Freescale Semiconductor
Application Note

Document Number: AN4448
Rev. 0, 01/2012

MC9S12VR Family Demonstration Lab Training

by: Carlos Aceff
Automotive & Industrial Solutions Group, Mexico

1 Introduction

This document explains how to use the code contained in the lab example package created for Freescale's MC9S12VR microcontroller. The lab example package contains reference code that shows how to configure and use the following modules:

- S12FTMRG (for flash memory writing)
- S12PWM8B8CV2 (for PWM generation)
- S12SCIV5 (for SCI communications)
- BATSV2 (to use the battery sensor)
- S12HSDRV1 (for high-side driver use)
- S12LSDRV1 (for low-side driver use)
- S12GMMCV1 (to navigate in the different flash memory pages)
- S12SPIV5 (for SPI usage)

The lab example package can be downloaded from www.freescale.com.

Contents

© Freescale Semiconductor, Inc., 2012. All rights reserved.

2 Setup

2.1 Tool setup

The lab examples are intended to run on the evaluation board S12VR64EVB; this EVB can be purchased from Freescale's website.

Also, it is a prerequisite to install an evaluation (or permanent) version of CodeWarrior Development Studio for S12VR; this software comes with the S12VR64EVB, and its installation guide is included in Freescale document LL18UHVQSG, *Quick Start Guide: S12VR64EVB,* which also comes with the EVB.

2.2 Board setup

To properly configure the S12VR64EVB, the action items in this list must be performed.

1. The S12VR64EVB board must be configured with the default jumper settings as shown in the *Quick Start Guide: S12VR64EVB*.
2. Power the board by connecting a 12 V power supply in J6 or in J4 and J5.
3. Push the on/off switch (SW3) to the on position.
4. Connect an A/B USB cable to an open USB port on the host PC, and the USB mini-connector to the S12VR64EVB demonstration board. Follow the onscreen instructions to install the necessary USB drivers, if required.

3 Demonstration lab examples

3.1 Flash programming example

The flash memory module is used mainly to write and protect the information in the flash memory from change. This is done by writing commands to the register area of the flash memory module. Therefore this program will show how to write these commands so that most of the flash memory sectors are written with a known pattern.

Procedure LaunchFlashCommand in file flash.c is the routine used to write the flash commands. Examining this procedure will help to understand how the registers need to be manipulated.

3.1.1 Setup

1. Open CodeWarrior.
2. From the CodeWarrior main menu, choose File → Open and select the 9S12VR64_FLASH_DEMO.mcp file.
3. Click Open. The project window opens.
4. From the main menu choose Project → Debug. This compiles the source code, generates an executable file, and downloads it to the demo board.
5. A new debugger environment will open.

Notice that this program is loaded in RAM; this is necessary because the program flash memory cannot be read while a mass erase is occurring or if the program flash memory is being updated.

Also, the flash configuration field will not be changed. If it is accidently erased (due to modifications in the software made by the user) the part will be secured and further write operations will not be permitted. For information on how to unsecure the uC, please refer to the P&E Micro website: http://www.pemicro.com/

3.1.2 Instructions

The purpose of this lab exercise is to show how to program most of the program flash memory that is available and all of the data flash memory.

Simply press F5 to execute the instructions of the program; the debugger will automatically stop at programmed locations where BGND instructions were inserted. BGND instructions activate the BDM, causing the program flow to stop when running from the debugger.

3.1.2.1 First BGND location

The debugger will stop right before executing the divider configuration of the flash module (setting the flash divider register is a prerequisite before being able to write flash commands). You can step through the code by pressing F11. After examining this code you can press F5 again to stop at the next BGND instruction.

3.1.2.2 Second BGND location

The debugger will stop now at the function that writes the flash memory. Step through (by pressing F11) to see the details of this function. You can press F5 anytime to continue with the execution of the program.

3.1.3 Summary

The lab exercise shows how to initialize the flash memory module and how to write commands in its registers so that all the data flash memory and most of the program flash memory gets written with a known data pattern.

3.2 BAT sensing demonstration

The BATS module provides the functionality to measure the voltage of the battery supply pin VSENSE or of the chip supply pin VSUP. The voltage present on either the VSENSE pin or the VSUP pin can be routed via an internal divider to the internal analog-to-digital converter. Independent of the routing to the ADC, it is possible to route one of these voltages to a comparator to generate a low- or a high-voltage interrupt to alert the microcontroller.

Please read function BATS_SetAdcConnectionChannel in file bats_hal.c; this has the code required to route the desired reference voltage to the ADCs.

3.2.1 Setup

1. Open CodeWarrior.
2. From the CodeWarrior main menu, choose File → Open and select the S12VR64_BATS_DEMO.mcp file.
3. Click Open. The project window opens.
4. From the main menu choose Project → Debug. This compiles the source code, generates an executable file, and downloads it to the demo board.
5. A new debugger environment will open.
6. Connect a serial cable between the evaluation board and the computer.
7. Open a serial terminal and configure it as follows:
   - 57600 bps
   - 8 data bits
   - 1 stop bit
   - No parity

3.2.2 Instructions

Press F5 at the debugger to start the program. The terminal will show the digitalized value of VSENSE. The serial terminal will show the changes in the power supply, so you will need to change the voltage in the power supply to see the changes in the digitalization value of VSENSE.

3.2.3 Summary

This lab example shows how to configure the BATS module so that the internal VSENSE is routed to the analog converter. The serial terminal shows the digitized value.

3.3 PWM module

Version 2 of the S12 PWM module is a channel-scalable and optimized implementation of S12 PWM8B8C Version 1. The channel is scalable in pairs from PWM0 to PWM7. The available channel numbers are 2, 4, 6, and 8.

This lab example shows one possible way to set up and use the pulse width modulation (PWM) module to create a 50% duty cycle output with different polarity settings. This behavior is best illustrated if all of the PWM signals can be displayed simultaneously on a two-channel oscilloscope.

Please examine the main.c file for details on the configuration of the module.

3.3.1 Setup

These steps must be completed before running the lab example.

1. Start CodeWarrior by selecting it in the Windows Start menu.

2. From the CodeWarrior main menu, choose File → Open and select the 9S12VR64_PWM_DEMO.mcp file.
3. Click Open. The project window will open.
4. The C code of this demonstration is contained within the main.c file. Double-click on the file to open it.
5. From the main menu choose Project → Debug. This compiles the source code, generates an executable file, and downloads it to the demo board.
6. A new debugger environment will open. After the download to the demo board is complete, close the debugger environment.

3.3.2 Instructions

1. Press the RESET button SW4. The code must run and configure the PWM channels in this way: 50% duty cycle, 24.7 kHz signals on ports PP0 and PP1 (visible at J11, pin 2 and 5).
2. Try probing the two signals simultaneously if possible. This will show the polarity difference of the signals.

3.3.3 Summary

The PWM is a common module on many microcontrollers; it can be used in applications such as a dimming control where the on period determines the luminosity of the application. In this lab example it was shown how to configure the PWM module to produce two signals of complementary polarity.

3.4 MMC program flash paging window

The memory of this microcontroller is paged, and the page is mapped between addresses 0x8000 and 0xC000. The page register is called PPAGE.

This lab example shows how to access all the memory regions by changing the memory page. For information regarding how to update the PPAGE register, please see the main.c file in S12VR64_MMC_DEMO.mcp.

3.4.1 Setup

These steps must be completed before running the lab example:

1. Start CodeWarrior by selecting it in the Windows Start menu.
2. From the CodeWarrior main menu, choose File → Open and select the S12VR64_MMC_DEMO.mcp file.
3. Click Open. The project window will open.
4. The C code of this demonstration is contained within the main.c file. Double-click on the file to open it.
5. From the main menu choose Project → Debug. This compiles the source code, generates an executable file, and downloads it to the demo board.
6. A new debugger environment will open. Do not close the debugger environment.

3.4.2 Instructions

This program is designed to stop at locations where the user is expected to see the memory page updated. Thus, the user is required to press F5 at the beginning and again every time the program stops. When the program stops, the user will be able to see the memory window and PPAGE register in the register section. The user will note that information at address 0x8000 will change every time the PPAGE register is updated. The user will have the chance to see the following pages: 0xD, 0xE, 0xF, 0xC.

The first time the program stops, register PPAGE will have a 0x0D value. At location 0x8000, the user will see the message 0D PPAGE. After that message is seen, the user should press F5 again to see the next page; in this case PPAGE will point to 0x0E and the memory region will show 0E PPAGE.

Please repeat this procedure to check all the memory regions.

3.4.3 Summary

The S12GMMC is capable of mapping up to 64 KB of flash memory, 512 bytes of EEPROM, and 2 KB of RAM into the global memory map. The access of some of this global memory is through a page register.

In this lab example it was shown how to change the PPAGE register to access all the paged memory.

3.5 SCI communications

The SCI allows asynchronous serial communications with peripheral devices and other CPUs.

This lab example shows how to configure the SCI module to transmit and receive data using the following configuration:

- 57600 bps
- 8 data bits
- No parity
- 1 stop bit

For information regarding the configuration of the hardware, please refer to the main.c file in this project.

3.5.1 Setup

These steps must be completed before running the lab example.

1. Start CodeWarrior by selecting it in the Windows Start menu.
2. From the CodeWarrior main menu, choose File → Open and select the 9S12VR64_SCI_DEMO.mcp file.
3. Click Open. The project window will open.
4. From the main menu choose Project → Debug. This compiles the source code, generates an executable file, and downloads it to the demo board.
5. A new debugger environment will open.

6. The software uses the RS-232 port to interact with the user. Open a terminal window (baud rate = 57600, data bits = 8, parity = N, stop bits = 1, flow control = none) to see the RS-232 port data.
7. Connect a serial cable between DB9 in the EVB and your host computer.

3.5.2 Instructions

Press F5 to execute the program; the code begins execution and configures the SCI to the selected baud rate. The serial terminal will show the information transmitted by the SCI module. You can send back data to the SCI following the information displayed in the serial terminal. Choose some options and observe the configuration of the registers.

3.5.3 Summary

In this lab example, the SCI module was used to provide information regarding the state of the hardware, showing how to configure the SCI to output and input information.

3.6 SPI communications

The SPI module allows duplex, synchronous, serial communication between the MCU and peripheral devices. In this lab example it is shown how to configure the module as master to continually transmit a data byte.

For information regarding the configuration of the module, please refer to file main.c in this project.

3.6.1 Setup

An oscilloscope and three scope probes are required for this demo. These steps must be completed before running the lab example:

1. Start CodeWarrior by selecting it in the Windows Start menu.
2. From the CodeWarrior main menu, choose File → Open and select the S12SPIV5.mcp file.
3. Click Open. The project window will open.
4. The C code of this demonstration is contained within the main.c file. Double-click on the file to open it.
5. From the main menu choose Project → Debug. This compiles the source code, generates an executable file, and downloads it to the demo board.
6. A new debugger environment will open. After the download to the demo board is complete, close the debugger environment.
7. Attach scope probes to signals CLK_SPI, SPI_MISO, and SPI_SS on header J16. For information regarding this jumper, consult the schematics of the evaluation board available at www.freescale.com.
8. Configure the oscilloscope to trigger on the falling edge of CLK_SPI.

3.6.2 Instructions

Execute these instructions to run the lab example.

1. Press the RESET button SW4. The code begins execution and configures the SPI to transmit a byte of data at a baud rate of 3.125 Mbit/s.
2. Monitor the SPI transmission on the oscilloscope to see the relationship between slave select, data transmitted on MOSI, and the serial clock signals.

3.6.3 Summary

The SPI module can be used to allow duplex synchronous serial communication between peripheral devices and the MCU. In this lab example it is shown how to set up and use the SPI module in master mode to transmit constant information.

3.7 LSDRV module

The LSDRV module provides two low-side drivers typically used to drive inductive loads (relays). This lab example shows how to configure the hardware to show the following functionality: low-side driver enable and over-current detection.

Details regarding use of the drivers can be found in file main.c in this project.

3.7.1 Setup

1. Start CodeWarrior by selecting it in the Windows Start menu.
2. From the CodeWarrior main menu, choose File → Open and select the S12VR64_LS_DEMO.mcp file.
3. Click Open. The project window will open.
4. The C code of this demonstration is contained within the main.c file. Double-click on the file to open.
5. From the main menu choose Project → Debug. This compiles the source code, generates an executable file, and downloads it to the evaluation board.

3.7.2 Instructions

This program has a BGND instruction inserted at a point where the user is required to input a parameter. Therefore the user can start the execution of the program by pressing F5. When the program stops, the user is required to change the variable “test” — this variable is shown in the data window. The possible values that this variable can take are:

To activate LS0	1
To activate LS1	2

To deactivate LS0	3
To deactivate LS1	4
To test the over-current functionality in LS0	7
To test the over-current functionality in LS1	8

After changing variable “test,” please press F5 to execute the test.

To test the over-current functionality you must:

1. Change the variable “test” to 7 or 8.
2. Run the program.
3. Connect a 20 Ω resistor between J21 and R12 (for LS0) or J22 and R13 (for LS1).
4. Observe how the variable LS0OC_Flag or LS1OC_Flag sets and the output is turned off.

To exit this test mode please reset the microcontroller by pressing Ctrl+R.

3.7.3 Summary

The LSDRV module provides two low-side drivers typically used to drive inductive loads (relays). In this lab example it was shown how to configure the low-side drivers to activate a relay and detect over-current conditions. Please note that the maximum current these drivers can handle is 150 mA.

3.8 HSDRV module

The HSDRV module provides two high-side drivers typically used to drive LED or resistive loads (typical 240 Ω). An incandescent or halogen lamp is not considered here as a possible load.

This lab example will show how to configure the hardware to enable the following functionality: high-side driver enable and over-current condition.

Please read file main.c to learn more about the hardware configuration of the module.

3.8.1 Setup

1. Start CodeWarrior by selecting it in the Windows Start menu.
2. From the CodeWarrior main menu, choose File → Open and select the S12VR64_HS_DEMO.mcp file.
3. Click Open. The project window will open.
4. The C code of this demonstration is contained within the main.c file. Double-click on the file to open it.
5. From the main menu choose Project → Debug. This compiles the source code, generates an executable file, and downloads it to the evaluation board.

3.8.2 Instructions

This program has a BGND instruction inserted at a point where the user is required to input a parameter. Therefore the user can start the execution of the program by pressing F5. When the program stops, the user is required to change the variable “test”; this is shown in the data window. The possible values that this variable can take are:

To activate the HS0	1
To activate the HS1	2
To deactivate the HS0	3
To deactivate the HS1	4
To test the over current functionality in HS0	7
To test the over current functionality in HS1	8

After changing variable “test,” please press F5 to execute the test.

To test the over-current functionality you must:

1. Change the variable “test” to 7 or 8.
2. Run the program.
3. Connect a 20 Ω resistor between the anode and cathode of D7 (for HS0) or the anode and cathode of D8 (for HS1).
4. Observe how the variable HS0OC_Flag or HS1OC_Flag sets and the output is turned off.

To exit this test mode please reset the microcontroller by pressing Ctrl+R.

3.8.3 Summary

The high-side drivers in the MCS12VR64 can be used to drive small loads. In this lab example it was shown how to configure them to drive LEDs and a relay; also, the open loop and over-current condition functionality was presented.

4 Conclusions

The MCS12VR family of microcontrollers (MCUs) offers the enhanced features of 16-bit microcontrollers; this family comes with high- and low-side drivers that allow direct handling of relays and LEDs. This family is ideal for a wide range of central body control applications, such as low-end window lifters or sunroof controllers.

A zip file, AN4448SW.zip, containing the complete CodeWarrior projects for the lab examples accompanies this application note. The file can be downloaded from www.freescale.com.

5 Useful reference materials

The following material is available at www.freescale.com.

Software development tools:

- CodeWarrior Development Studio for HCS12(X) Microcontrollers, Rev 5.9

Application notes:

- AN2612 — *PWM Generation Using HCS12 Timer Channels,* Rev. 1, 1/2006
- AN2428 — *An Overview of the HCS12 ATD Module,* Rev. 0, 01/2003
- AN2883 — *Serial Communication Interface as UART on HCS12 MCUs,* Rev. 1.0, 05/2005

Reference manual:

- MC9S12VRRMV2, *MC9S12VR-Family Reference Manual,* Rev. 2.2, June 20, 2011

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4448
Rev. 0
01/2012

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

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

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

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.
All other product or service names are the property of their respective owners.
© Freescale Semiconductor, Inc. 2012. All rights reserved.