Freescale Semiconductor
Application Note

Document Number: AN3381
Rev. 0, 11/2006

Using SMAC with the MC9S08QD4 MCU

by: Juan Cazares Blanco
RTAC Guadalajara

1 Introduction

This application note explains how to modify any Simple Media Access Controller (SMAC) application created by the BeeKit Wireless Connectivity Toolkit^{TM} to run the application on the MC9S08QD4 microcontrollers. While this application note is useful to import any BeeKit Wireless Connectivity Toolkit application to other microcontrollers, the target of this application note is the MC9S08QD4 device.

2 Overview

The BeeKit Wireless Connectivity Toolkit is a comprehensive codebase of wireless networking libraries, application templates, and sample applications. The BeeKit Graphical User Interface (GUI), part of the BeeKit Wireless Connectivity Toolkit, allows users to create, modify, and update various wireless networking implementations. This application note describes the steps to import these applications to MC9S08QD4 microcontrollers. The BeeKit Wireless Connectivity Toolkit allows selecting demo applications from a list of pre-loaded applications that you can configure. The configure options include: select number of serial port,

Contents

number of RF channel to transmit and receive, power transmission, bootloader, etc. After the application is configured, an XML file is created by the BeeKit Wireless Connectivity Toolkit and it must be imported to the CodeWarrior^{TM} development tool to generate the source code. This source code can be modified at any time in the CodeWarrior software. Using the BeeKit Wireless Connectivity Toolkit, a solution is created, related projects are added to the solution, and software components are defined in the projects. Then, users must export the solution from BeeKit for deployment in the CodeWarrior Integrated Development Environment (IDE).

3 Where to use Low-End Microcontrollers

The MC9S08QD4 is a cost-effective, low-end microcontroller pin-to-pin compatible with MC9RS08KA2 and MC9S08QG4 microcontrollers. It is a perfect choice for networking applications where performance and speed are not critical, such as in input/output (GPIO) lighting, switching, access control, air control, etc. However, it performs perfectly with some powerful microcontrollers such as i.MX, ColdFire, and MC9S08GT60, which can be used in nodes where the expected performance is high.

4 Functional Description

4.1 Hardware Configuration

The MC9S08QD4 microcontroller has six general-purpose pins. For the purpose of using SMAC, these pins communicate with the transceiver MC1319x. The transceiver has seven GPIO pins; two control the transceiver communication and the rest to sense inputs or to control some external component (LEDs, relays, etc.). Table 1 describes the necessary connections to get the transceiver in line and transmit and receive data using the MC9S08QD4 microcontroller. See Figure 1 for microcontroller and transceiver connections.

Table 1. Pin Function Description

Pin	MCU Pin Name	Type	Pin	Transceiver Pin Name	Type
8	PTA0/KBI1P0	Digital output	19	Chip enable	Digital input
7	PTA1/KBI1P1	Digital output	12	Reset	Digital input
6	PTA2/KBI1P2	Digital input	18	MISO	Digital output
5	PTA3/KBI1P3	Digital output	17	MOSI	Digital input
2	PTA4/BKGD	Digital output	16	SPI clock	Digital clock input
1	PTA5/RST/IRQ	Digital input	20	IRQ	Digital output

Two more signals needed for the transceiver to work properly are RXTXEN and ANTCTL. RXTXEN switches the mode of operation (reception, transmit, doze, hibernate or IDLE). ANTCTL manages an antenna switch device. Designs that have the same antenna to transmit and receive use the antenna switch. If the design has a dual antenna, one to transmit and the other to receive, the ANTCTL signal is not needed.

Because all the pins on the microcontroller manage other signals, GPIO1 and GPIO2 pins provided by the transceiver manage RXTXEN and ANTCTL signals, respectively. Table 2 and Table 3 describe the connections needed to control RXTXEN and ANTCTL.

Table 2. RXTXEN Connection

Pin	Transceiver Pin Name	Type	Pin	Transceiver Pin Name	Type
11	GPIO1	Digital output	13	RXTXEN	Digital input

Table 3. ANTCTL Connection

Pin	Transceiver Pin Name	Type	Pin	Antenna Switch Pin Name	Type
10	GPIO2	Digital output	4	VCONT	Digital input

Figure 1 shows the microcontroller-transceiver connections described in Table 1, Table 2, and Table 3.

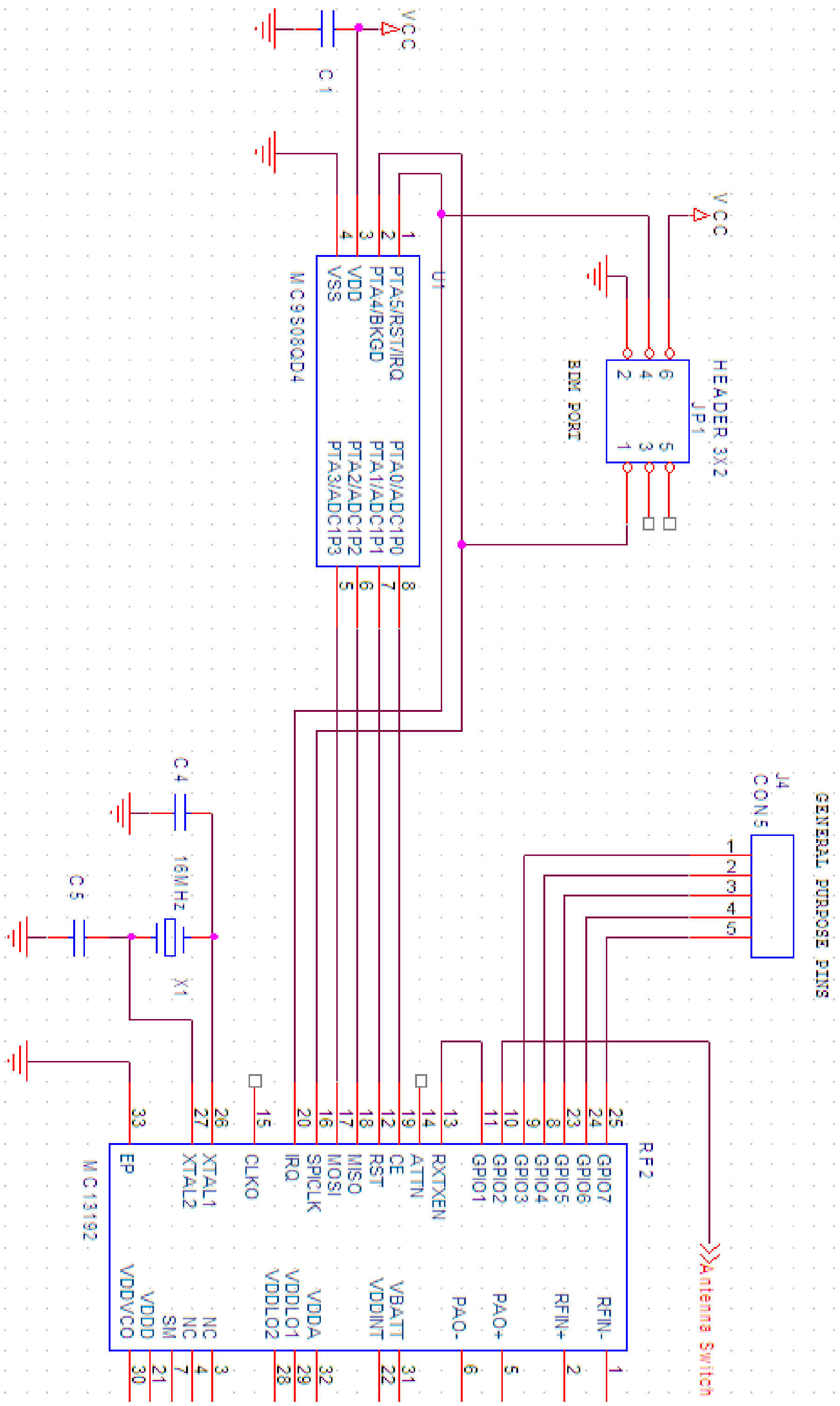


Figure 1. Microcontroller and Transceiver Connections

5 Requirements for CodeWarrior to use the MC9S08QD4 Microcontroller

Before starting a new project, the following software must be installed:

- CodeWarrior development tool for HC08 version 5.1 or later
- Update and patch for the MC9S08QD4, which can be found in www.freescale.com by typing CW08 V5.1 QD4 Service Pack in the search engine labeled Enter Keyword

NOTE

Download the CodeWarrior software updates and patches from www.freescale.com. Go to Products, select CodeWarrior Development Tools, select Downloads from the left navigation bar, and go to Updates and Patches.

6 Create a BeeKit Wireless Connectivity Toolkit Application

The MC9S08QD4 microcontroller does not have an SPI peripheral. Therefore, the following considerations must be taken to use the MC9S08QD4 with BeeKit Wireless Connectivity Toolkit:

- Selected platform must always be MC1319XSARD.
- LCD option must always be disabled.
- Security option must always be disabled.
- OTAP option must always be disabled.
- Embedded bootloader option must always be disabled.

See the BeeKit Wireless Connectivity ToolKit User's Guide (BKWCTKUG) for more details.

7 Import a BeeKit .XML Solution File to the CodeWarrior Tool

After BeeKit successfully creates and exports a solution, there are different ways to use individual projects. You can, as defined in Beekit, compile and download to the user-selected platform. Also, you can use the BeeKit imported project as a template for more complex software development specific to user requirements within the CodeWarrior IDE. The solutions are output as XML files.

8 Create a New Project for MC9S08QD4 in the CodeWarrior tool

All SMAC projects have the same format: one folder for all SMAC files and the other for user applications. This format is recognized by the new CodeWarrior project. A new project must be created using the CodeWarrior tool’s project wizard option and the derivative chosen must be MC9S08QD4. After the project is created, the folders inside it must be modified to add the SMAC files.

9 Edit the MC9S08QD4 Project

Perform the following steps to integrate the SMAC files into the project created for the MC9S08QD4.

1. Add three new folders to the project by creating groups. To create groups, right click on the project window and pick Create Group. Folders names must be: shared, drivers, and SMAC. See Figure 2.

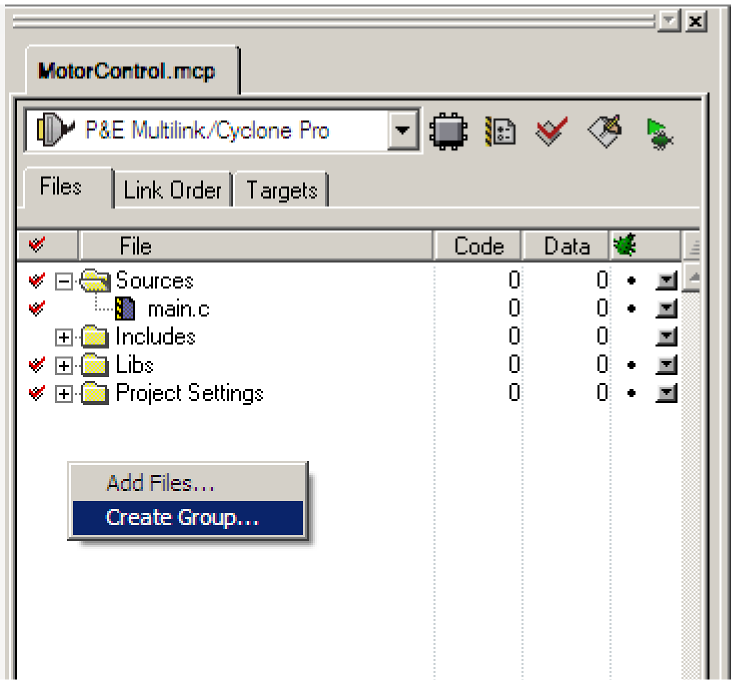


Figure 2. Creating Groups

2. Move the drivers and SMAC folders into the folder called shared. See Figure 3.

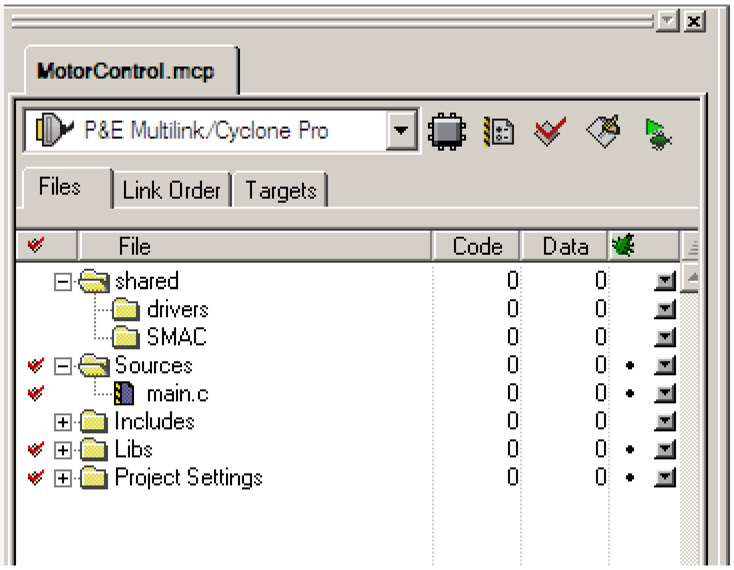


Figure 3. New Folders Tree

3. Go to the path where the MC9S08QD4 project is stored and create the same three folders. See Figure 4.

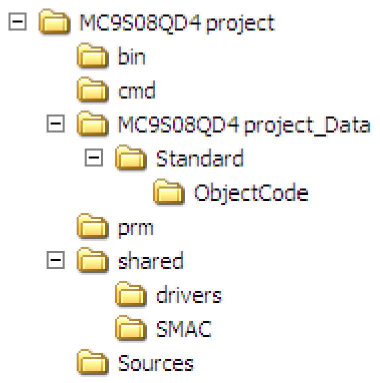
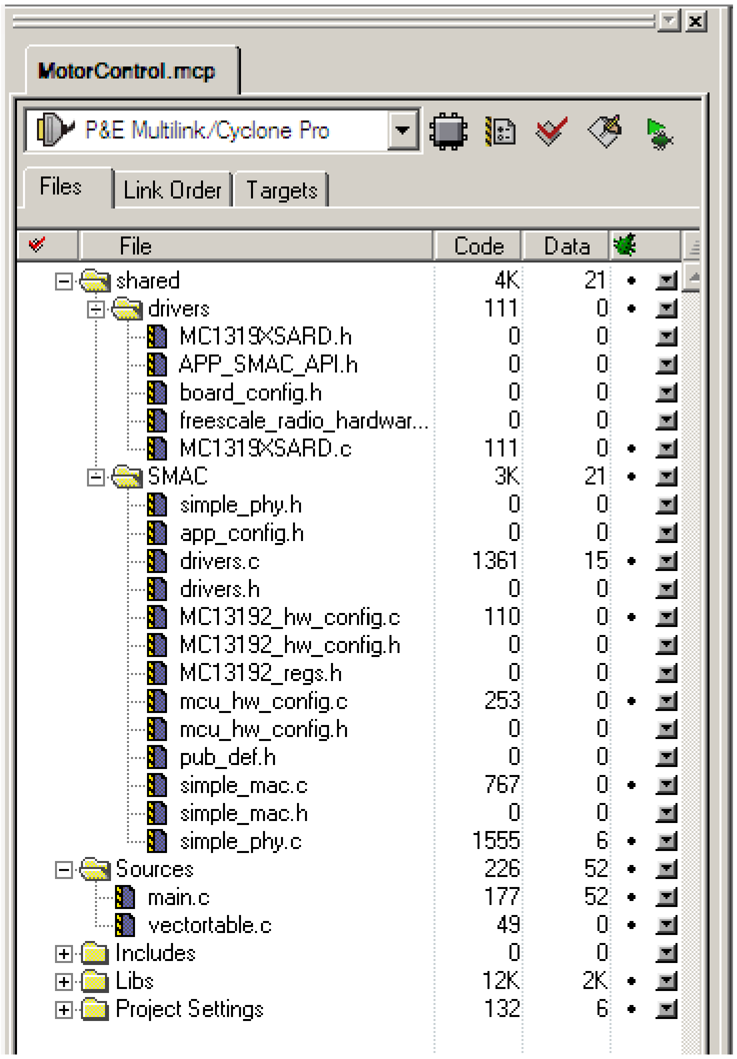


Figure 4. MC9S08QD4 Project Structure

4. The following files must be copied from the drivers folder inside the BeeKit Wireless Connectivity Toolkit imported project to the drivers folder inside the new MC9S08QD4 project:
   – APP_SMAC_API.h
   – Board_config.h
   – Freescale_radio_hardware.h
5. The next files must also be copied from the SMAC folder inside the BeeKit Wireless Connectivity Toolkit project to the SMAC folder inside the new MC9S08QD4 project:
   – app_config.h
   – MC13192_hw_config.c
   – MC13192_hw_config.h
   – MC13192_regs.h
   – mcu_hw_config.h
   – pub_def.h
   – simple_mac.h
   – simple_phy.c
   – simple_phy.h
6. Because the MC1319x transceiver uses an SPI bus to transfer information to the microcontroller, this peripheral is necessary for all SMAC applications. Therefore, the SPI driver and other useful files are found under the same application number in the www.freescale.com documentation site. All files modified to use SMAC with the MC9S08QD4 microcontroller are listed and explained below:
   – drivers.c: the timeout management in the IRQ interrupt service routine was modified. The SPI initialization was changed to use an SPI developed by firmware. SPIDvrWrite, SPIDvrRead, SPIDvrRead2, RAMDvrWriteTx, and RAMDvrReadRx functions were modified in order to use the SPI developed by firmware. VSPIRxMsg function was added.
   – drivers.h: the function prototype for VSPIRxMsg function was added.

   - mcu_hw_config.c: deassert and assert RXTXEN macro was modified because this signal is now managed by the transceiver’s GPIO1 pin. UseExternalClock and UseMcuClock functions were modified because the MC9S08QD4 microcontroller has only an internal source clock; consequently there is no external pin to connect an external source clock such as crystals or oscillators. The watchdog timer and the timer routine were modified into MCUInit function.
   - simple_mac.c: MLMERXEnableRequest and MLMERXDisableRequest functions were modified to use the transceiver’s timer1 without halting the transceiver’s timer counter.
   - MC1319XSARD.c: the configuration of GPIO1 pin of the transceiver as output was added at the radio initialization.
   - MC1319XSARD.h: all macros to control input and output pins were re-defined to use the MC9S08QD4 microcontroller. The reason for this is pin assignments and the number of pins between MC9S08GT60 and the MC9S08QD4 microcontrollers are very different.
   - vectortable.c: the complete pseudo-vector table was changed because the interrupt vectors for MC9S08GT60 and MC9S08QD4 are mapped at different addresses.
7. A template project for MC9S08QD4 and all the files explained in the last step can be downloaded from www.freescale.com under the application note number (AN3381SW). Copy the next downloaded files into MC9S08QD4 project as follows:
   - MC1319XSARD.c and MC1319XSARD.h files must be stored into the drivers folder.
   - drivers.c, drivers.h, mcu_hw_config.c, and simple_mac.c files must be stored into the SMAC folder.
   - vectortable.c and main.c file must be stored into the Sources folder.
8. After the files are copied in the specified path, all files stored in the different folders must be added to the project in the CodeWarrior software. Follow the instructions below.
   - Right click on the drivers folder and choose the Add Files… option. On the pop-up window, find the drivers folder path into MC9S08QD4 project and choose all files inside, then click on open button. Repeat the same instructions for the SMAC folder.
   - Right click on the Sources folder and choose the Add Files… option. On the pop-up window, find the Sources folder path into MC9S08QD4 project and choose only vectortable.c and main.c files, then click on the open button.



MotorControl.mcp

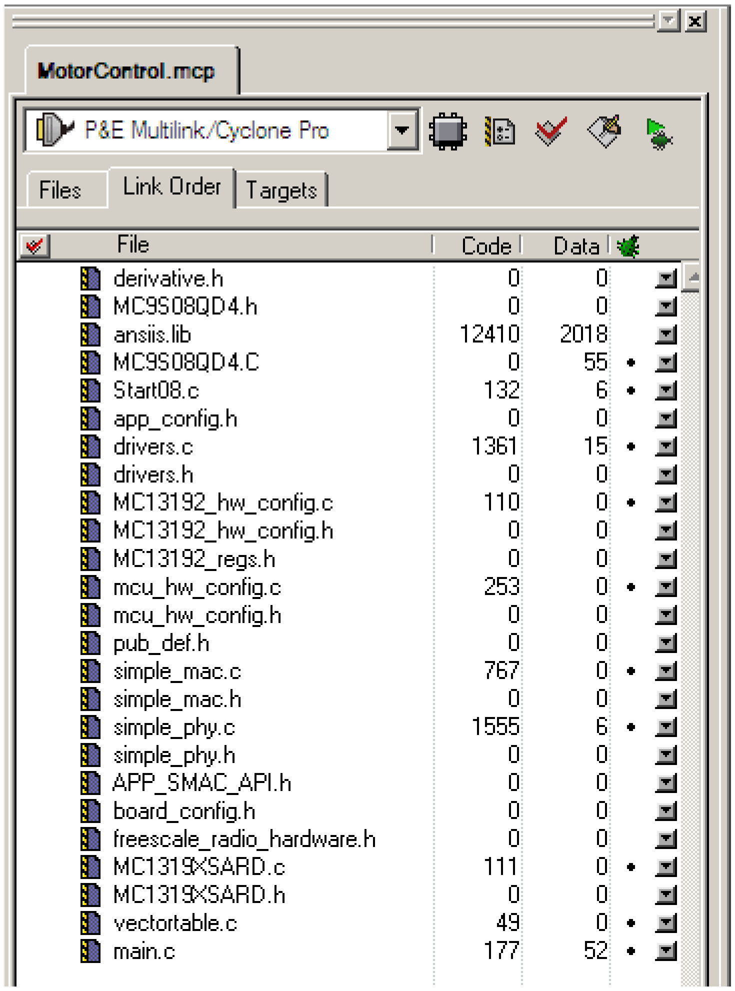
P&E Multilink/Cyclone Pro

Files | Link Order | Targets

File	Code	Data	
shared	4K	21	•
drivers	111	0	•
MC1319XSARD.h	0	0	
APP_SMAC_API.h	0	0	
board_config.h	0	0	
freescale_radio_hardwar...	0	0	
MC1319XSARD.c	111	0	•
SMAC	3K	21	•
simple_phy.h	0	0	
app_config.h	0	0	
drivers.c	1361	15	•
drivers.h	0	0	
MC13192_hw_config.c	110	0	•
MC13192_hw_config.h	0	0	
MC13192_regs.h	0	0	
mcu_hw_config.c	253	0	•
mcu_hw_config.h	0	0	
pub_def.h	0	0	
simple_mac.c	767	0	•
simple_mac.h	0	0	
simple_phy.c	1555	6	•
Sources	226	52	•
main.c	177	52	•
vectortable.c	49	0	•
Includes	0	0	
Libs	12K	2K	•
Project Settings	132	6	•

Figure 5. File and Folder Placement in the MC9S08QD4 Project

9. To ensure the correct code performance, all files added to the project must be linked in a specific order. The main application file must always be linked last. Figure 6 shows the link order to follow. The main.c file is at the end of the list. This means the main.c file is the last file linked.



File	Code	Data	
derivative.h	0	0	
MC9S08QD4.h	0	0	
ansiis.lib	12410	2018	
MC9S08QD4.C	0	55	•
Start08.c	132	6	•
app_config.h	0	0	
drivers.c	1361	15	•
drivers.h	0	0	
MC13192_hw_config.c	110	0	•
MC13192_hw_config.h	0	0	
MC13192_regs.h	0	0	
mcu_hw_config.c	253	0	•
mcu_hw_config.h	0	0	
pub_def.h	0	0	
simple_mac.c	767	0	•
simple_mac.h	0	0	
simple_phy.c	1555	6	•
simple_phy.h	0	0	
APP_SMAC_API.h	0	0	
board_config.h	0	0	
freescale_radio_hardware.h	0	0	
MC1319XSARD.c	111	0	•
MC1319XSARD.h	0	0	
vectortable.c	49	0	•
main.c	177	52	•

Figure 6. Shows the link order to follow

10. Now the project is complete and ready to work, just press click on the Make button and load the code in the MC9S08QD4 microcontroller.

10 User Guide

Up to now, this document explained only how to connect the transceiver and the microcontroller and how to import code for the MC9S08QD4 device. The section below will explain an example created for the MC9S08QD4 device called MotorControl and provided as software for this application note.

11 Motor Control Application

11.1 Overview

This application is a step motor controller where each motor coil is managed with one GPIO provided by the transceiver. As was said in a previous section, the transceiver has seven GPIOS. Two of them control the transceiver. This means the application can use the five other transceiver’s GPIOs. The application manages a step motor that has four coils. Table 4 describes which general purpose pins of the transceiver control the step motor coils.

Table 4. Step Motor Connections

Pin	Transceiver Pin Name	Type	Power interface
25	GPIO7	Digital Output	Coil1
24	GPIO6	Digital Output	Coil2
23	GPIO5	Digital Output	Coil3
8	GPIO4	Digital Output	Coil4

Each coil must be energized in the correct sequence to move the motor in the desired direction. If the sequence is inverted, the motor direction is changed.

Table 5 shows the coils sequence to move the motor clockwise and counterclockwise.

Table 5. Coils Sequence

Sequence	GPIO7	GPIO6	GPIO5	GPIO4	Coils
1	0	0	0	1	Only Coil4 energized
2	0	0	1	0	Only Coil3 energized
3	0	1	0	0	Only Coil2 energized
4	1	0	0	0	Only Coil1 energized

As Table 5 shows, if the sequence order is 1 – 4, the motor moves clockwise. If the sequence order changes to 4 – 1, the motor moves counterclockwise.

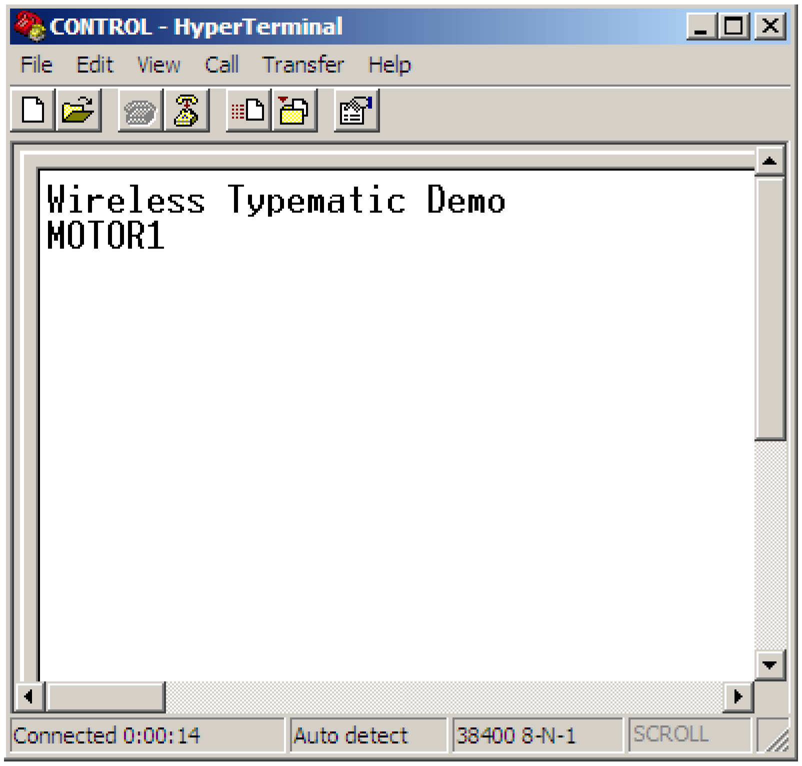
The SMAC protocol receives information from any other SMAC compliant wireless device. The motor control application has a set of commands where each one executes a different action to control the step motor. The commands to control the step motor are listed in Table 6.

Table 6. Commands

Command	Character received	Description
A command	A	Stop motor
B command	B	Low speed
C command	C	Medium speed
D command	D	High speed
E command	E	Counterclockwise direction
F command	F	Clockwise direction

12 Setup Procedure

1. To test the motor control application, use the wireless UART application loaded on THE SRB, NCB, SARD, or EVB board. Make sure the wireless UART application uses channel 7 to transmit and receive.
2. Connect the board to the serial port.
3. Open HyperTerminal. Port settings must be:
   — Bits per second: 38400
   — Data bits: 8
   — Parity: None
   — Stop bits: 1
   — Flow control: None
4. Turn on the wireless UART application. Wireless Typematic Demo is displayed in the HyperTerminal.
5. Turn on motor control application. The message MOTOR1 is displayed in the HyperTerminal.
6. Try any command described in the Table 6. To do so, type any uppercase letter between A and F in the hyperterminal window. For example, type B and the GPIO pins start to generate the sequence described in Table 5.



```
Wireless Typematic Demo
MOTOR1
```

Connected 0:00:14 | Auto detect | 38400 8-N-1 | SCROLL

Figure 7. Device MOTOR1 Detected on the Wireless Network

NOTE

All characters typed must be uppercase. No command displays information on the HyperTerminal.

13 Conclusion

This application note shows any SMAC application can be used with almost any 8-bit Freescale microcontrollers from low-end to the high-end. This document did not discuss the BeeKit Wireless Connectivity Toolkit software in detail, but does demonstrate that BeeKit Wireless Connectivity Toolkit is and easy and fast way to develop SMAC applications. Sometimes, developers do not use low-end microcontrollers because these microcontrollers have hardware limitations such as communication peripherals or number of pins. See the BeeKit Wireless Connectivity ToolKit User's Guide (BKWCTKUG) for more details. However, as this application note shows, you can find the way to solve any kind of limitation. Here the unique limitation is your imagination. Think big.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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

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

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

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

Document Number: AN3381
Rev. 0
11/2006