAN message:

The Message is displayed on SCI and answered with ID0 incremented by one.

The Message is parsed and the Motro or LED is set up accordingly (see above for defined protocol messages)

Upon receiving a character '0'..'9' via SCI:

A predefined CAN message (different for every number) is sent.

Upon receiving a character out of R, S, F, +, - via SCI: Motor control (speed or direction) is changed

Every 100ms: the state machine for DC motor control is executed.

This state machine stops the motor for 100ms before reversing to prevent damage or high current spikes.

4 Programming the Board

Programming can be done using our flash programming software or with the build in flash tool of the Metrowerks debugger.

4.1 Setup Software HC08_ISP

Install the software on the PC. The software does no changes to the registry and can therefore be deleted after evaluation without any problems.

The software installation process does not make an entry to the start menu.

To start the software, generate a shortcut on your desktop or simply double click onto the



HC08_ISP_En.exe icon in the installation directory.

For first setup, make the following entries to the program. This has to be done only once as all settings are remembered if the project is saved.

4.1.1 Target configuration

Interface: MONIF08-LC

Micro Controller: 68HC908GZ16

Reset Mode: automatic

POR delay: 1000

Power up delay: 500

Security Bytes: Make one entry with FF-FF-FF-FF-FF-FF-FF. If the controller is already programmed and the .S19-file of the program is available, select „from file“ in another line.

Checkmark FF-FF.. if security bytes are unknown, or the checkmark the appropriate sec. bytes line.

4.1.2 Communication

Port Number: Number of COM port to which the MONIF08 is connected

Max Baud Rate: 38400

Target Crystal Frequency: 5000000 (frequency in Hz, must match the frequency of the crystal soldered to the board)

PTC3: high (if not solder jumper SJ2 is closed)

Click on „calculate Baud rates“ once to set up the baud rate.

4.1.3 Source

Click on select to choose a file, e.g. „GZ_CANmon1.s19

Checkmark „load“.

If for testing different files shall be loaded, more than one file can be selected, but only checkmark „load“ on the one you actually want.

If more than one „load“ is checkmarked, the files will be merged.

4.1.4 Options

For the start no entries are necessary here. See software documentation.

It may be sensible to checkmark “AutoProgram Options” - “Flash – Program” and “Other – Update Sec Bytes” to use Auto Program. Auto program is explained below.

4.1.5 Target info

No entries are necessary (and not even possible) here. This is just an information about how the memory ranges etc. are configured.

4.1.6 Buffer

On this tab it is possible to read out the data from the micro controller and to write them to file.

For the start no entries are necessary here.

4.1.7 Save the Project

Click on „project“, then „save as“ in the menu line. Select a proper name, e.g. „GZ16_test“ or something you want. The last used project comes up on the next start of the software.

4.2 Programming with HC08_ISP

Activate the proper security bytes on the target configuration tab.

If the controller is blank, select „FF-FF-FF...“

Click on the blue „P“ in the short cut line or choose „Flash Program“ from the „Action“ menu.

4.2.1 Security Byte Issues

If security bytes are unknown first delete the flash completely. This is done with the action „FlashDelete“ or with clicking the blue „D“ in the short cut line.

After this select blank security bytes (FF-FF...) as the deleted state is all FF.

Now it very important to do a power on reset by completely removing the power from the board (disconnect mains adapter from mains for about 10 seconds).

After this, a regular program action can be done.

4.2.2 Additional Power down cycle after programming of blank or deleted parts

After programming blank or deleted parts (reset vector was FFFF) a power down cycle is needed to start the loaded program. Without power down the controller remembers to be in “forced monitor mode” and always enters the monitor after reset.

4.2.3 Auto Program with automatic Security Byte Update

Set up the “Options” tab as shown in 4.1.4 .

Auto program is started clicking on the red-blue „AP“ in the short cut line or with the action „FlashAutoProgram“ (F5).

The micro controller is programmed and after this, the first line of security bytes in the „Target Configuration“ tab is updated. With this feature the security bytes in HC08_ISP always match with the last programmed ones and it is easy to enter the monitor mode without the need for POR-cycles.