弘憶國際股份有限公司 (深圳分公司)
G.M.I. TECHNOLOGY INC. (SZ BRANCH)
深圳罗湖区沿河北路 1002 号京广商业大厦 18 楼
TEL 電話 :(0755)33388247 FAX 傳真:(0755)33388206

MC9RS08KA1/2 中文资料

MC9RS08KA2 在一个超小型封装中集成了比常规的低端解决方案更多的 Flash 和 RAM，从而在减省成本的同时增加了设计的灵活性。另外这款器件还具有低工作电压、系统保护和模拟控制的能力，从而使设计更为轻松。

应用：从儿童玩具、电动牙刷到扬声器和照明系统等各种消费类产品。亦可作为电压监控和其它简单逻辑电路的低成本替代方案。

MC9RS08KA2 的特点

- 包含 8 位 RS08 内核
  - 1.8V 工作电压下总线频率高达 10MHz，最小的指令执行时间为 100ns
  - RS08 指令集
  - 支持微寻址/短寻址模式
  - 14 字节快速存取 RAM
  - 可以模仿 HC08/HCS08 内核的零偏移变址寻址模式指令
- 第三代 Flash 和 RAM（极快的字节写入编程）
  - 63 字节 RAM
  - 2K Flash (1K Flash 同时供应)
- 灵活的时钟选择
- 4 条双向输入/输出线路，可以通过软件选择上拉（无需外置电阻）
- 模拟比较器
- 实时中断
- 8 位预定标的 8 位定时器
- 系统保护
  - 程序失控或出错时自动复位
  - 低电压检测
  - 非法操作码和非法地址检测
  - Flash 保密功能
- 单线调试和仿真接口；无需昂贵的仿真工具或费时的硬件开发

freescale 飞思卡尔半导体
Contact Window: Johnson zhan (詹志明) Mobile:**13728781297**
E-mail: Johnsonzhan@gmitec.com Website: www.freescale.com.cn
Msn: plczhan@hotmail.com QQ: 747804604

8 位的 RS08 中央处理器

- 增加了高性能指令
  1. LDA ,STA 和 CLR 指令 支持短寻址模式 从$0000 到$001F 都能通过单字节指令访问;
  2. ADD,SUB,INC 和 DEC 支持 TINY 寻址模式;从$0000 到$000F 能够在减少的指令周期内被单字节访问;
  3. PC 映射寄存器指令:SHA 和 SLA
- 待定的中断指示
- 通过 D[X] 和 X 寄存器来索引寻址
- 在整个存储器空间里通过分页窗口直接访问页

存储器

- FLASH EEPROM
  MC9RS08KA2: 2048 字节 1 字节=8 位
  MC9RS08KA1: 1024 字节
- 63 字节内部 RAM

省电模式

- 等待和停止模式
- 能够通过 RTI ,KBI 或 ACMP 方式唤醒

时钟源

- ICS---精整的 20MHZ 内部时钟源
  高达 10MHZ 的内部总线频率
  0.2%的精度（正常情况）2% 当温度和电压范围超过时

系统保护

- 当系统跑飞时 COP 能复位，其总线时钟源是独立的
- 低电压检测通过复位或者唤醒

外围接口

- MTIM --- 8 位的时钟模块
- ACMP ----模拟比较器
  全部支持 RAIL-TO-RAIL 的操作
  可选择的内部参考电压
  能在停机模式工作
- KBI--- 键盘中断接口
  3 个 KBI 口 ---在 6 脚的封装里
  5 个 KBI 口 ---在 8 脚的封装里

开发工具

- 后台调试系统 BDS
- 在线调试时可支持单个断点

封装选项

- 6 PIN 双平面无引脚 （DFN）封装
  2 个 GPIO 1 GPI 1 PGO 总共 4 个 I/O

- 8 PIN 塑胶双列直插 （PDIP）封装
  4 GPIO 1 GPI 1 GPO 总共 6 个 I/O
- 8 PIN 窄体贴片 SOIC 封装
  4 个 GPIO 1 个 GPI 1 个 GPO 共 6 个 I/O

引脚描述：

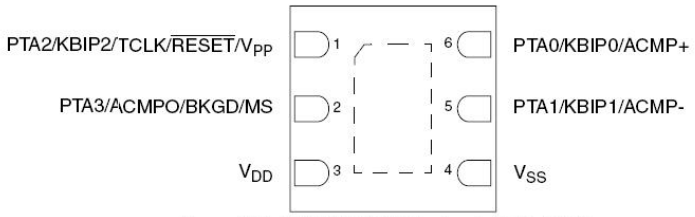


Figure 2-1. MC9RS08KA2 Series in 6-Pin DFN

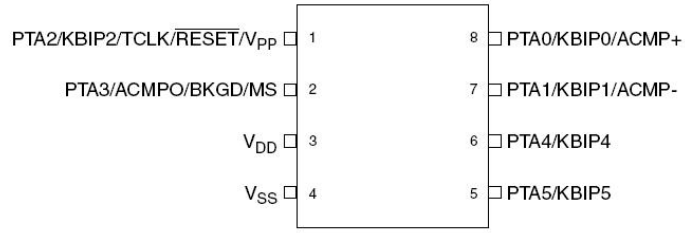


Figure 2-2. MC9RS08KA2 Series in 8-Pin PDIP

引脚功能描述：

Table 2-1. Pin Sharing Reference

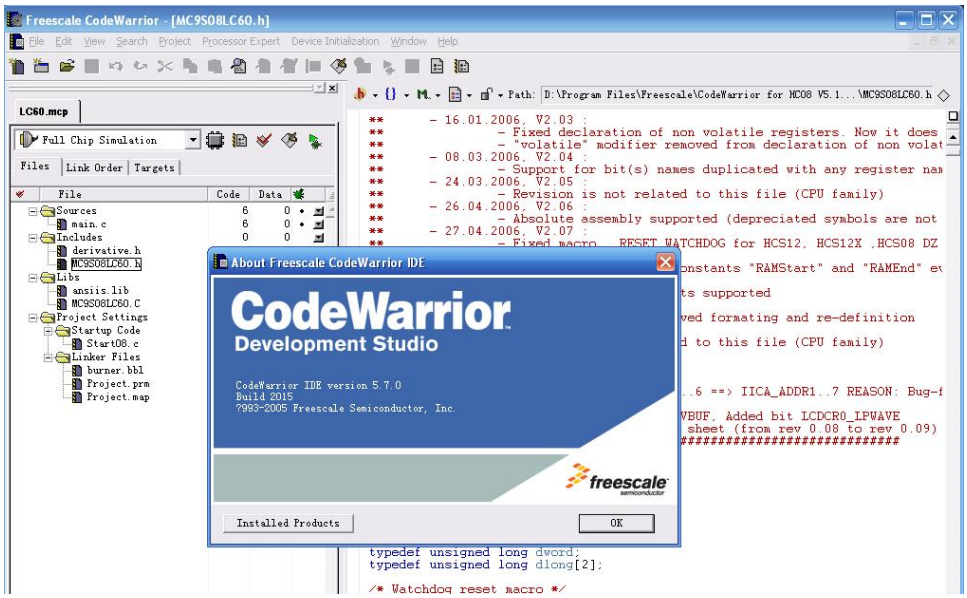
Pin Name	Direction	Pullup/Pulldown[1]	Alternative Functions[2]	
V_{DD}	—	—	Power	
V_{SS}	—	—	Ground	
PTA0	I/O	SWC	PTA0 KBIP0 ACMP+	General-purpose input/output (GPIO) Keyboard interrupt (stop/wait wakeup only) Analog comparator input
PTA1	I/O	SWC	PTA1 KBIP1 ACMP-	General-purpose input/output (GPIO) Keyboard interrupt (stop/wait wakeup only) Analog comparator input
PTA2	I	SWC[4]	PTA2 KBIP2 TCLK RESET V_{PP}	General-purpose input Keyboard interrupt (stop/wait wakeup only) Modulo timer clock source Reset V_{PP}
PTA3	I/O[3]	—[4]	PTA3 ACMPO BKGD MS	General-purpose output Analog comparator output Background debug data Mode select
PTA4[5]	I/O	SWC	PTA4 KBIP4	General-purpose input/output (GPIO) Keyboard interrupt (stop/wait wakeup only)
PTA5[5]	I/O	SWC	PTA5 KBIP5	General-purpose input/output (GPIO) Keyboard interrupt (stop/wait wakeup only)

1 SWC is software-controlled pullup/pulldown resistor; the register is associated with the respective port.
2 Alternative functions are listed lowest priority first. For example, GPIO is the lowest priority alternative function of the PTA0 pin; ACMP+ is the highest priority alternative function of the PTA0 pin.
3 Output-only when configured as PTA3 function.
4 When PTA2 or PTA3 is configured as RESET or BKGD/MS, respectively, pullup is enabled. When V_{PP} is attached, pullup/pulldown is disabled automatically.
5 This pin is not available in 6-pin package. Enabling either the pullup or pulldown device is recommended to prevent extra current leakage from the floating input pin.

弘憶國際股份有限公司 (深圳分公司)
G.M.I. TECHNOLOGY INC. (SZ BRANCH)
深圳罗湖区沿河北路 1002 号京广商业大厦 18 楼
TEL 電話 :(0755)33388247 FAX 傳真:(0755)33388206

开发工具和开发环境

CodeWarrior IDE 能上网自由下载 32K 限制版使用



仿真器

1.USBMULTILINKBDME 仅售 99USD 能仿真，并支持下载所有 HCS08（8 位机） 和 HCS12(16 位机) 内核的 MCU。

Contact Window: Johnson zhan (詹志明) Mobile:**13728781297**
E-mail: Johnsonzhan@gmitec.com Website: www.freescale.com.cn
Msn: plczhan@hotmail.com QQ: 747804604

弘憶國際股份有限公司 (深圳分公司)
G.M.I. TECHNOLOGY INC. (SZ BRANCH)
深圳罗湖区沿河北路 1002 号京广商业大厦 18 楼
TEL 電話 :(0755)33388247 FAX 傳真:(0755)33388206

2. USBSPYDER08 简易低价型，但目前只能支持 MC9RS08KA MC9S08QD MC9S08QG 系列 仅仅售 29USD,能仿真和下载。

烧录器（indart one） 价格是 399USD SOFTEC 公司出品

能支持所有 HC08，HCS08，RS08，-S12,-12X 所有系列，可以支持 USB 联机烧录。

Contact Window: Johnson zhan (詹志明) Mobile:**13728781297**
E-mail: Johnsonzhan@gmitec.com Website: www.freescale.com.cn
Msn: plczhan@hotmail.com QQ: 747804604

弘憶國際股份有限公司 (深圳分公司)
G.M.I. TECHNOLOGY INC. (SZ BRANCH)
深圳罗湖区沿河北路 1002 号京广商业大厦 18 楼
TEL 電話 :(0755)33388247 FAX 傳真:(0755)33388206

大批量烧录器 （M68CYCLONEPROE）售价为 499USD PEMICRO 出品。

能支持 HC08,HCS08,RS08,HCS12 全系列，并且支持 USB，以太网，串口和电脑相连，支持脱机烧录，单按键编程，方便快速。

演示板（开发板） 售价 50USD

DEMO9RS08KA2 演示板的内置电路和 USB 接口可简化和电脑的连接从而加快代码的开发。

低成本的 KA2 演示板是支持我们最新推出的 RS08 架构的第一款开发工具。**连接、下载、评估**。对，就是这么简单。

DEMO9RS08KA 的特点

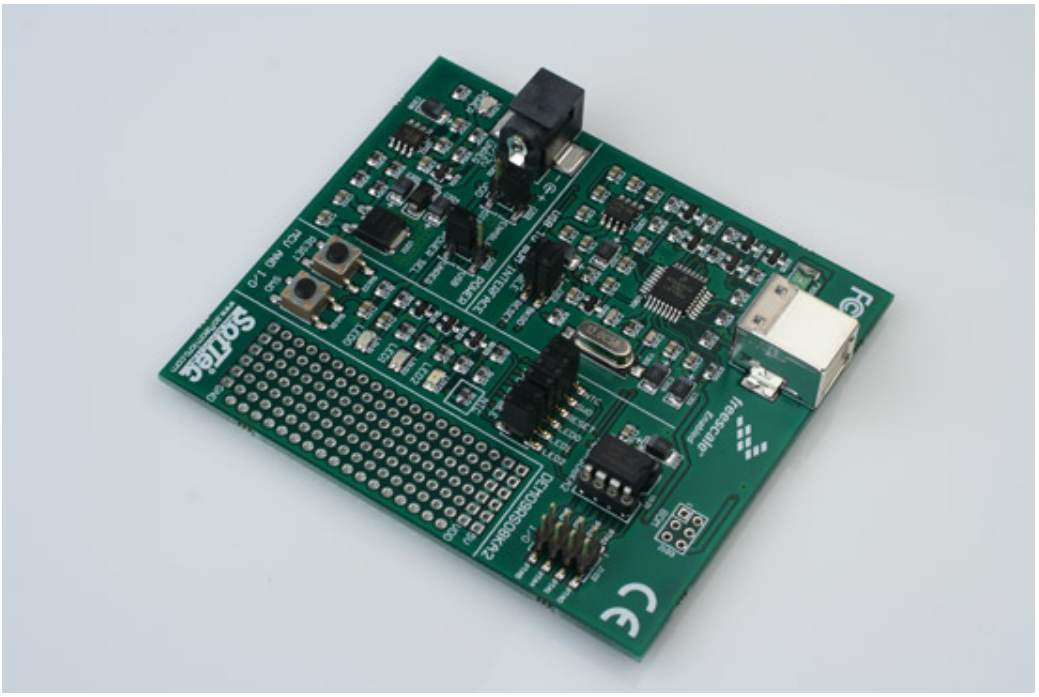
8 引脚、双列直插式 RS08KA2 微控制器
GPIO 插头， 4x2 引脚
内置 USB BDM 接口
2 个按钮开关电路：1 个用户自定义，1 个用于复位
4 条 LED 灯电路连线：3 条用户自定义，1 条用于指示 VDD
BDM 插头
电源输入选择
　　USB 接口：最大电流为 500mA
　　直流电源插头：电压范围 7V～18V，通常为 9VDC

freescale 飞思卡尔半导体
Contact Window: Johnson zhan (詹志明) Mobile:**13728781297**
E-mail: Johnsonzhan@gmitec.com Website: www.freescale.com.cn
Msn: plczhan@hotmail.com QQ: 747804604

弘憶國際股份有限公司 (深圳分公司)
G.M.I. TECHNOLOGY INC. (SZ BRANCH)
深圳罗湖区沿河北路 1002 号京广商业大厦 18 楼
TEL 電話 :(0755)33388247 FAX 傳真:(0755)33388206

DEMO9RS08KA2 套件包含

DEMO9RS08KA2 演示板
CodeWarrior™集成开发环境和服务包光盘
USB 电缆
快速启动指南
用户手册
装箱单
3 个备用的 8 引脚 PDIP 封装的 9RS08KA2 微控制器样片



Contact Window: Johnson zhan (詹志明) Mobile:**13728781297**
E-mail: Johnsonzhan@gmitec.com Website: www.freescale.com.cn
Msn: plczhan@hotmail.com QQ: 747804604

弘憶國際股份有限公司 (深圳分公司)
G.M.I. TECHNOLOGY INC. (SZ BRANCH)
深圳罗湖区沿河北路 1002 号京广商业大厦 18 楼
TEL 電話 :(0755)33388247 FAX 傳真:(0755)33388206

竞争优势：

	MC9RS08KA2	PIC10F200	PIC10F202	PIC10F206	ATtiny11L	ATtiny12L
Flash	2K Bytes	256 words	512words	512words	1K bytes	1K bytes
RAM	63 bytes	16 bytes	24 bytes	24 bytes	32 bytes	32 bytes
Bus Freq	8,4,2,1 MHz 16,8,4,2 KHz	1MHz	1MHz	1MHz	0-2MHz	1-2MHz(tiny12v) 4MHz (tiny12L) 8MHz (tiny12)
Timer	8bit (MTIM)	8bit	8bit	8bit	8bit	8bit
Watchdog	Y	Y	Y	Y	Y	Y
LVI	Y	N	N	N	Y	Y
ACMP	Y	N	N	Y	Y	Y
Opera Voltage	1.8-5.5V	2-5.5V	2-5.5V	2-5.5V	2.7-5.5V	1.8-5.5V(tiny12V) 2.7-5.5V(tiny12L) 4.0-5.5V(tiny12)
Packaging	6or8 pins	6or8pins	6or8 pins	6or8pins	8pins	8pins
Other	ICS	ICP	ICP	ICP	ICP	ICP
1K resale	**0.35$**	0.52$	0.62$	0.7$	0.40$	0.43$

目标市场：

- BLDC FAN
- 小家电
- 温度计
- HBL
- 玩具

批量价格不超过 0.3USD

freescale 飞思卡尔半导体
Contact Window: Johnson zhan (詹志明) Mobile:**13728781297**
E-mail: Johnsonzhan@gmitec.com Website: www.freescale.com.cn
Msn: plczhan@hotmail.com QQ: 747804604

下面我们来看看其一些应用方案，相信可以起到抛砖引玉的作用：

1. 高亮度 LED 的控制

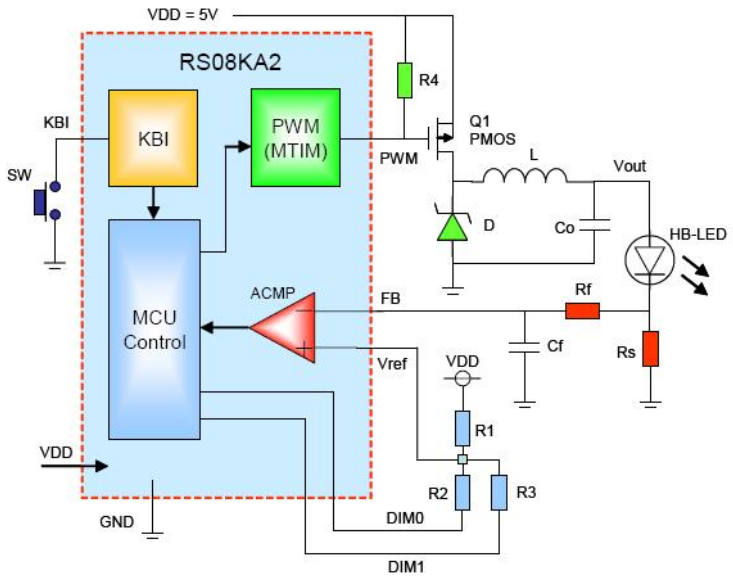
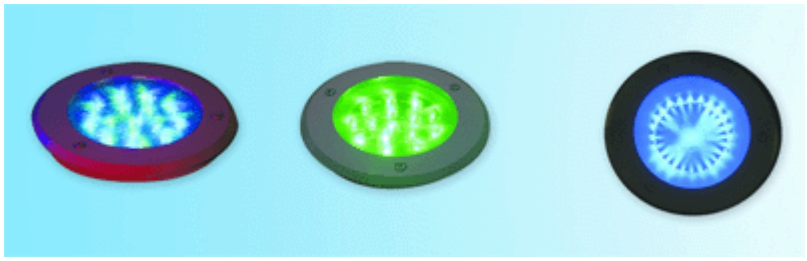
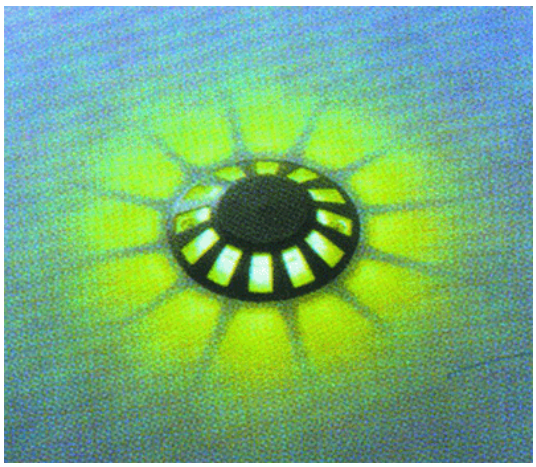


Figure 3-2. Functional Block Diagram

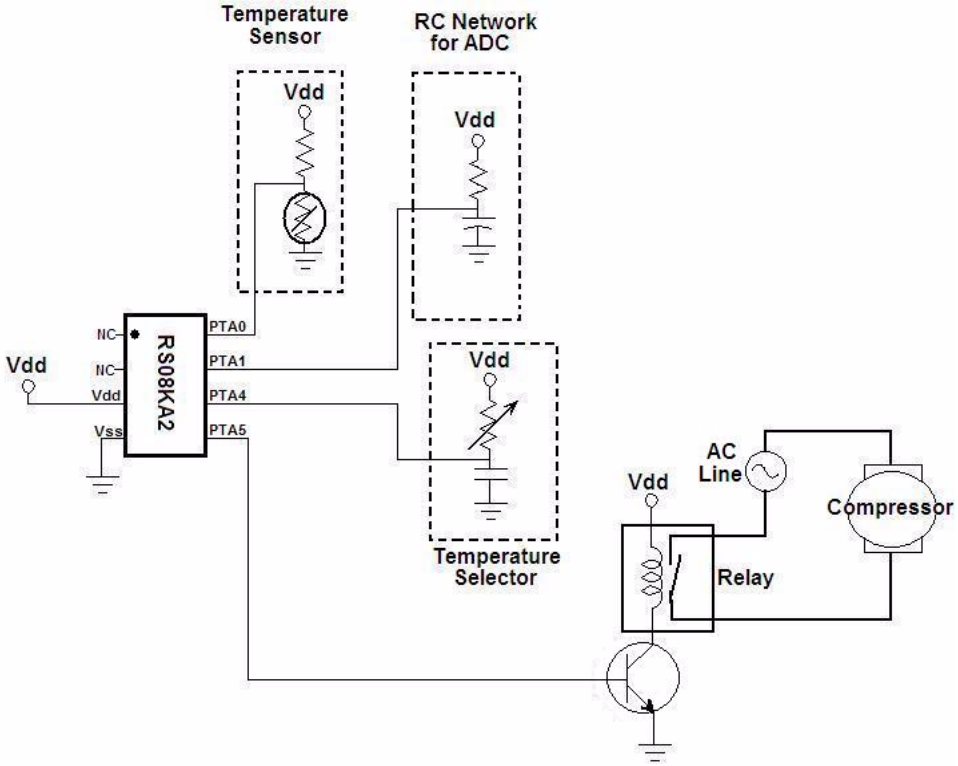
2. 地埋七彩灯





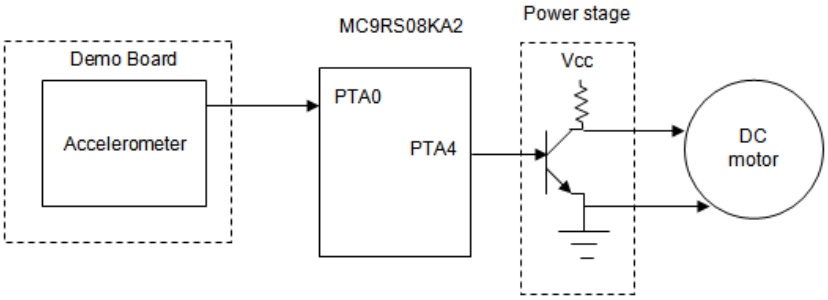
3. BLDE FAN 的控制

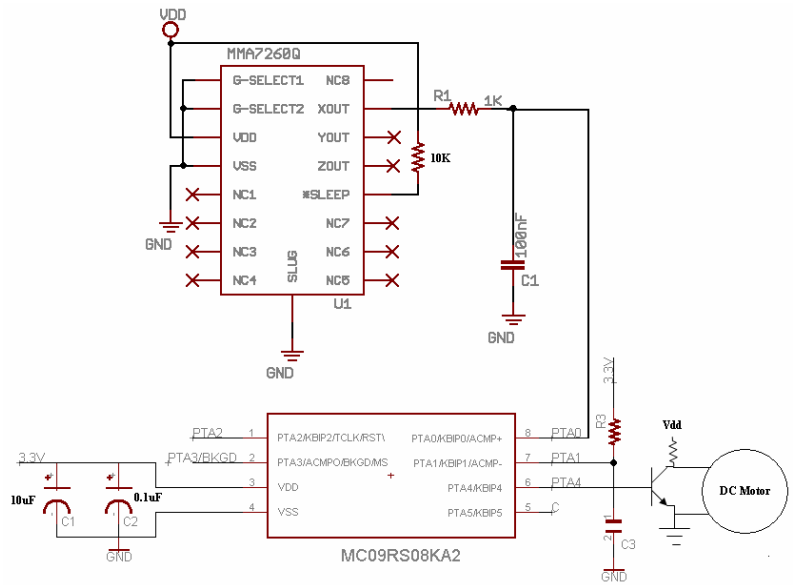
4. 冰箱温度控制



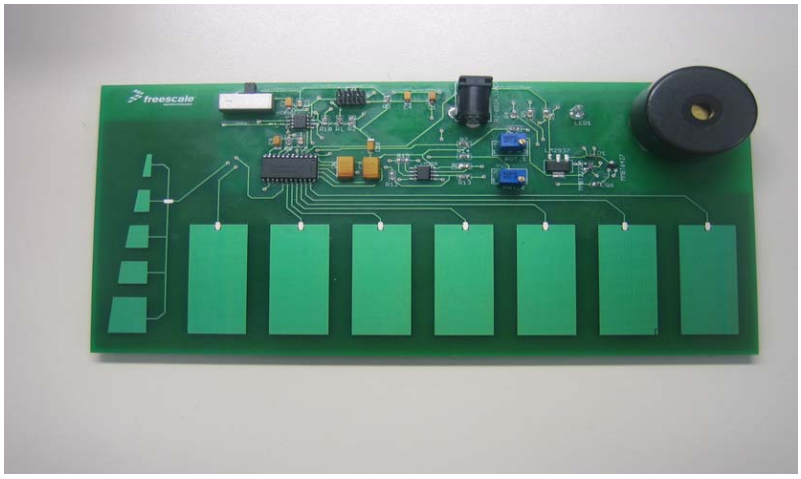
5. 电动缝纫机控制

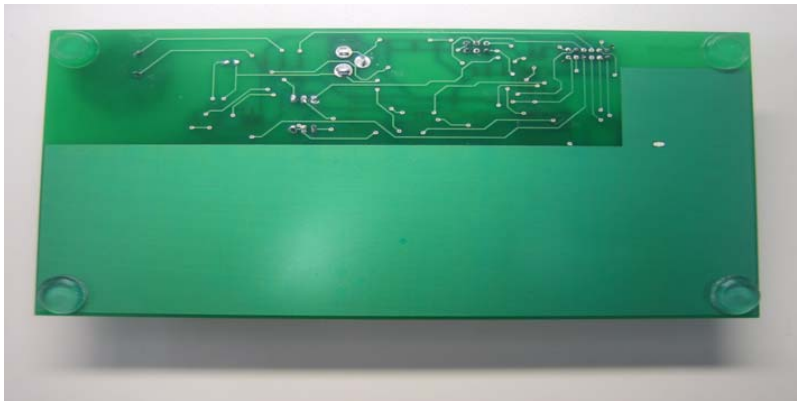

Left side view
Accelerometer
Pedal
Accelerometer
Bottom side view


MC9RS08KA2
Power stage
Demo Board
Accelerometer
PTA0
PTA4
Vcc
DC
motor


VDD
MMA7260Q
G-SELECT1
G-SELECT2
VDD
VSS
NC1
NC2
NC3
NC4
NC8
XOUT
YOUT
ZOUT
*SLEEP
NC7
NC6
NC5
SLUG
U1
R1
1K
10K
100nF
C1
GND
3.3V
R3
Vdd
DC Motor
C3
PTA2
PTA3/BKGD
PTA2/KBIP2/TCLK/RST\
PTA3/ACMPO/BKGD/MS
PTA0/KBIP0/ACMP+
PTA1/KBIP1/ACMP-
PTA4/KBIP4
PTA5/KBIP5
PTA0
PTA1
PTA4
MC09RS08KA2
10uF
0.1uF
C2

6. 电子琴和调光台灯





7. 低价定时器

2007.08.31

High Brightness LED Controller using the MC9RS08KA2

Designer Reference Manual

RS08
Microcontrollers

DRM080
Rev. 0
5/2006

freescale.com

freescale™
semiconductor

High Brightness LED Controller using the MC9RS08KA2

Designer Reference Manual

by: Dennis Lui, Vincent Ko
Freescale Semiconductor, Inc.
Hong Kong

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 that you have the latest information available, refer to http://www.freescale.com

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

Date	Revision Level	Description	Page Number(s)
05/2006	0	Initial release	N/A

Table of Contents

Chapter 5
Testing

Chapter 6
Demo Setup

Appendix A.
Schematic

Appendix B.
Program Listing

Chapter 1
Introduction

1.1 Introduction

This document describes a reference design of a high brightness LED control solution using the MC9RS08KA2 microcontroller.

Recently, light emitting diodes (LEDs) have become very popular in general lighting area as a replacement technology for halogen low-voltage lighting. Customers are quickly recognizing the advantages of using LED lighting, which include long operating life, no fragile glass, no mercury, and low voltage DC operated.

In general, LEDs have a nonlinear I-V behavior and thus current limitation is required to prevent the power dissipation to exceed a maximum limit. Thus, the ideal source for LED driving is a constant current source. The concept of this application is a MCU based LED driver with closed-loop current control. A compact LED light source with dimming control is implemented to demonstrate the advantages of using MCU to drive a high brightness LED with different average current settings.

All hardware schematic diagrams and firmware source codes are available as reference materials.

1.2 Features

- High brightness LED driver with 350mA current driving capability
- Control a buck converter to regulate supply voltage to match with different LED forward voltages
- Up to 80% efficiency
- Internally generated PWM switching frequency
- Feedback control on LED forward current through a current sense resistor
- Dimming control by a single button

1.3 System Overview

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

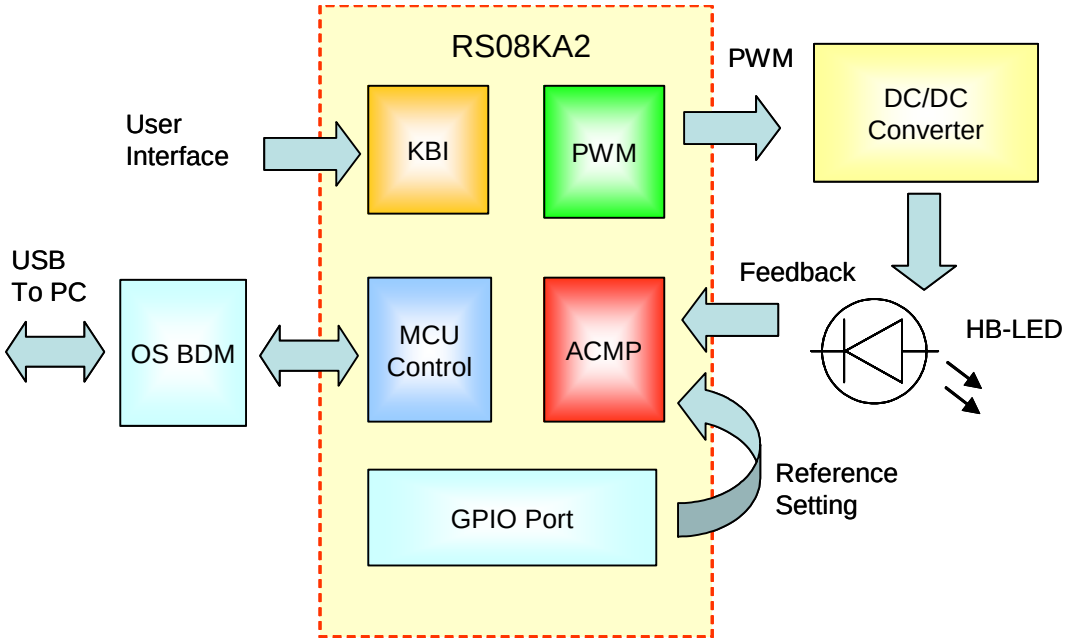


Figure 1-1. System Block Diagram

1.4 Freescale's New Generation Ultra Low Cost MCU

The MC9RS08KA2 (KA2) microcontroller unit (MCU) is an extremely low cost, small pin count device for home appliances, toys, small geometry, and LED control applications. This device is composed of standard on-chip modules including a very small and highly efficient RS08 CPU core, 62 bytes RAM, 2K bytes FLASH, an 8-bit modulo timer, keyboard interrupt, and analog comparator. The device is available in small 6- and 8-pin packages.

MC9RS08KA2 Features:

- Simplified S08 instruction set with added high-performance instructions
- 2048 bytes on-chip FLASH EEPROM
- 62 bytes on-chip RAM
- Internal clock source
- Up to 10-MHz internal bus operation
- Background debug system
- Power-saving modes
- Low-voltage detection
- 8-bit modulo timer
- Analog comparator
- Keyboard interrupt ports

Timer system features include:

- 8-bit up-counter
  - Free-running or 8-bit modulo limit
  - Software controllable interrupt on overflow
  - Counter reset bit (TRST)
  - Counter stop bit (TSTP)

- Four software selectable clock sources for input to prescaler:
  - System bus clock — rising edge
  - Fixed frequency clock (XCLK) — rising edge
  - External clock source on the TCLK pin — rising edge
  - External clock source on the TCLK pin — falling edge
- Nine selectable clock prescale values:
  - Clock source divide by 1, 2, 4, 8, 16, 32, 64, 128, or 256

The analog comparator has the following features:

- Full rail-to-rail supply operation
- Less than 40 mV of input offset
- Less than 15 mV of hysteresis
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output
- Option to compare to fixed internal bandgap reference voltage
- Option to allow comparator output to be visible on a pin, ACMPO
- Remains operational in stop mode

The KBI features include:

- Each keyboard interrupt pin has individual pin enable bit
- Each keyboard interrupt pin is programmable as falling edge (or rising edge) only, or both falling edge and low level (or both rising edge and high level) interrupt sensitivity
- One software-enabled keyboard interrupt
- Exit from low-power modes

Chapter 2
Control Theory

2.1 Switching Regulator

A switching regulator regulates a current flow by chopping up the input voltage and controlling the average current by means of the duty cycle. When a higher load current is required by the load, the percentage of on-time is increased to accommodate the change. There are two basic types of switching regulators: forward-mode regulators and flyback-mode regulators. The name of each type is derived from the way the magnetic elements are used within the regulator.

Forward-mode switching regulators have four functional components: a power switch, a rectifier, a series inductor, and a capacitor (see Figure 2-1). The power switch may be a power transistor or a metal oxide semiconductor field-effect transistor (MOSFET) placed directly between the input voltage and the LC filter section. The shunt diode, series inductor, and shunt capacitor form an energy storage tank whose purpose is to store enough energy to maintain the load voltage and current over the entire off-time of the power switch. The power switch serves only to fill up the energy lost to the load during its off-time. Flyback-mode switching regulators have the same four basic elements as the forward-mode regulators except that they have been rearranged in another configuration. In this design, forward-mode switching is used.

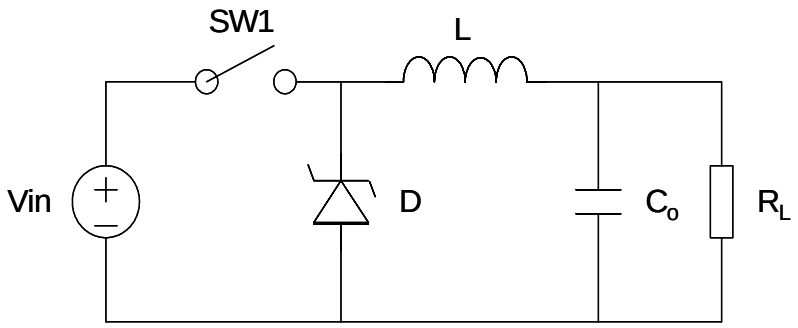


Figure 2-1. Forward Mode Switching Regulator

2.2 Buck Converter Basics

A “buck” or step-down converter is the most elementary forward-mode converter. Its basic schematic can be seen in Figure 2-1.

The operation of this regulator topology has two distinct time periods. The first one occurs when the series switch SW1 is on, the input voltage V_{in} is connected to the input of the inductor L. The output of the inductor is the output voltage, and the rectifier (or catch diode) is reverse biased. During this period, since there is a constant voltage source connected across the inductor, the inductor current begins to linearly ramp upwards, as described by the following equation:

$$I_{L(on)} = [(V_{in} - V_{out}) \times t_{on}]/L \quad \textbf{(EQ 2-1)}$$

During this “on” period, energy is stored within the core material in the form of magnetic flux. If the inductor is properly designed, there is sufficient energy stored to carry the requirements of the load during the “off” period.

The next period is the “off” period of the power switch. When the power switch turns off, the voltage across the inductor reverses its polarity and is clamped at one diode voltage drop below ground by the catch diode. The current now flows through the catch diode thus maintaining the load current loop. This removes the stored energy from the inductor. The inductor current during this time is:

$$I_{L(off)} = [(V_{out} - V_D) \times t_{off}] / L \quad \textbf{(EQ 2-2)}$$

This period ends when the power switch is once again turned on. Regulation of the converter is accomplished by varying the duty cycle of the power switch according to the loading conditions. To achieve this, the power switch requires electronic control for proper operation. It is possible to describe the duty cycle as follows:

$$d = t_{on} / T \quad \textbf{(EQ 2-3)}$$

where T is the switching period.

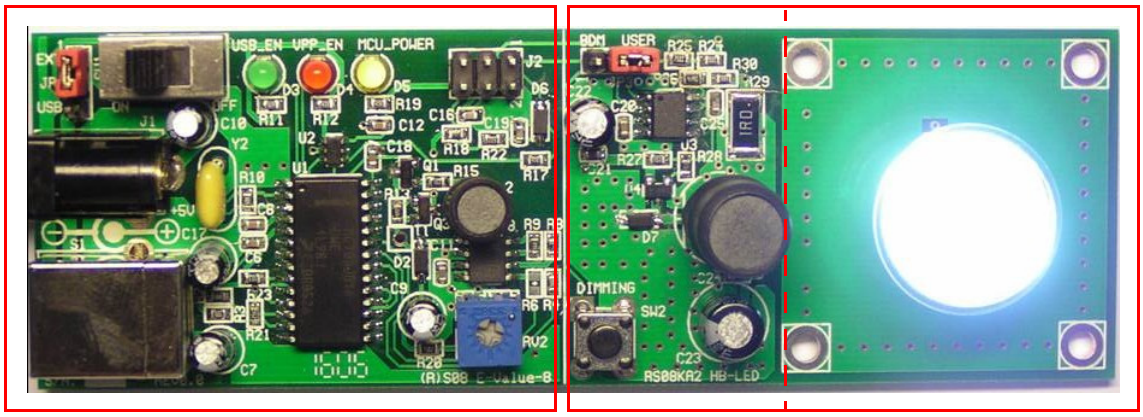
For the buck converter with ideal components, the duty cycle can also be described as:

$$d = V_{out} / V_{in} \quad \textbf{(EQ 2-4)}$$

Chapter 3
Hardware Implementation

3.1 Introduction

The system consists of the 8-pin packaged KA2 and external components. There are four major external components: P-channel MOSFET, schottky diode, inductor, and capacitor which are used for switching regulation.The KA2 is used as a PWM controller to control the HB-LED brightness level. A simple DC to DC step-down converter (buck topology) is implemented to convert the 5V input supply to a current regulated output for LED driving. The integrated open source BDM (background debug mode) design allows users to program the MCU FLASH and debug applications via a USB connection.



OS-BDM **KA2 Control** **HB-LED**

Figure 3-1. High Brightness LED Control Board

3.2 Functional Pin Assignment

- PA0 – Reference Voltage Input
- PA1 – Feedback Input
- PA2 – Push Button Input
- PA3 – Dimming Control DIM0
- PA4 – PWM Output
- PA5 – Dimming Control DIM1

3.3 Hardware Functions

Figure 3-2 shows a block diagram of the HB-LED reference design. The hardware functions can be divided into the following parts:

- PWM Control
- Feedback Loop
- Reference Voltage
- User Interface
- Background Debug Interface

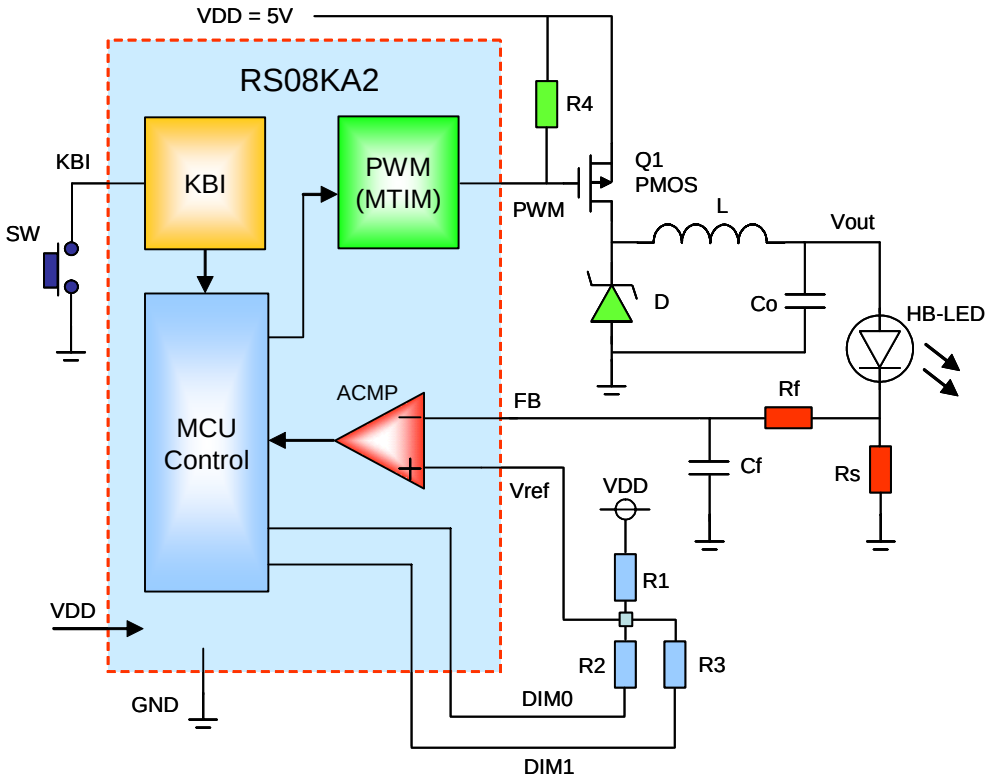


Figure 3-2. Functional Block Diagram

3.3.1 PWM Control

A fixed frequency step-down (buck) DC to DC converter is implemented with the KA2 for usage in high brightness LED driving applications. The KA2 consists of all active functions required to directly implement a step-down converter with a minimum number of external components. The KA2 enables control of PWM on power switch device to regulate the LED current. The PMOS device ON/OFF switching is controlled by MCU timer module and the frequency is set to around 30 kHz. Base the on analog comparator output result, the MCU adjusts the PWM duty cycle in closed-loop to keep the LED driving current as a constant. Since a P-channel MOSFET device is used as the power switch, the PWM control signal should be generated in active low polarity, that means the power switch will turn on if the PWM signal is in LOW level.

A switching cycle is initiated by the falling edge of the PWM signal. At the moment the power switch Q1 is turned on, the inductor is connected to V_{DD} and the inductor current begins to ramp up linearly. When the PWM signal is in HIGH state, the power switch Q1 turns off and the current flows through the diode D to maintain the current loop. The capacitor C_O filters the regulated output and provides the converter loop stability. The converter is designed to operate in continuous conduction mode, which means the inductor current will not decrease to zero before the new cycle start. The average inductor current I_L is almost equal to the average loading current I_O.

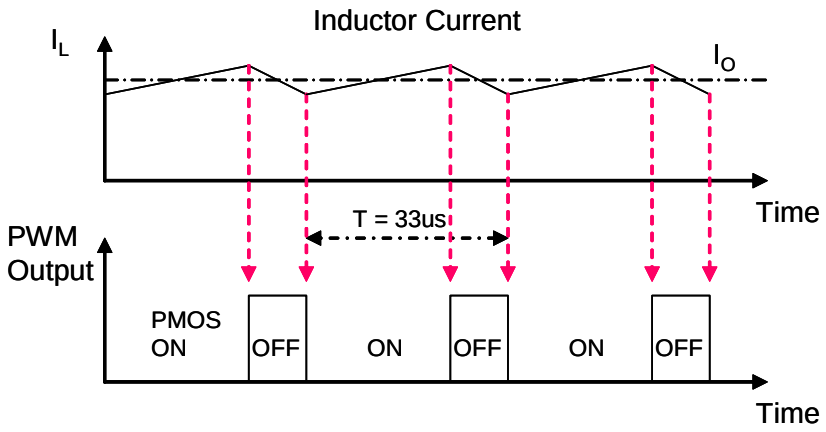


Figure 3-3. PWM Control Waveforms

The external pull-up resistor R4 ensures the PMOS device Q1 is in OFF state during power up period, and prevents a high current surge through the LED before MCU controls the system.

3.3.2 Feedback Loop

The KA2 features an analog comparator which can be used to determine the LED current. The inverting input pin and non-inverting input pin are used for current sense feedback input and reference voltage input respectively.

Current limiting is implemented by monitoring the LED forward current, which is one of the key parameters affecting the performance of LED, particularly on the operating life time, so it is important to control and make sure the average current and peak current are also within the limits specified in the LED specification.

The LED forward current is converted to a voltage by using an external resistor connected in series with the LED. The voltage across the current sense resistor R_S is directly proportional to the current through LED. This feedback voltage is compared against a reference by the analog comparator. When the feedback voltage is higher than the level at reference input, the PMOS switch ON time reduces to limit the current accumulated at the inductor L. In contrast, the PMOS switch ON time increases again to pump more current into the inductor when the feedback voltage is lower than the reference level. The upper and lower threshold levels are defined by the hysteresis range of the comparator.

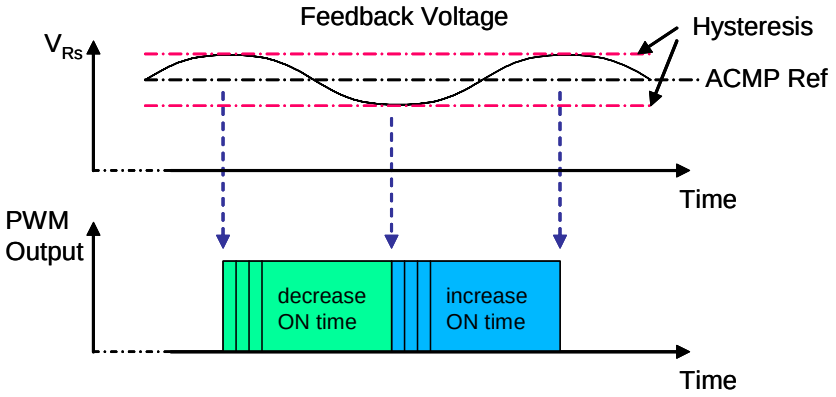


Figure 3-4. Feedback Control Waveforms

In steady-state condition, the LED forward current I_{LED} is equal to the reference voltage V_{REF} divided by the current sense resistor R_S.

The LED forward current is determined by the following equation:

$$I_{LED} = \frac{V_{REF}}{R_S} \quad \textbf{(EQ 3-1)}$$

A simple low-pass filter is added between the current sense resistor R_S and the feedback path to eliminate any high frequency noises coupling into the feedback pin, However, the response time of the filter should be fast enough to track the deviation of the voltage across the current sense resistor R_S.

In this reference design, the PWM switching frequency is around 30kHz and the filter cut-off frequency is set to 5kHz.

The following equation shows the relationship between the filter cut-off frequency and the RC values:

$$f = \frac{1}{2\pi RC} \quad \textbf{(EQ 3-2)}$$

3.3.3 Reference Voltage

The reference voltage in the comparator non-inverting input is determined by a resistor network which consists of three resistors and the connection paths are controlled by the MCU. The pin DIM0 is always configured as output pin and set to output LOW. The resistor R2 connects to ground and generates a reference voltage divided from V_{DD}. This is the default reference voltage and corresponds to 100% brightness level. When the pin DIM1 is set to output LOW, R3 is also connected to ground and in parallel with R2, the reference voltage will be further reduced to a lower level. The DIM1 pin should be configured as input state if it is not used in dimming selection.

The equations for determining the reference voltage with the divider network are:

For High brightness:

$$V_{REF} = V_{DD}\left(\frac{R_2}{R_1 + R_2}\right) \quad \textbf{(EQ 3-3)}$$

For Low brightness:

$$V_{REF} = V_{DD}\left(\frac{R_p}{R_1 + R_p}\right) \quad \textbf{(EQ 3-4)}$$

where R_p is equal to R2 parallel with R3.

3.3.4 User Interface

A single push button, SW, is used as user interface to toggle the LED brightness level between 100% and 40% (default brightness level is 100%). The dimming control is accomplished by changing the LED forward current which is set by the reference voltage applied at the comparator's non-inverting input.

Table 3-1. Dimming Control

Control Pins	100% Brightness	40% Brightness
DIM0	Output LOW	Output LOW
DIM1	Hi-Z	Output LOW

3.3.5 Background Debug Interface

The on-board open source BDM provides a cost effective debug solution for Freescale RS08 MCUs. The module utilizes the modified version of popular open-source BDM solution from the Internet for in-circuit emulation and device programming. The board comprises a 12V DC-DC converter for RS08 FLASH programming support.

NOTE

In circuit debug can not be used for this application program since the BKGD pin is used for dimming control purpose. Use JP3 to switch between BDM and user mode.

3.4 Design Procedures

This section presents guidelines for selecting external components.

3.4.1 High Brightness LED

The system is designed to drive a high brightness LED with typical 350 mA forward current at 3.5V forward voltage, i.e. the power dissipation on the LED is around 1.2W.

3.4.2 Current Sense Resistor

The value of the current sense resistor R_S is determined by two factors: power dissipation on R_S and the threshold level for comparator input. Smaller R_S reduces power dissipation but the detection of feedback signal in comparator is more difficult.

Base on equation (EQ 3-1), setting R_S to 1Ω, the feedback voltage is equal to 350mV.

Power dissipation on R_S is around 120mW, $I^2R = (350mA)^2 \times 1\Omega$, which is reasonable compared to LED power.

3.4.3 RC Network

Base on equations (EQ 3-3) and (EQ 3-4), set V_{DD} = 5V, R1 = 10 kΩ, R2 = 750Ω, R3 = 500Ω.

The reference voltage for high and low brightness settings are equal to 350mV and 146mV respectively. Low brightness setting is corresponds to around 40% brightness level.

3.4.4 Inductor Value

We can determine the inductor value by using the following equations:

$$V_L = L \times \frac{di}{dt} \quad \textbf{(EQ 3-5)}$$

$$L = (V_S - V_O) \times \frac{D \times T}{I_{ripple}} \quad \textbf{(EQ 3-6)}$$

$$L = (5 - 3.85) \times \frac{\frac{3.85}{5} \times \frac{1}{30K}}{0.35 \times 0.25} = 337\mu H \quad \textbf{(EQ 3-7)}$$

where: *V_L is inductor voltage, V_S is source voltage, V_O is output voltage*
D is conduction duty cycle, T is conduction period
I_{ripple} is inductor peak-to-peak ripple current, set to 25% of output current

The inductor current rating must be higher than the maximum peak current flowing through the inductor.

3.4.5 Output Capacitor

Low ESR (equivalent series resistance) aluminium or solid tantalum capacitor is recommended for low output ripple voltage. The ESR of the output capacitor and the inductor ripple current are two major factors contributing to the output ripple voltage.

$$\text{Output ripple voltage} = \text{Inductor ripple current} \times \text{ESR} \quad \textbf{(EQ 3-8)}$$

3.4.6 P-Channel MOSFET

For superior switching performance, a P-channel MOSFET device with low on-state resistance and low gate charge should be selected. The current rating must be at least 1.2 times greater than the maximum load current and the drain-to-source breakdown voltage should be at least 1.25 times the maximum input voltage. The P-channel MOSFET NTR4502 from On Semiconductor is selected in this design.

3.4.7 Schottky Diode Rating

A diode with a high forward voltage drop or long turn-on delay time should not be used, so a fast switching and low drop schottky diode is selected. The current and reverse voltage rating requirements should be similar to the P-channel MOSFET device. The schottky diode MBR130 from On Semiconductor is used in this design.

3.4.8 PCB Layout Guidelines

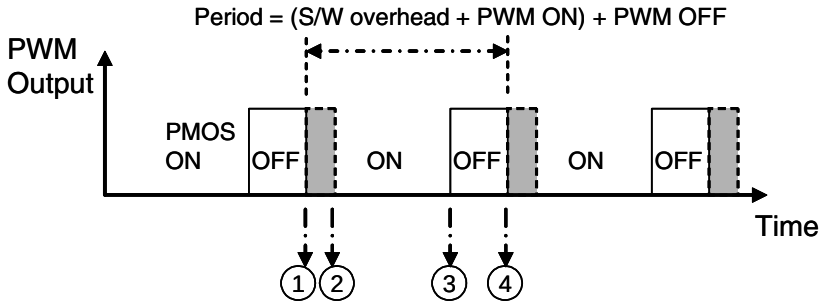
The PCB layout is very important for switching converter design and is critical to reduce noise and ensure specified performance. Care should be taken in PCB layout to avoid rapidly switching currents to generate voltage transients and affect the desired operation.

- Minimize inductance and ground loops, the length of the leads indicated by heavy lines should be kept as short as possible. (e.g. PMOS / coil / schottky diode.)
- Minimize the trace from the current sense resistor to the feedback input and keep it away from noise sources to avoid noise pick up.
- Locate the sensitive voltage reference divider network close to MCU.
- Single ground point or ground plane design should be used.

Chapter 4
Software Implementation

4.1 Introduction

The software is designed with a main loop to monitor the feedback voltage from the LED and generate a PWM controlled waveform for DC to DC switching operation. The PWM timing is constructed by the modulo timer overflow periods of PWM ON time & OFF time and the software overhead using for user interface detection & feedback loop control (see Figure 4-1). The timing used for each branch routine paths should be keep as constant such that the period of the generated PWM is stable and independent of the executed routine path.



1. Set PWM ON; detect KBI, Check ACMP, and adjust ONTime & OFFTime (software overhead)
2. MTIMMOD = ONTime
3. MTIM overflow; set PWM OFF; MTIMMOD = OFFTime
4. MTIM overflow again

Figure 4-1. PWM Timing

4.2 Software Design

This section describes the design of the software blocks. The software description comprises these topics:

- Main Software Flow
- Initialization
- KBI Detection
- Feedback Comparison
- PWM Generation

4.2.1 Main Software Flow

Figure 4-2 shows the flow of the main program, it also indicates the functions that KA2 are to perform. The initialization subroutine initializes the modules of real-time interrupt, modulo timer, analog comparator, and keyboard interrupt.

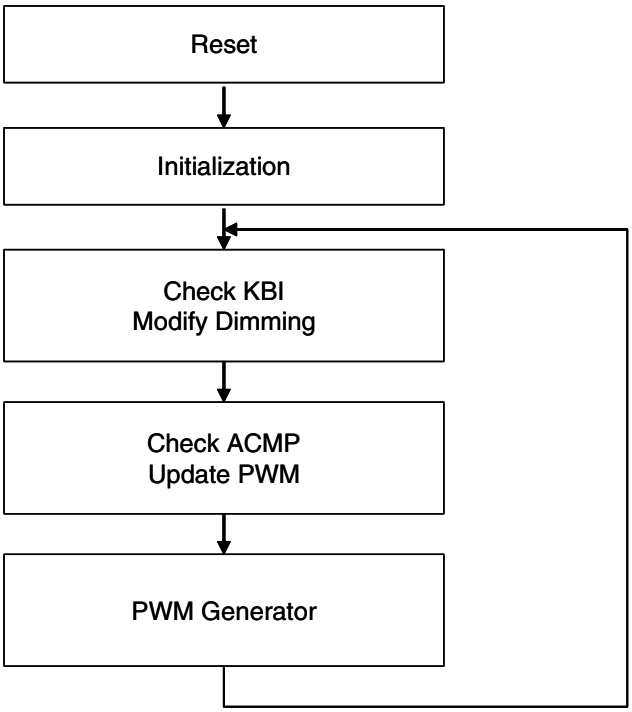


Figure 4-2. Main Software Flow

4.2.2 Initialization

The Initialization routine initializes the MCU with following configurations:

- Initializes ICS and sets bus clock to 10MHz
- Initializes page register, disables COP and BKGD
- Enables LVI, and sets RTI to 128ms
- Initializes RAM variables
- Initializes GPIO
- Initializes KBI
- Initializes the ACMP
- Initializes modulo timer
  - Timer prescalar = 1, timer clock = 10MHz
  - Maximum of period = 25.6μs
  - Timer resolution = 100ns

4.2.3 KBI Detection

The state of the user interface is scanned every 128ms to see if any button press event has occurred. When a key press event is detected, it sets the corresponding “Pressed” flag. The output state of DIM1 pin will toggle to setup a new reference voltage for ACMP in the next 128ms period. This is to set up a key debounce period of 128ms.

The real-time interrupt function is used to generate a periodic interrupt for KBI scanning. The RTI is driven from the 1kHz internal clock reference and the wake-up period is set to 128ms.

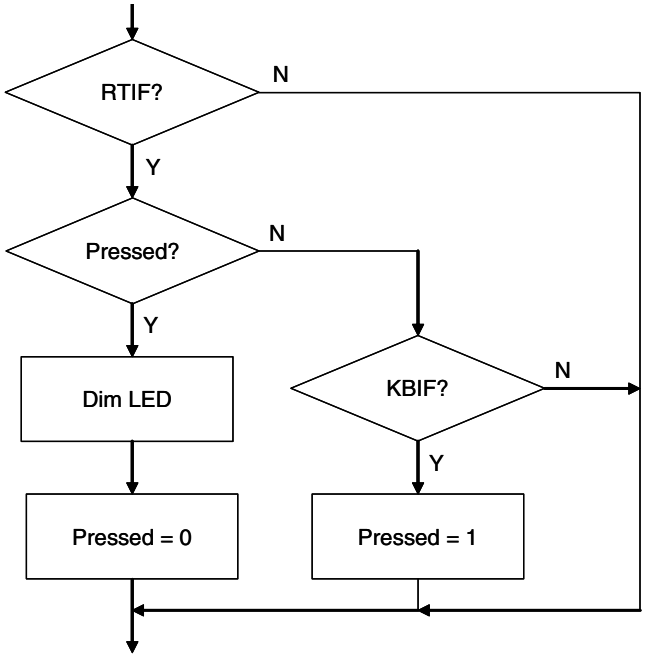


Figure 4-3. KBI Detection Routine

4.2.4 Feedback Comparison

The function of the feedback comparison routine is to control the PWM duty cycle periodically. When it detects a crossover point between the feedback voltage and the reference voltage, the ONTime and OFFTime variables will be updated according to the result of the analog comparator output that is used as a feedback signal to indicate whether the LED current is higher or lower than the expected level.

These two variables should be adjusted in opposite polarity such the overall period is kept constant (i.e. OFFTime = 255 – ONTime). The PWM duty cycle is determined by the value of ONTime. The variables ONTime and OFFTime should be set in the range of 0 to 255.

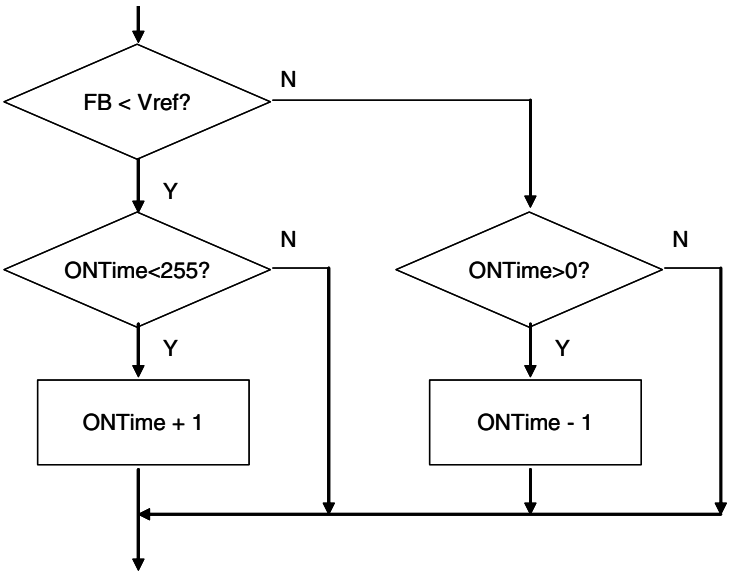


Figure 4-4. Feedback Comparison Routine

4.2.5 PWM Generation

The 8-bit modulo timer is configured to generate the PWM signal. The timer is running in modulo mode such that the overflow flag will be set when the count value matches the modulo value in the MTIM module register, MTIMMOD. PWM output is implemented by programming two timer overflow variables, ONTime and OFFTime alternately into the MTIM module register.

The overall PWM period is equal to the sum of PWM ON, PWM OFF and the time used for software looping in KBI detection and feedback comparison routines.

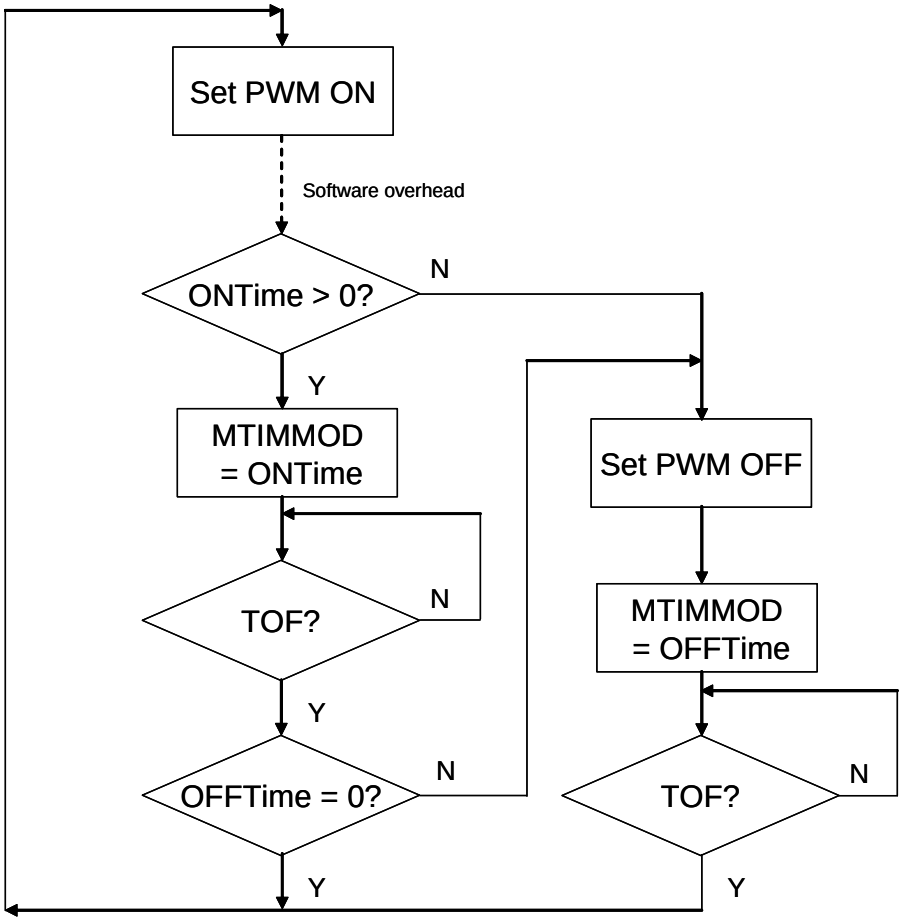


Figure 4-5. PWM Generation Routine

Chapter 5
Testing

The reference design was tested with Lumileds high brightness LED in typical conditions. The LED forward current was regulated with expected performance with 5V supply input. Dimming control was achieved by changing the LED current with a single button user interface.

5.1 Measurement Data

Figure 5-1 shows the voltage waveform measured across the current sense resistor R_S in 100% brightness setting. The LED forward current was determined by this voltage divided by one (R_S = 1Ω), which was equal to around 385mA and the peak-to-peak ripple current was 26mA. The measurements show that the LED forward current was regulated and tracking to the reference input accurately. The difference between the calculated target (350mV) and measured voltage (385mV) showing in Figure 5-2 is due to the tolerance of resistors used.

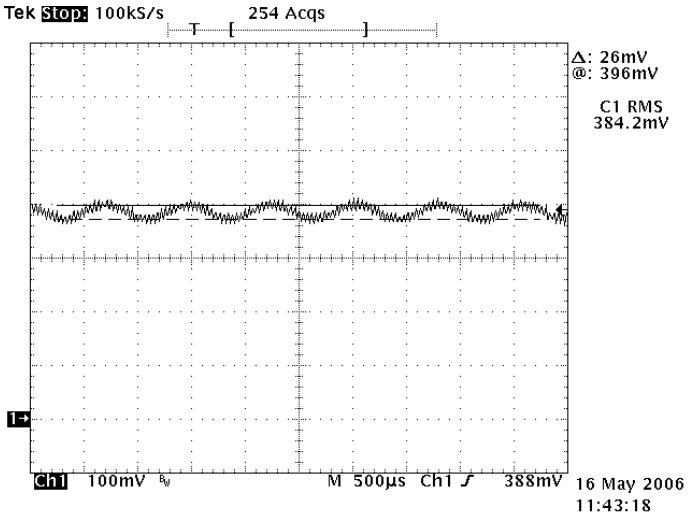


Figure 5-1. Waveform at R_S

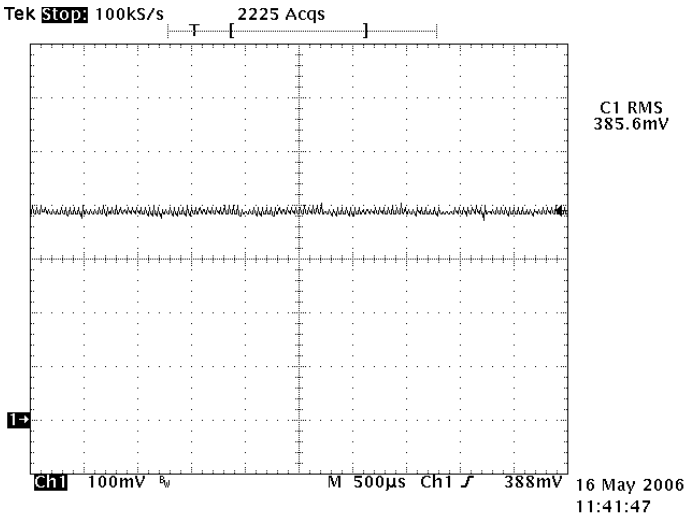


Figure 5-2. Waveform at Reference Input

The efficiency of the whole converter was measured as:

1. Supply input = 5V, supply current = 302mA, DC to DC output voltage = 3.5V
2. Voltage across R_S = 385mV, LED forward voltage = 3.115V

$$\text{Efficiency} = \frac{\text{Output Power}}{\text{InputPower}} \quad \textbf{(EQ 5-1)}$$

$$\text{Efficiency} = \frac{3.115 \times 0.385}{5 \times 0.302} = 79.4\% \quad \textbf{(EQ 5-2)}$$

3. Overall efficiency = 79.4%

5.2 Customization

5.2.1 Hardware

The on-board open source BDM is used for development purposes only. It can be removed to make the design more compact and fit into a smaller housing.

The inductor value can be adjusted to match with new LED forward current requirement.

Chapter 6
Demo Setup

6.1 Introduction

This section shows how to setup the MC9RS08KA2 HB-LED demo board and run the included demo program. Users can use the on-board open source BDM module to program the MCU FLASH and debug new applications via the USB connection.

6.2 Default Settings

Figure 6-1 shows the default settings for the demo board. The black blocks indicate the default position of the jumpers. Please check these settings before continuing.

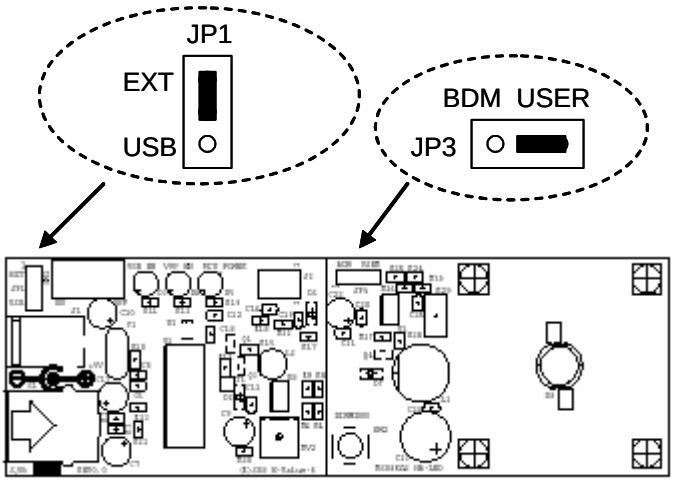


Figure 6-1. Demo Board Default Settings

6.3 Programming and Running the HB-LED Demo

Further application development and debug for the MC9RS08KA2 is supported through the on-board OS-BDM interface. A USB type B connector (S1) provides the connection between the demo board and your host PC. This tool is tested to run on Windows XP with Freescale CodeWarrior version 5.1. Use the procedure below to operate the HB-LED demo board.

The board must be powered by external 5V supply when HB-LED is turned on. Do not use the USB 5V for powering the HB-LED.

Program:

1. Move jumper JP3 to the 1-2 position “BDM” to power up device in background debug mode.
2. Programme the device with the corresponding software via the USB connection. Refer to documentation in the CD-ROM that came with the HB-LED demo board.

NOTE

In-circuit debugging cannot be used for this application program since the BKGD pin is used for dimming control purpose. Use JP3 to switch between BDM and user mode.

Run:

1. Move jumper JP3 to the 2-3 position “USER” to isolate the “BDC_BKGD” pin.
2. Power up the device for software execution in user mode.

NOTE

All RS08 devices require a power-on-reset to switch between user mode and background debug mode.

6.4 Troubleshooting

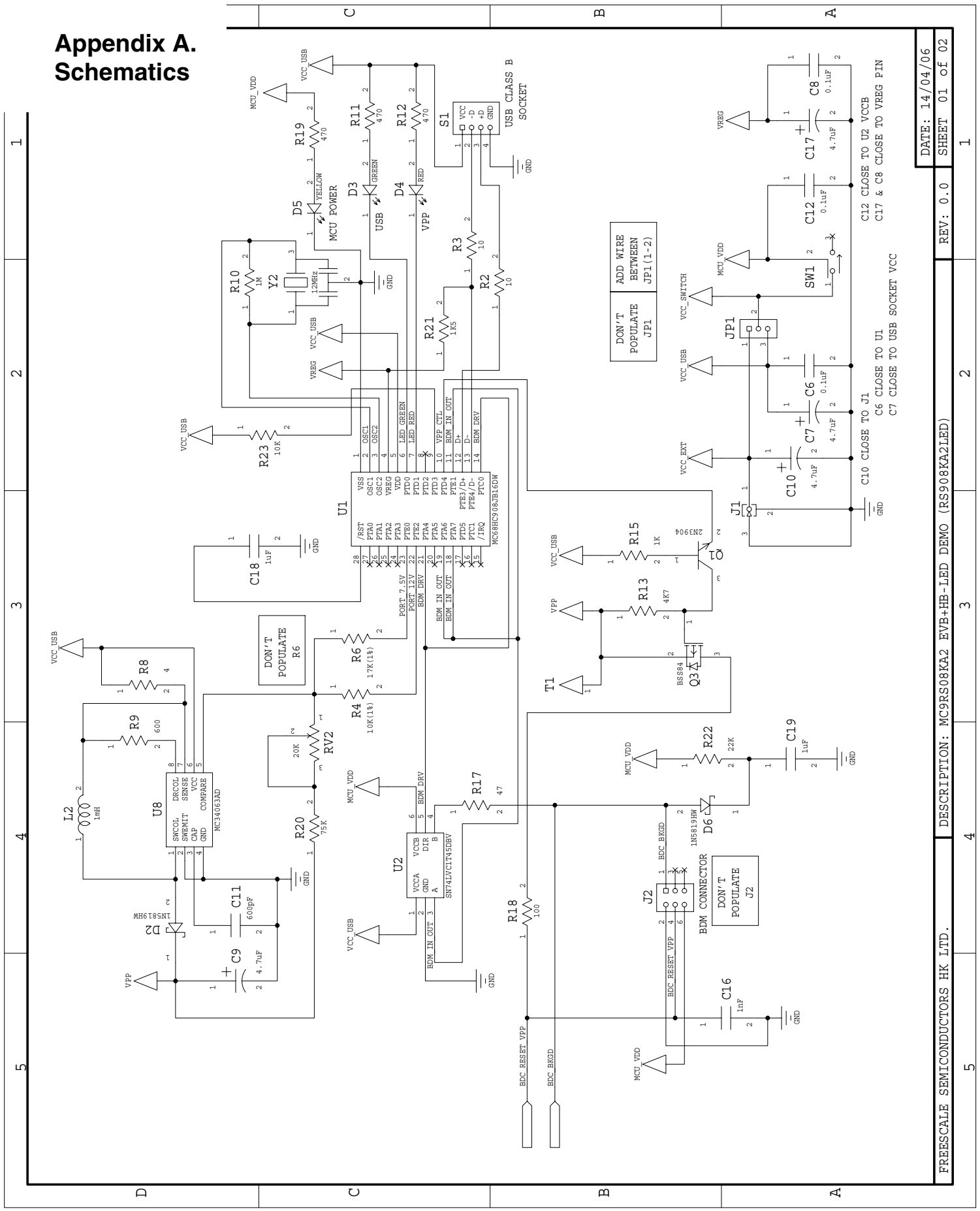
The high brightness LED does not turn on:

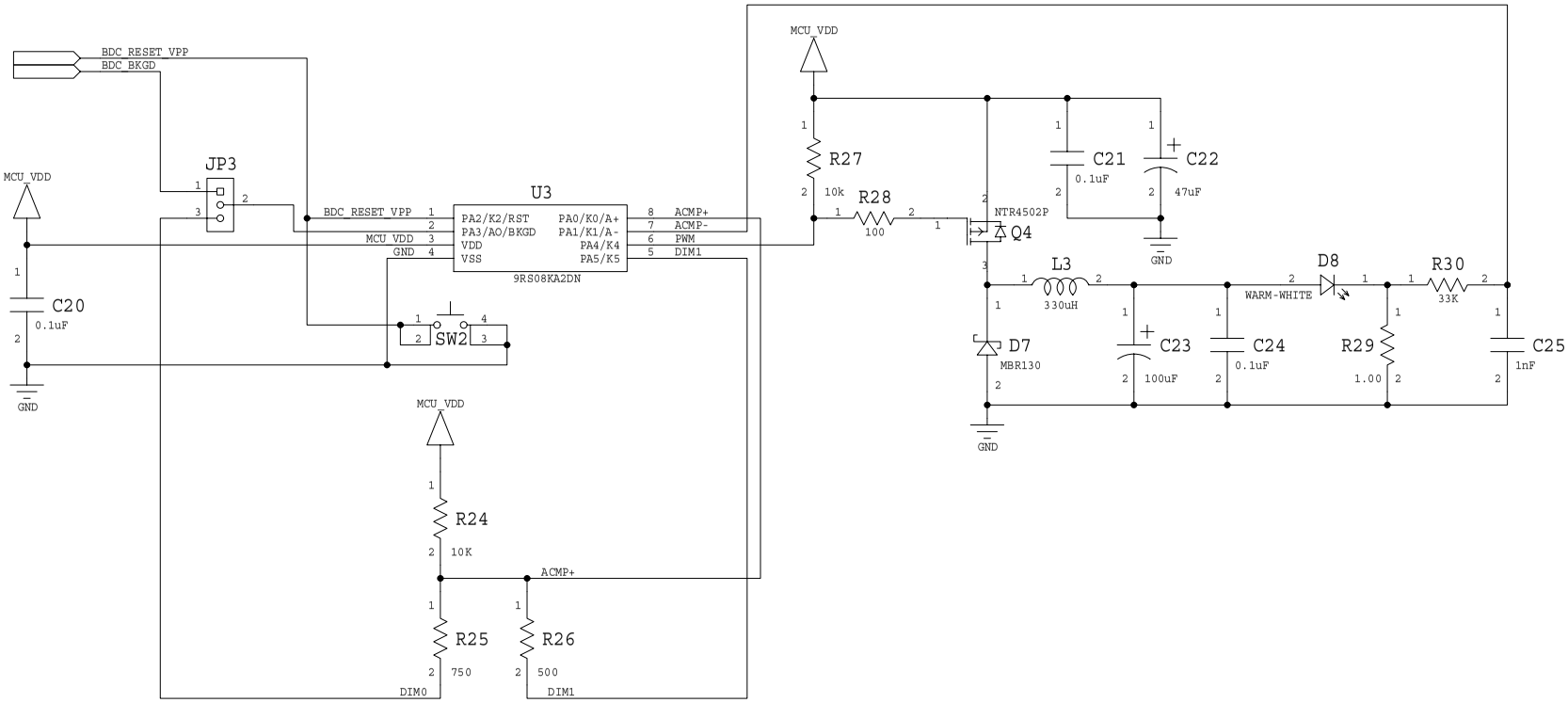
- Make sure jumper JP3 is set to “USER” (2-3) position.
- Make sure supply input selection (JP1) is selected for external power supply.

Unable to program the MCU using the on-board OS-BDM:

- Make sure jumper JP3 is set to “BDM” (1-2) position.

Appendix A. Schematics





FREESCALE SEMICONDUCTORS HK LTD.	DESCRIPTION: MC9RS08KA2 EVB+HB-LED DEMO (RS908KA2LED)	REV: 0.0	DATE: 14/04/06 SHEET 02 of 02

Appendix B.
Program Listing

```
;*************************************************************
;
; (c) Copyright Freescale Semiconductor, Inc. 2006
; ALL RIGHTS RESERVED
;
;*************************************************************
;*************************************************************
;* HB LED Coding for 9RS08KA2
;*
;* Author:    Vincent Ko
;* Date:      Apr 2006
;*
;* PTA0/KBI0/ACMP+             LED Driver
;* PTA1/KBI1/ACMP-             Voltage Reference
;* PTA2/KBI2/TCLK/RESETb/VPP   Dimming Button
;* PTA3/ACMPO/BKGD/MS          DIM0
;* PTA4/KBI4                   PWM
;* PTA5/KBI5                   DIM1
;*
;*************************************************************
; include derivative specific macros
       XDEF   Entry

       include "MC9RS08KA2.inc"

;===============================================================================
; ICS Definition
;===============================================================================
ICS_DIV_1     equ     $00
ICS_DIV_2     equ     $40
ICS_DIV_4     equ     $80
ICS_DIV_8     equ     $c0

;===============================================================================
; MTIM Definition
;===============================================================================
MTIM_DIV_1    equ     $00
MTIM_DIV_2    equ     $01
MTIM_DIV_4    equ     $02
MTIM_DIV_8    equ     $03
MTIM_DIV_16   equ     $04
MTIM_DIV_32   equ     $05
MTIM_DIV_64   equ     $06
MTIM_DIV_128  equ     $07
MTIM_DIV_256  equ     $08

MTIM_BUS_CLK          equ   $00
MTIM_XCLK             equ   $10
```

```
MTIM_TCLK_FALLING      equ   $20
MTIM_TCLK_RISING       equ   $30

;===============================================================================
; ACMP Definition
;===============================================================================
ACMP_OUTPUT_FALLING    equ   $00
ACMP_OUTPUT_RAISING    equ   $01
ACMP_OUTPUT_BOTH       equ   $03

;===============================================================================
; RTI Definition
;===============================================================================
RTI_DISABLE    equ      $00
RTI_8MS        equ      $01
RTI_32MS       equ      $02
RTI_64MS       equ      $03
RTI_128MS      equ      $04
RTI_256MS      equ      $05
RTI_512MS      equ      $06
RTI_1024MS     equ      $07

;===============================================================================
; Application Definition
;===============================================================================
VREF           equ     PTAD_PTAD0
mVREF          equ     mPTAD_PTAD0
LED            equ     PTAD_PTAD1
mLED           equ     mPTAD_PTAD1
BUTTON         equ     PTAD_PTAD2
mBUTTON        equ     mPTAD_PTAD2
DIM0           equ     PTAD_PTAD3
mDIM0          equ     mPTAD_PTAD3
PWM            equ     PTAD_PTAD4
mPWM           equ     mPTAD_PTAD4
DIM1           equ     PTAD_PTAD5
mDIM1          equ     mPTAD_PTAD5

;===============================================================================
; Application Macro
;===============================================================================
Delay_1_cycle: macro
       nop             ;1 cycles
       endm
Delay_2_cycles: macro
       tsta            ;2 cycles
       endm
Delay_3_cycles: macro
       brn             ;3 cycles
       endm
Delay_4_cycles: macro
       brn             ;3 cycles
       nop             ;1 cycles
```

```
       endm
Delay_5_cycles: macro
       tstx            ;5 cycles
       endm
Delay_10_cycles: macro
       tstx            ;5 cycles
       tstx            ;5 cycles
       endm
Delay_15_cycles: macro
       tstx            ;5 cycles
       tstx            ;5 cycles
       tstx            ;5 cycles
       endm

       org    TINY_RAMStart
; variable/data section
ONTime        ds.b 1
Pressed       ds.b 1

       org    RAMStart
; variable/data section

       org    ROMStart
; code section
main:
Entry:
;-----------------------------------------------------
; Config ICS
; Device is pre-trim to 20MHz ICLK frequency
; TRIM value are stored in $3FFA:$3FFB
;-----------------------------------------------------
       mov    #HIGH_6_13(NV_ICSTRM), PAGESEL
;      mov    MAP_ADDR_6(NV_FTRIM), ICSSC   ; $3FFB
;      mov    MAP_ADDR_6(NV_ICSTRM), ICSTRM ; $3FFA
       mov    #ICS_DIV_1, ICSC2             ; Use 10MHz

;-----------------------------------------------------
;Config System
;-----------------------------------------------------
       mov    #HIGH_6_13(SOPT), PAGESEL     ; Init Page register
       mov    #(mSOPT_COPT|mSOPT_STOPE), MAP_ADDR_6(SOPT); BKGD disable,COP disabled
       mov    #(mSPMSC1_LVDE|mSPMSC1_LVDRE), MAP_ADDR_6(SPMSC1); LVI enable
       mov    #(RTI_128MS), MAP_ADDR_6(SRTISC); 128ms RTI

;-----------------------------------------------------
; Init RAM
;-----------------------------------------------------
clr    ONTime
clr    Pressed

;-----------------------------------------------------
; Config GPIO
; PWM - init H
```

```
;-------------------------------------------------------
       mov    #(mPWM), PTAD                ; Initial port
       mov    #(mDIM0|mPWM), PTADD         ; DIM0 and PWM Output pins
;-------------------------------------------------------
;Config KBI
;-------------------------------------------------------
       lda    #mBUTTON
       sta    MAP_ADDR_6(PTAPE)            ;Enable Pullup
       sta    KBIPE                        ;KBI Enable
       bset   KBISC_KBACK, KBISC

;-------------------------------------------------------
;Config ACMP
;-------------------------------------------------------
       mov    #(mACMPSC_ACME), ACMPSC       ; Enable ACMP

;-------------------------------------------------------
;Config MTIM
;
;Timer prescalar=1 -> Timer clk = 10MHz
;Bus = 10MHz
;Max OF period = 25.6us
;Timer resolution = 100ns
;-------------------------------------------------------
       mov    #(MTIM_BUS_CLK|MTIM_DIV_1), MTIMCLK
       mov    #255, MTIMMOD

;-------------------------------------------------------
;PWM Control Loop
;-------------------------------------------------------
PWMControlLoop:

       bclr   PWM, PTAD                    ; Switch close

; --- Check Button ---
       brset SRTISC_RTIF, MAP_ADDR_6(SRTISC), CheckPress;5
       Delay_10_cycles
       Delay_10_cycles
       Delay_4_cycles
       bra    CheckACMP                    ;3
CheckPress:
       bset   SRTISC_RTIACK, MAP_ADDR_6(SRTISC);5
       brset  0, Pressed, ToggleDim        ;5
CheckButton:
       brset KBISC_KBF, KBISC, ButtonPressed;5
       Delay_5_cycles
       Delay_4_cycles
       bra    CheckACMP                    ;3
ButtonPressed:
       bset   KBISC_KBACK, KBISC           ;5
       inc    Pressed                      ;4
       bra    CheckACMP                    ;3
ToggleDim:
```

```
       bset   KBISC_KBACK, KBISC            ;5
       lda    PTADD          ;3
       eor    #mDIM1         ;2
       sta    PTADD          ;2
       clr    Pressed        ;2
       Delay_3_cycles

; --- Check ACMP ---
CheckACMP:
       lda    ONTime                        ;3
       brset  ACMPSC_ACO, ACMPSC, LowerThanRef;5
HigherThanRef:
       cbeqa  #0, DummyCycles               ;4
       deca                                 ;1
       bra   PWMGen                         ;3
LowerThanRef:
       cbeqa  #255, DummyCycles             ;4
       inca                                 ;1
       bra    PWMGen                        ;3
DummyCycles:
       nop                                  ;1
       bra    PWMGen                        ;2
PWMGen:
       sta    ONTime                        ;2
       bne    EnableOnTime                  ;3
       Delay_10_cycles
       Delay_3_cycles
       bra    EnableOffTime                 ;3
EnableOnTime:
       sta    MTIMMOD                       ;2 ON period
       mov    #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC;4 Reset and Start Timer
       lda    #255           ;2
       sub    ONTime         ;3
       wait                                 ;2+
       beq    ZeroOffTime                   ;3 Duty cycle is off
EnableOffTime:
       bset   PWM, PTAD                     ;5 Switch open
       sta    MTIMMOD                       ;2 OFF period
       mov    #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC;4 Reset and Start Timer
       wait                                 ;2+
       jmp   PWMControlLoop                 ;3
ZeroOffTime:
       Delay_10_cycles
       Delay_3_cycles
       jmp    PWMControlLoop                ;3

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Reset Vector
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       org    $3ffc
Security:
       dc.b   $FF
       jmp    main
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

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

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

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

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

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

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

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

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

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

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

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

DRM080
Rev. 0, 5/2006

freescale™
semiconductor

Variable Speed DC Fan Control using the MC9RS08KA2

Designer Reference Manual

RS08
Microcontrollers

DRM079
Rev. 0
5/2006

freescale.com

freescale™
semiconductor

Variable Speed DC Fan Control using the MC9RS08KA2

Designer Reference Manual

by: Vincent Ko
Freescale Semiconductor, Inc.
Hong Kong

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 that you have the latest information available, refer to http://www.freescale.com

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

Date	Revision Level	Description	Page Number(s)
05/2006	0	Initial release	N/A

Table of Contents

Chapter 1
Introduction

1.1 Introduction

This document describes the implementation of a DC brushless fan controller using the Freescale ultra low cost MC9RS08KA2 8-bit microcontroller (MCU). The design contains a temperature sensor the MCU reads with control on fan speed against the ambient temperature. Complete coding and schematic are included.

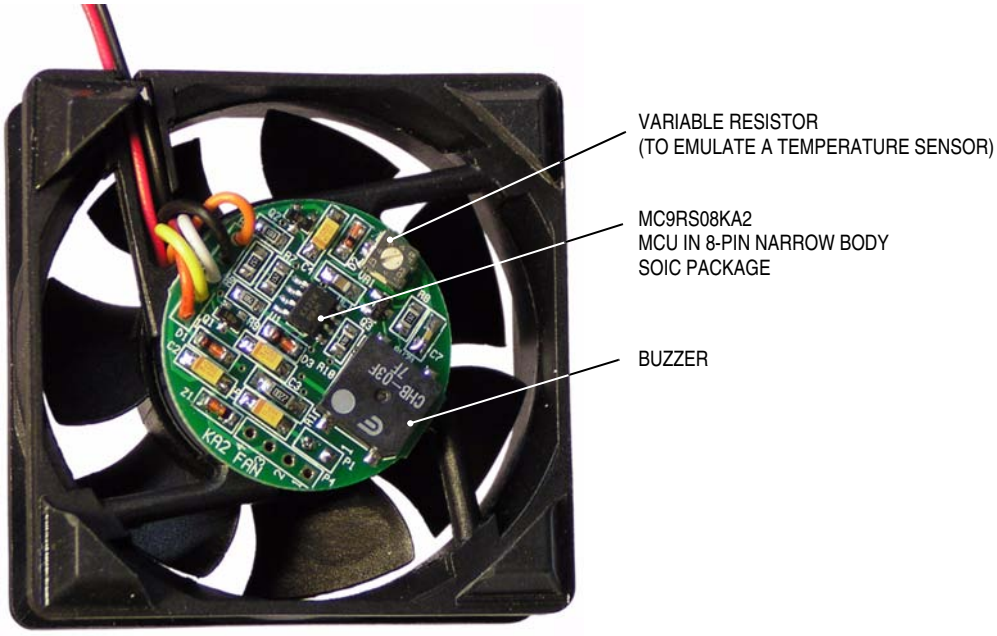


Figure 1-1. The MC9RS08KA2 DC Fan Reference Design

The DC fan used is a brushless DC motor fan. It is widely used in chip cooling or system ventilation applications. In the market, most of the DC fans are of the constant air flow design. As the high performance electronic products continue to increase, cooling requirement becomes more and more sophisticated. MCU approach provides a cost effective solution to this application. There are several advantages of a MCU based design over traditional solutions.

1. Instead of having a constant air flow the MCU provides enough processing power to modify the fan speed according to environment changes such as the temperature of the target system.
2. Fault detection can easily be implemented by the MCU. For example, the MCU can detect for the air flow blocking or motor jam, the motor driver can be stopped completely to avoid further damage.
3. Buzzer alarm or digital output acknowledgement can be generated under the faulty situation.

The MCU chosen for this purpose must be low cost and it must provide small geometry package to integrate into the fan controller printed circuit board (PCB). The MC9RS08KA2 is ideal for this application.

1.2 Freescale's New Generation Ultra Low Cost MCU

The MC9RS08KA2 microcontroller unit (MCU) is an extremely low cost, small pin count device for home appliances, toys, and small geometry applications, such as a DC fan controller. This device is composed of standard on-chip modules including a very small and highly efficient RS08 CPU core, 62 bytes RAM, 2K bytes FLASH, an 8-bit modulo timer, keyboard interrupt, and analog comparator. The device is available in small 6- and 8-pin packages.

Features of the MC9RS08KA2 include:

- 8-bit RS08 core
  - Up to 10 MHz (bus frequency) at 1.8V for 100 ns minimum instruction time
  - RS08 instruction set
  - Supports tiny/short address mode
  - 14-byte fast-access RAM
  - Allows emulation of HC08/HCS08 zero-offset index addressing mode instructions
- Third-generation Flash and RAM (extremely fast, byte writable programming)
  - 63 Byte RAM
  - 2K Byte Flash
- Flexible clock options
- 4 Bidirectional I/O lines with software selectable pull-up (eliminates need for external resistors)
- Analog comparator
- Real time interrupt
- 8-bit timer with 8-bit prescale
- System protection
  - Resets in instance of runaways or corrupted code
  - Low voltage detection
  - Illegal opcode and illegal address detection
  - Flash security feature
- Single wire debugging and emulation interface; eliminates need for expensive emulation tools or development hardware

1.3 DC Fan Reference Design Targets

Table 1-1. Design Targets

Item	Requirement
Motor Type	Bi-phase BLDC motor
Fan Dimensions	60mm x 60mm x 25mm
Operating Voltage	12V
Current Rating	0.18A (max.)
Speed	1000 to 4000 RPM
Temperature Feedback	Yes
Fault Detection	Air flow blocking (motor jam)
Fault Notification	Buzzer alarm

1.4 Bi-Phase BLDC Motor

The brushless DC motor (BLDC) design for DC fan is commonly consist of a permanent magnet attached on the rotor and the stator phase coil windings are mounted on the motor shaft as illustrated in Figure 1-2. The BLDC has no brushes on the rotor and the commutation is performed electronically at certain rotor positions.

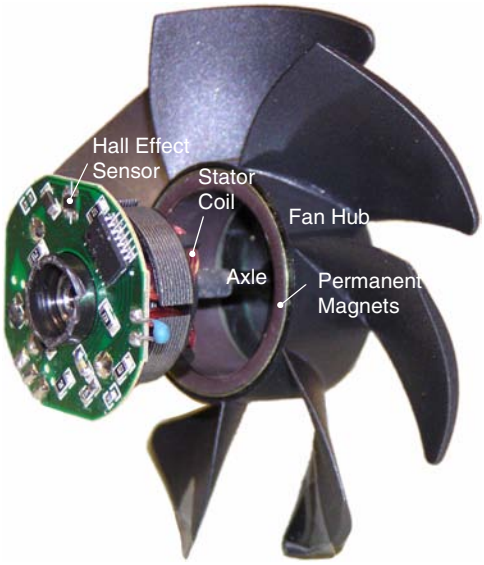


Figure 1-2. Bi-Phase BLDC Motor Diagram

Chapter 2
Motor Control

2.1 Commutation

The typical bi-phase BLDC has one pole-pair per phase. Each commutation rotates the rotor by 90 degrees and four commutation steps complete a mechanical revolution. Each pole-pair is implemented by two coils, with four coils in total for a bi-phase motor. Energizing a pair of coils, either coil A & C or coil B & D as shown in Figure 2-1, induces magnetic fields that push the equal polarity rotor magnets away from the energized coils and at the same time the opposite polarity rotor magnets are pulled toward the coils. Rotation starts and this is called a commutation step. When the rotor magnetic pole is aligned with the energized coils, the coils are deactivated and the previously un-energized pair of coils are then energized. As the magnetic field switches to the next motor position or pole, the inertia of the rotor keeps the motor running. As a result, two commutation steps moves the rotor by 180 degrees or one motor phase. One mechanical revolution is contributed by four commutation steps.

To avoid conflict to the magnetic field, adjacent coils cannot be energized at the same time. Dead-time, where all coils are un-energized must be added between each commutation step.

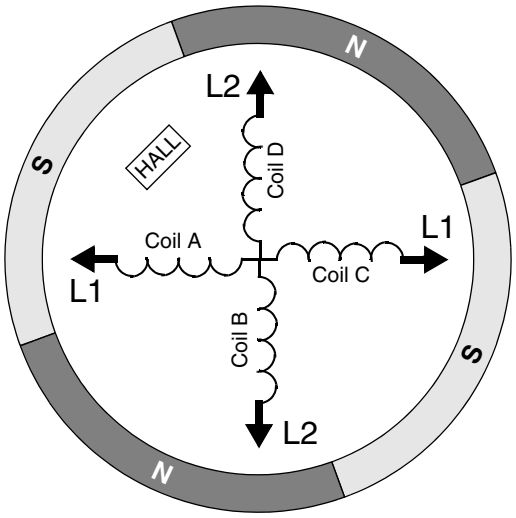


Figure 2-1. Bi-phase BLDC Motor Schematic

2.2 Rotor Position Control

The key idea to prevent a motor lockup concerns rotor position detection. The time to switch the commutation is critical. Energizing coil-pair for too long will kill the rotor inertia and the motor stops running. This is called motor lockup. Switching the commutation too soon will lose control to the rotor and eventually stall the motor. The rotor position in this design is determined by a hall sensor which will respond to the change in magnetic field. Hall sensor output toggles when the magnetic field changes its polarity. Positioning the hall sensor between the coils at 45 degree to the stator coils, as shown in Figure 2-1, can effectively detect the rotor position. In this case the hall sensor output toggles when the rotor magnets is aligned to the coils. Commutation should switch at this time from one coil-pair to the next coil-pair.

2.3 Commutation Waveforms

In general, in a bi-phase motor design, alternate coils are tied together and give a single connection to the driver. In this design, the driver connection for coil A and coil C is called L1 (see Figure 2-1). Similarly, the driver connection for coil B and coil D is called L2. Driving to either of the connections will energize a coil-pair. The commutation waveform is shown in Figure 2-2. The coil driving period is aligned with the Hall sensor output. When the sensor output toggles, coil driving is stopped, the coils are de-energized for a period of time before the next coil-pair is energized.

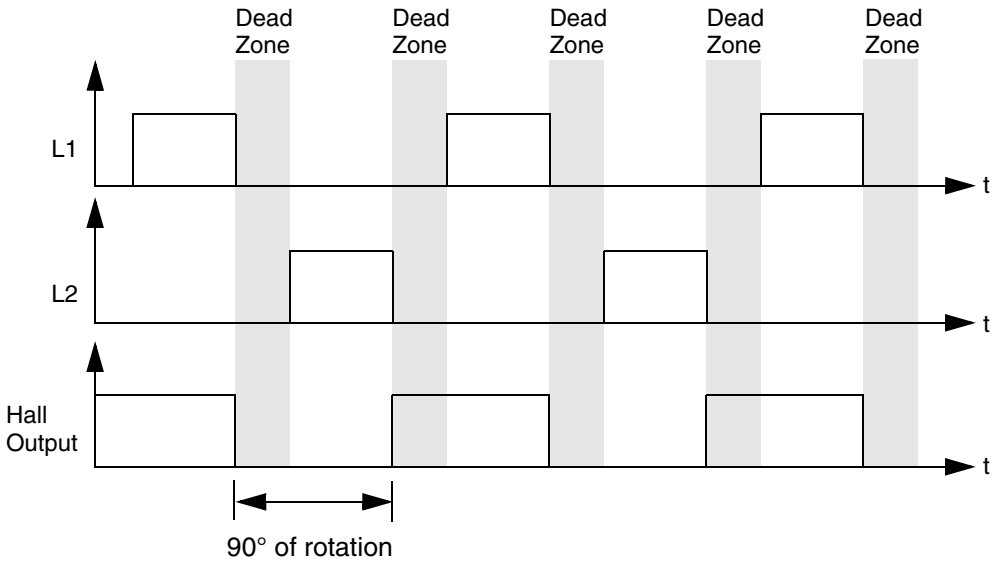


Figure 2-2. Bi-Phase BLDC Motor Commutation Waveform

2.4 Speed Control

Motor speed is normally defined as the mechanical revolution per one minute of time (rpm). In electrical terms, one commutation contributes to 90 degrees of a revolution. Thus, control the time taken per commutation can effectively control the overall speed. One commutation step includes a dead-time (where the coils are not energized) and the coils energization time. The whole commutation period could be considered as a pulse width modulation (PWM) output cycle. The PWM period defines the motor speed in this case. The coils energization time is, in fact, the PWM driving period which is defined by the time that the coils are energized until the Hall sensor is toggled. The Hall sensor output indicates the position of the rotor and defines the time to switch to the next commutation step.

In this design the motor speed or the PWM period is continuously monitored. It is a closed-loop control design. If the motor speed is faster (PWM period is shorter) than the target value, the dead-time duration is extended until the target PWM period is reached. Similarly, when the motor speed is slower than the target value, the dead-time duration is shortened.

The rotor starts off at the slowest speed. Shortening the dead-time causes the coils to energize earlier and the rotor is pushed/pulled to the next pole position sooner, causing motor speed to increase. Similarly, when the dead-time is extended the rotor hangs loose for a longer time before it is pushed/pulled to the next pole position. As a result the motor speed decreases. The target motor speed against temperature is predefined. It is updated periodically based on the information from the temperature sensor.

Dramatic changes in the dead-time value will cause the motor to stall. In this design a software loop in the MCU will control the dead-time variation. Even with the dramatic change in the temperature sensor reading, the software loop will only allow the dead-time to change to the new value gradually.

2.5 Motor Startup

In this DC fan application, it is desirable to only allow the motor to operate in an uni-direction, such that the airflow to the target system will always be in one direction. With the bi-phase motor design it is difficult to guarantee the direction of rotation. Commutation order or the coil energizing sequence happens to be the same for both directions of rotation. The rotor position or axis must initially be known in order to guarantee the direction of rotation. When the first commutation step is activated where the adjacent coil-pair to the initial axis is energized, the rotor starts to move. Since the adjacent coil-pairs are connected together and energized at the same time, there are equal pulling/pushing force induced on the rotor in both directions. There is chance for the rotor to startup in either direction. It is necessary to monitor the initial direction of rotation. If the direction is not correct, the motor must be locked back to the startup axis again and the commutation step repeated. The direction of rotation can be detected by the Hall sensor output. If the initial rotor axis is known, the output edge polarity, rising edge or falling edge, determines the direction of rotation.

In the modern bi-phase motor design the direction of rotation is normally defined by the manufacturer. The stator design is not symmetric such that the motor will have a high tendency to rotate in one direction than the other. However, the direction of rotation cannot be guaranteed without proper monitoring techniques in place.

2.6 Fault Detection

Motor fault is identified as the rotor not moving, which is normally the case when the rotor is jammed (may be cause by blocked airflow). During each commutation step, the Hall sensor output is monitored. If it is not toggled within a defined duration, commutation sequence is terminated, all coils are de-energized. In this design, when a motor fault occurs, a buzzer is activated as the alarm.

Chapter 3
Implementation

3.1 Block Diagram

The block diagram of the DC fan design is illustrated in Figure 3-1. A 12V low cost bi-phase BLDC motor is used in this application. The MCU performs alternate outputs to the two NPN transistors that drive the motor coils. Open drain output Hall sensor is required and positioned close the rotor. The device responds to magnetic field changes during the motor operation, digitizing output feedback of the rotor position to the MCU for close loop motor control and fault detection. Ambient temperature information is measured from an external temperature sensor. In the faulty situation, such as motor jam, the buzzer alarm is driven by the MCU through a pulse width modulated (PWM) output.

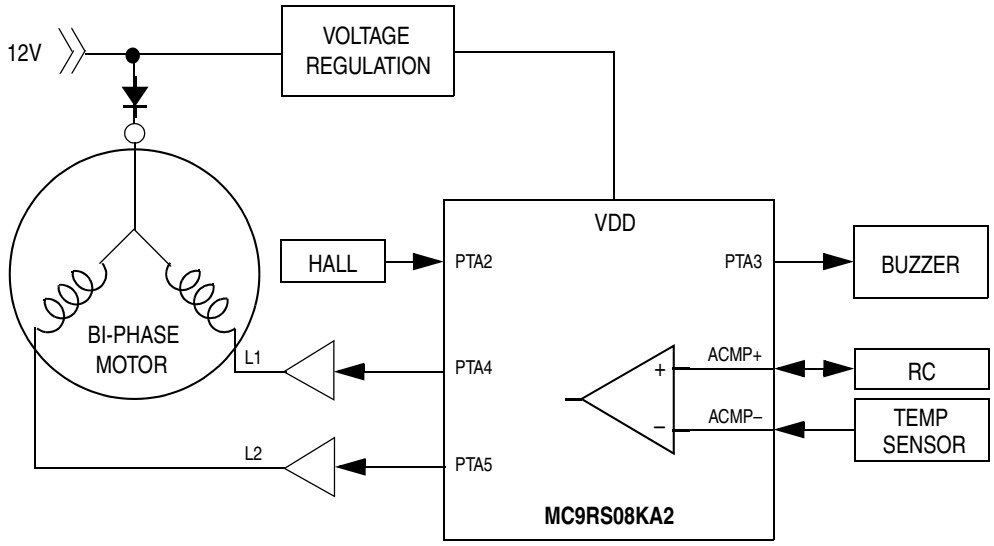


Figure 3-1. DC Fan Design Block Diagram

3.2 Hardware Resources

In this application, the low cost MC9RS08KA2 MCU is used. The device has a built-in 8-bit modulo timer which is used to control the timing for the PWM drive. Bus frequency is chosen to be 4MHz. The design target for the maximum motor speed is 4000 rpm, the timer must have enough resolution to measure the shortest PWM period that is less the 3.75ms per commutation step. Timer prescalar is selected as 256 and the timer resolution becomes 64μs.

Table 3-1. Hardware Configuration

Bus Frequency	4MHz
Timer Clock for motor speed monitoring	4MHz/256 = 16kHz
Timer Resolution	64μs

Hall sensor output is connected to the MCU's GPIO port, PTA2, which has a programmable edge trigger keyboard interrupt (KBI). The programmable edge trigger feature provides an effective way to monitor the Hall sensor signal. As mentioned in the previous section, the direction of rotation can be detected by the polarity of the Hall sensor output edge. Monitoring the signal edge is achieved by altering the KBI edge trigger polarity for each commutation step.

Ambient temperature reading is taken from a temperature sensor which is equivalent to a diode. Temperature variation alters the diode channel current as well as the effective channel resistance. The temperature sensor is combined with a 7.5kΩ resistor in a potential divider arrangement. The built-in analog comparator is used to compare the temperature sensor ladder voltage with an defined RC network to deduce the absolute temperature.

As described in the previous section, the motor speed is controlled by varying the absolute dead-time. This is updated every 128ms in the application. As in all RS08/S08 devices, the MC9RS08KA2 MCU has a programmable real time interrupt (RTI) feature. In this case, it is used to notify the MCU to refresh the target PWM period every 128ms.

3.3 Control Loop

Figure 3-2 shows the firmware control loop flow chart. The KBI or Hall sensor output is continuously monitored for trigger signals within a defined time. A motor fault condition occurs when there are no trigger signal, and the firmware goes into a forever loop. Commutation is stopped and the buzzer is alarmed.

The target PWM period based on the temperature sensor reading is updated every 128ms. And on each 180 degrees rotation of the rotor (two commutation steps) the actual PWM period is compared with the target PWM period. If they are different, the absolute dead-time will be altered, and the actual PWM period will gradually change towards the target PWM period.

On each commutation step, reading of the temperature sensor contributes a delay to the actual dead-time duration. This delay is deterministic such that the software control loop can easily deduce the actual speed of the motor. Hence, this delay can be considered as a part of the total dead-time delay for each commutation.

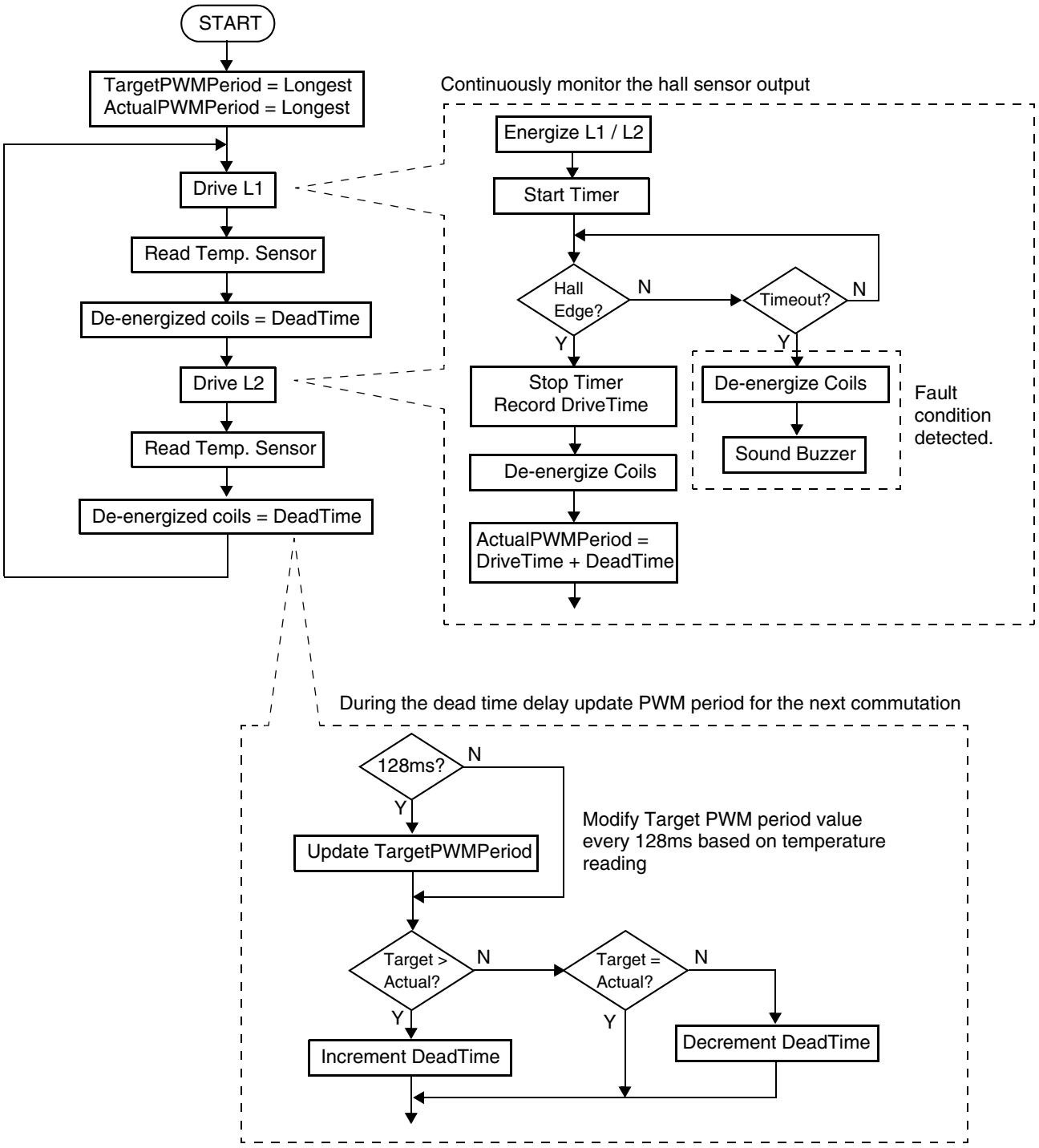


Figure 3-2. Firmware Control Loop

3.4 Temperature Sensor Measurement

The temperature sensor measurement is performed based on the methodology of an emulated ADC described in the application note, AN3266 “Getting Started with RS08”.

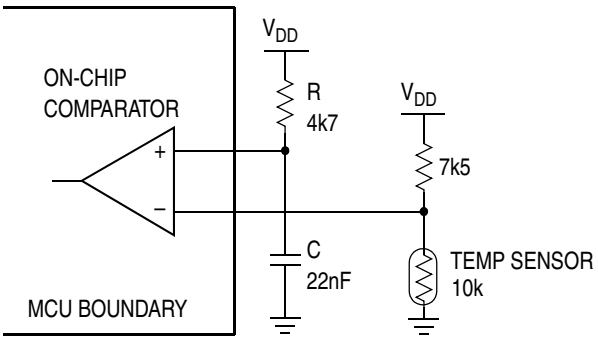


Figure 3-3. Emulated ADC Schematic

The schematic of the emulated ADC in this application is shown in Figure 3-3. The ADC input is the temperature sensor resistor ladder. When the comparator is not measuring, the capacitor, C, is fully discharged where the positive terminal of the comparator is pulling low. When the temperature sensor measurement is required, the comparator is then enabled and the terminal turns to analog input, voltage across C starts to ramp up. The 8-bit internal modulo timer is used to monitor the time taken for the RC to charge to a level that matches the voltage across the temperature sensor. The timer counter value is captured and used as the basis for the emulated ADC conversion.

With a 10kΩ temperature sensor and 7.5kΩ pullup resistor the ADC absolute dynamic range is from 0V to about 0.57 × V_{DD}, i.e. about 2.85V. Timer clock is chosen to be eight times slower than the bus clock, the timer resolution becomes 2μs. The RC charging profile follows EQ 3-1. Given the RC constant is 4K7Ω × 22nF the timer counter value against the temperature sensor reading with 5V V_{DD} is shown in Table 3-2.

$$V = V_{DD}\left(1 - e^{-\frac{t}{RC}}\right) \quad \textbf{(EQ 3-1)}$$

Table 3-2. RC Charging Profile Against Timer Count

Time (μs)	Voltage across the Temperature Sensor (V)	ADC Readout (Timer Count)
0	0	0
2	0.10	1
4	0.19	2
6	0.28	3
and so on...		
86	2.82	43
88	2.87	44
90	2.91	45
and so on...		
126	3.52	63

Table 3-2 shows the entire dynamic range of the temperature sensor voltage can be covered by about 44 timer counts. For convenience, the timer overflow period is set to 63, which is identical to the size of the paging window ($00C0 to $00FF) in the MC9RS08KA2. The timer value captured can be used directly as an index to the paging window for the target PWM period value lookup.

The code below shows how the timer value is captured using RS08 instructions.

```
ReadSensor:
       mov    #(MTIM_BUS_CLK|MTIM_DIV_8), MTIMCLK; Change Timer resolution
       mov    #63, MTIMMOD                 ; OF period
       mov    #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC; Reset and Start Timer

       mov    #(mACMPSC_ACME|mACMPSC_ACIE|ACMP_OUTPUT_RAISING), ACMPSC
                                           ; Enable ACMP, start RC rise
       bset   ACMPSC_ACF, ACMPSC           ; Clear ACMP Flag
       wait
       brclr  ACMPSC_ACF, ACMPSC, NoReading
       mov    MTIMCNT, SensorReading       ; Capture timer count
       bset   ACMPSC_ACF, ACMPSC           ; Clear ACMP Flag
       clr    ACMPSC                       ; disable ACMP
       wait                                ; delay to OF and make the
                                           ; read process deterministic
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC; mask interrupt and clear
                                           ; flag
       mov    #(MTIM_BUS_CLK|MTIM_DIV_256), MTIMCLK; Reset Timer resolution
       rts
NoReading:
       mov    #$00, SensorReading          ; Smallest Number
       clr    ACMPSC                       ; disable ACMP
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; mask interrupt and clear
                                           ; flag
       mov    #(MTIM_BUS_CLK|MTIM_DIV_256), MTIMCLK; Reset Timer resolution
       rts
```

As described in the previous section the overall dead-time duration should be deterministic, the double WAIT statements in the subroutine can ensure the execution time to be mostly constant. When the MCU is woken up from the first WAIT (which is normally triggered by the comparator), the timer counter value is captured and the MCU is then returned to WAIT mode until the timer is overflowed. The subroutine execution time would be equivalent to the timer overflow period (~128μs) plus some software overhead.

3.4.1 Temperature Conversion

In general, the channel resistance of the temperature sensor reduces as the temperature increases. The corresponding channel resistance against temperature can usually be retrieved from the sensor data sheet. For this application the operating temperature range is defined from 25°C to 100°C. When the ambient temperature is 100°C or above the motor is at maximum speed. The speed drops as the temperature decreases in 5°C steps. Given the sensor channel resistance values the voltage across the sensor can be calculated. The corresponding motor speed for a specific temperature range are also defined and shown in Table 3-3.

EQ 3-2 shows how the target PWM period value is calculated. The target value is compared with the measured PWM period every 180 degrees of rotation. The ADC readout delay is considered as constant, therefore, it is omitted from the motor speed measurement and should be deducted from the target period calculation, too.

$$\mathrm{TargetPWMPeriod} = \frac{\frac{60/\mathrm{RPM}}{4} - \mathrm{ADCDelay}}{\mathrm{TimerResolution}} \qquad \textbf{(EQ 3-2)}$$

The timer resolution used in the application is 64μs, the ADC readout time contributes a constant delay to the overall PWM period, which is ~128μs in this application. The target PWM period used for motor speed control is shown in Table 3-3. The table is stored in the upper memory (FLASH). In RS08 architecture upper memory access is done through the paging window (address $00C0 to $00FF) where the PAGESEL register is defining the page to be accessed. Simple table lookup method which uses the captured timer value from the temperature sensor readout as an index in the paging window for the target PWM period conversion.

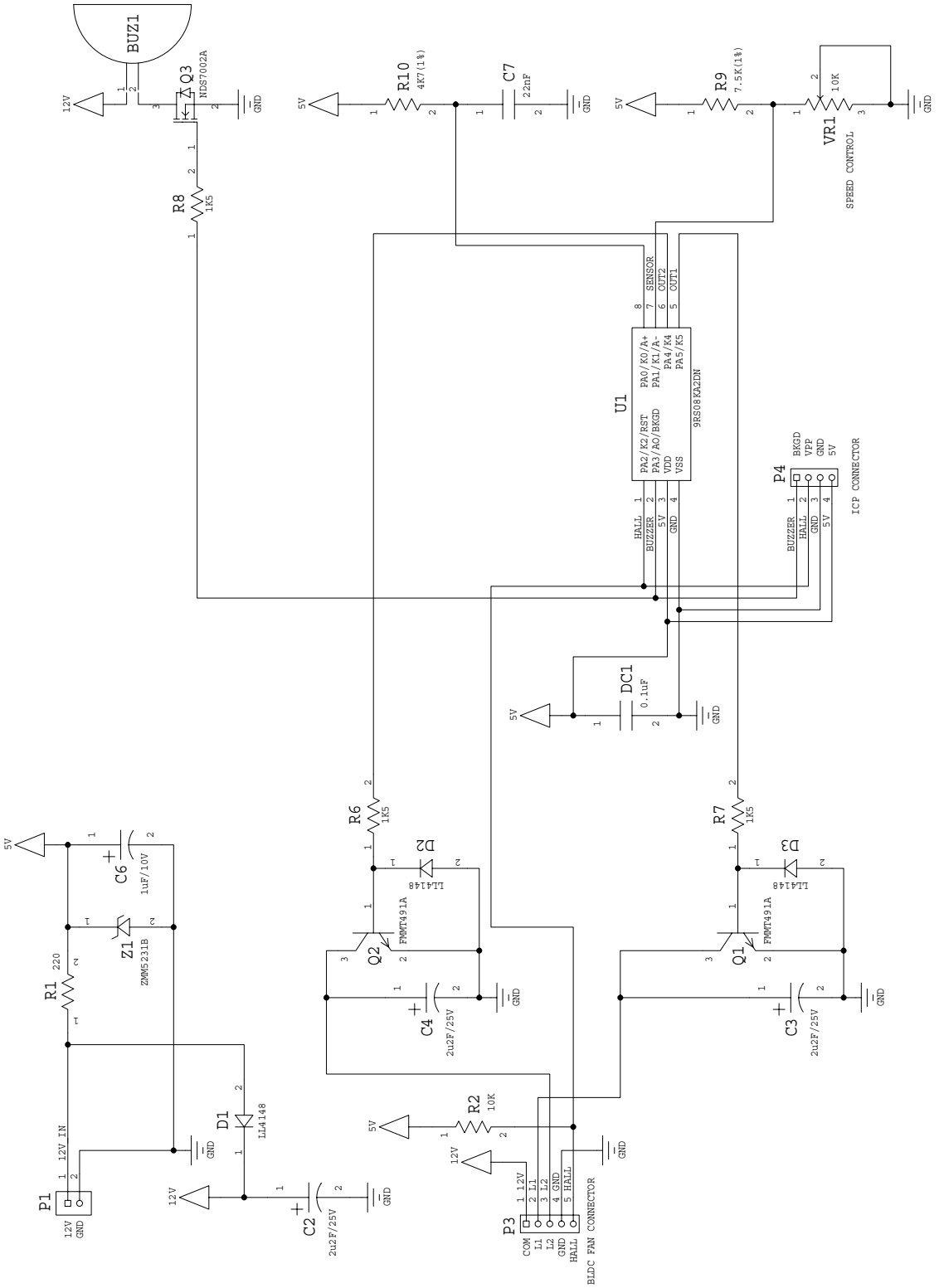
For software implementation, the target motor speed must be deduced in terms of timer counts, where it is used as the target PWM period per commutation. By using Table 3-2 and Table 3-3, a look-up table can be constructed where the ADC readout value is used as an index to retrieve the target PWM period for a specific temperature range.

Table 3-3. Temperature Conversion Table

Temperature (°C)	Channel Resistance (kΩ) (from sensor data sheet)	Voltage across Sensor (V)	Predefined Motor Speed (rpm)	Target PWM Period (Timer Counts[1])
25 or below	10	2.86	1000	232
30 – 34	8.082	2.59	1200	193
35 – 39	6.577	2.34	1400	165
40 – 44	5.387	2.09	1600	144
45 – 49	4.441	1.86	1800	128
50 – 54	3.683	1.65	2000	115
55 – 59	3.024	1.44	2200	105
60 – 64	2.53	1.26	2400	96
65 – 69	2.128	1.11	2600	88
70 – 74	1.799	0.97	2800	82
75 – 79	1.528	0.85	3000	76
80 – 84	1.304	0.74	3200	71
85 – 89	1.118	0.65	3400	67
90 – 94	0.962	0.57	3600	63
95 – 99	0.831	0.50	3800	60
100 or above	0.698	0.43	4000	57

NOTES:
1. The resolution of a timer count is 64μs.

Appendix A. Schematic



Variable Speed DC Fan Control using the MC9RS08KA2, Rev. 0

Appendix B.
Program Listing

```
;*************************************************************
;
;  (c) copyright Freescale Semiconductor. 2006
;  ALL RIGHTS RESERVED
;
;*************************************************************
;*************************************************************
;* DC Fan Coding for 9RS08KA2
;*
;* Author:    Vincent Ko
;* Date:      Jan 2006
;*
;* PTA0/KBI0/ACMP+             RC input
;* PTA1/KBI1/ACMP-             Temp sensor input
;* PTA2/KBI2/TCLK/RESETb/VPP   Hall input
;* PTA3/ACMPO/BKGD/MS          Buzzer
;* PTA4/KBI4                   PWM+
;* PTA5/KBI5                   PWM-
;*
;*************************************************************
; include derivative specific macros
       XDEF    Entry

       include "MC9RS08KA2.inc"

;===============================================================================
; ICS Definition
;===============================================================================
ICS_DIV_1          equ$00
ICS_DIV_2          equ$40
ICS_DIV_4          equ$80
ICS_DIV_8          equ$c0

;===============================================================================
; MTIM Definition
;===============================================================================
MTIM_DIV_1     equ     $00
MTIM_DIV_2     equ     $01
MTIM_DIV_4     equ     $02
MTIM_DIV_8     equ     $03
MTIM_DIV_16    equ     $04
MTIM_DIV_32    equ     $05
MTIM_DIV_64    equ     $06
MTIM_DIV_128   equ     $07
MTIM_DIV_256   equ     $08

MTIM_BUS_CLK   equ     $00
MTIM_XCLK      equ     $10
```

```
MTIM_TCLK_FALLING      equ   $20
MTIM_TCLK_RISING       equ   $30

;===============================================================================
; ACMP Definition
;===============================================================================
ACMP_OUTPUT_FALLING    equ   $00
ACMP_OUTPUT_RAISING    equ   $01
ACMP_OUTPUT_BOTH       equ   $03

;===============================================================================
; RTI Definition
;===============================================================================
RTI_DISABLE    equ      $00
RTI_8MS        equ      $01
RTI_32MS       equ      $02
RTI_64MS       equ      $03
RTI_128MS      equ      $04
RTI_256MS      equ      $05
RTI_512MS      equ      $06
RTI_1024MS     equ      $07

;===============================================================================
; Application Definition
;===============================================================================
RC             equ     PTAD_PTAD0
mRC            equ     mPTAD_PTAD0
TEMPSEN        equ     PTAD_PTAD1
mTEMPSEN       equ     mPTAD_PTAD1
HALL           equ     PTAD_PTAD2
mHALL          equ     mPTAD_PTAD2
BUZZER         equ     PTAD_PTAD3
mBUZZER        equ     mPTAD_PTAD3
PWM2           equ     PTAD_PTAD4
mPWM2          equ     mPTAD_PTAD4
PWM1           equ     PTAD_PTAD5
mPWM1          equ     mPTAD_PTAD5

MinDeadTime    equ     2
MaxDeadTime    equ     150

TableStart:    equ     $00003E00
;===============================================================================
; Application Macro
;===============================================================================
StartTimer: macro
       mov    DeadTime, MTIMMOD             ; OF period
       mov    #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC; Reset and Start Timer
       endm

       org    TINY_RAMStart
; variable/data section
DeadTime       ds.b 1
```

```
TargetPeriod  ds.b 1
ActualPeriod  ds.b 1
DriveTime     ds.b 1
SensorReading ds.b 1
MotorRunning  ds.b 1

       org    RAMStart
; variable/data section

       org    ROMStart
; code section
main:
Entry:
;------------------------------------------------------
; Config ICS
; Device is pre-trim to 16MHz ICLK frequency
; TRIM value are stored in $3FFA:$3FFB
;------------------------------------------------------
       mov    #HIGH_6_13(NV_ICSTRM), PAGESEL
       mov    MAP_ADDR_6(NV_FTRIM), ICSSC   ; $3FFB
       mov    MAP_ADDR_6(NV_ICSTRM), ICSTRM ; $3FFA
       mov    #ICS_DIV_2, ICSC2             ; Use 4MHz

;------------------------------------------------------
;Config System
;------------------------------------------------------
       mov    #HIGH_6_13(SOPT), PAGESEL     ; Init Page register
       mov    #(mSOPT_COPT|mSOPT_STOPE), MAP_ADDR_6(SOPT)
                                            ; BKGD disable, COP disabled
       mov    #(mSPMSC1_LVDE|mSPMSC1_LVDRE), MAP_ADDR_6(SPMSC1); LVI enable
       mov    #(RTI_128MS), MAP_ADDR_6(SRTISC) ; 128ms RTI

;------------------------------------------------------
; Init RAM
;------------------------------------------------------
       mov    #MaxDeadTime, DeadTime
       mov    #232, TargetPeriod            ; 1000 rpm
       mov    #232, ActualPeriod            ; 1000 rpm
       clr    SensorReading
       clr    MotorRunning

;------------------------------------------------------
; Config GPIO
; RC - init L
; Buzzer - init L
; PWMn/PWMp - init L
;------------------------------------------------------
       clr    PTAD                          ; Initial low
       mov    #(mRC|mPWM1|mPWM2), PTADD     ; Set Output pins
;------------------------------------------------------
; Config KBI
;------------------------------------------------------
       lda    #mHALL
```

```
       sta    KBIES         ;HALL rising Edge Trigger
       sta    KBIPE         ;KBI Enable

;--------------------------------------------------------
;Config MTIM
;
;Timer prescalar=256 -> Timer clk = 16kHz
;Bus = 4MHz
;Max OF period = 16.384ms
;Timer resolution = 64us
;--------------------------------------------------------
       mov    #(MTIM_BUS_CLK|MTIM_DIV_256), MTIMCLK
       mov    #255, MTIMMOD

;--------------------------------------------------------
;Motor Start Sequence
;--------------------------------------------------------
ResetPosition:
       mov    #mPWM1, PTAD                ; Lock FAN in reset position
       lda    #30                         ;
Dly1   bsr    Delay                       ; for Delay 0.5s
       dbnza  Dly1                        ;
       clr    PTAD                        ; de-energize coils
       bsr    Delay

; Drive L2
       ldx    #mPWM2                      ; Select L2 Coils
       bsr    SetPWM                      ; Drive coil
       bsr    Delay                       ; De-energize coils
       inc    MotorRunning                ; otherwise Update Software flag

;--------------------------------------------------------
;Fan Control Loop
;--------------------------------------------------------
FanControlLoop:

;1) Drive L1 coil
       clr    KBIES                       ; HALL falling edge trigger
       ldx    #mPWM1                      ; Select L1 Coil
       bsr    SetPWM                      ; Drive coil

;2) Read Temp Sensor
       jsr    ReadSensor                  ; Read Sensor value

;3) Dead time control
       StartTimer                         ; Wait dead time period
       wait
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC; mask interrupt and clear flag

;4) Drive L2 coil
       bset   HALL, KBIES                 ; HALL rising edge trigger
       ldx    #mPWM2                      ; Select L2 Coil
       bsr    SetPWM                      ; Drive coil
```

```
;5) Read Temp Sensor Again
       bsr    ReadSensor                    ; Read Sensor value

;6) Dead time control
       StartTimer

;7) During the dead time, update dead time period every 128ms
       brclr  SRTISC_RTIF, MAP_ADDR_6(SRTISC), UpdateLater; Update PWM duty cycle
       jsr    TableLookup
UpdateLater:
       lda    ActualPeriod
       sub    TargetPeriod  ; Actual-Target
       blo    IncPeriod
       beq    WaitAgain     ; if same, Fan speed reach target then exit

DecPeriod:                  ; if bigger, decrement DeadTime
       lda    DeadTime
       cmp    #MinDeadTime
       blo    WaitAgain
       dec    DeadTime
       bra    WaitAgain

IncPeriod:                  ; if smaller, increment DeadTime
       lda    DeadTime
       cmp    #MaxDeadTime
       bhs    WaitAgain
       inc    DeadTime
       bra    WaitAgain

WaitAgain:
;8) Bump COP
       sta    MAP_ADDR_6(SRS)               ; Bump COP
       wait
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC; mask interrupt and clear flag

;9) Repeat the control cycle
       bra    FanControlLoop

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Delay 16ms
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Delay:
       mov    #255, MTIMMOD                 ; OF period
       mov    #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC; Reset and Start Timer
       wait
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC; mask interrupt and clear flag
       sta    MAP_ADDR_6(SRS)               ; Bump COP
       rts

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Drive coil
;
```

```
; X indicate the coil to be driven
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SetPWM:
       mov    #255, MTIMMOD                  ; OF period
       mov    #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC; Reset and Start Timer

       lda    #20
       mov    #(mKBISC_KBIE), KBISC          ; Enable Interrupt & Edge only
       bset   KBISC_KBACK, KBISC             ; Clear Flag
       stx    PTAD                           ; Drive coil
TimingLoop:
       bclr   MTIMSC_TOF, MTIMSC             ; Clear TOF
       wait
       brset  KBISC_KBF, KBISC, HallFound    ; HALL sensor edge found
       dbnza  TimingLoop
       jmp    MotorHang      ; If no HALL output, Stop the driving
HallFound:
       mov    MTIMCNT, DriveTime
       cbeqa  #20, StableDrive
       mov    #MaxDeadTime, DriveTime
StableDrive:
       lda    DeadTime
       add    DriveTime
       sta    ActualPeriod
       clr    PTAD                           ; Disconnect coil
       mov    #(mKBISC_KBACK), KBISC         ; Clear Flag and mask interrupt
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC; mask interrupt and clear flag
       rts

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Read Temperature Sensor Value
; Timer prescalar=8 -> Timer clk~250kHz
; Bus = 2MHz
; Max OF period = 1.02ms
; Timer resolution = 4us
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ReadSensor:
       mov    #(MTIM_BUS_CLK|MTIM_DIV_8), MTIMCLK; Change Timer resolution
       mov    #63, MTIMMOD                   ; OF period
       mov    #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC; Reset and Start Timer
       mov    #(mACMPSC_ACME|mACMPSC_ACIE|ACMP_OUTPUT_RAISING), ACMPSC
                             ; Enable ACMP, start RC rise
       bset   ACMPSC_ACF, ACMPSC             ; Clear ACMP Flag
       wait                  ; delay to OF and make the read process deterministic
       brclr  ACMPSC_ACF, ACMPSC, NoReading
       mov    MTIMCNT, SensorReading
       bset   ACMPSC_ACF, ACMPSC             ; Clear ACMP Flag
       clr    ACMPSC                         ; disable ACMP
       wait
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC; mask interrupt and clear flag
       mov    #(MTIM_BUS_CLK|MTIM_DIV_256), MTIMCLK; Reset Timer resolution
       rts
```

```
NoReading:
       mov    #$00, SensorReading           ; Smallest Number
       clr    ACMPSC                        ; disable ACMP
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; mask interrupt and clear flag
       mov    #(MTIM_BUS_CLK|MTIM_DIV_256), MTIMCLK; Reset Timer resolution
       rts

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; 6-bit Table Lookup
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TableLookup:
       bset   SRTISC_RTIACK, MAP_ADDR_6(SRTISC);5
       mov    #HIGH_6_13(TableStart), PAGESEL;5 Calculate the PAGE
       lda    SensorReading                 ;3
       add    #$c0                          ;2 Reference to paging window
       tax                                  ;2
       lda    ,x                            ;3
       sta    TargetPeriod                  ;2
       mov    #HIGH_6_13(SOPT), PAGESEL     ;5
       rts                                  ;3

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Error Handling
; Stop the motor
; Sound the buzzer (about 520Hz)
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MotorHang:
       clr    PTAD                          ; clear PWMp and PWMn
       lda    MotorRunning                  ; Check software flag
       bne    SoundBuzzer                   ; =1, Motor is running
       jmp    ResetPosition

SoundBuzzer:
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC; mask interrupt and clear flag
       clr    KBISC                         ; mask KBI

       lda    #255
       sta    MAP_ADDR_6(SRS)               ; Bump COP
Beep:                                       ; a 20% duty cycle loop
       bset   BUZZER, PTAD                  ; Drive buzzer
       mov    #6, MTIMMOD
       mov    #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC; Reset and Start Timer
       wait
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC; mask interrupt and clear flag
       sta    MAP_ADDR_6(SRS)               ; Bump COP

       bclr   BUZZER, PTAD                  ; Clear buzzer
       mov    #24, MTIMMOD
       mov    #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC; Reset and Start Timer
       wait
       mov    #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC; mask interrupt and clear flag
       sta    MAP_ADDR_6(SRS)                                          ; Bump COP
       dbnza  Beep
```

```
       lda    #255
Quiet:
       bclr   BUZZER, PTAD                  ; Clear buzzer
       mov    #30, MTIMMOD
       mov    #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC; Reset and Start Timer
       wait
       mov   #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC; mask interrupt and clear flag
       sta    MAP_ADDR_6(SRS)               ; Bump COP
       dbnza  Quiet

       bra    SoundBuzzer

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Lookup Table
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
org    TableStart

       dc.b    57, 57, 57, 57, 57, 60, 63, 67, 71, 76, 82, 82, 88, 88, 96, 96
       dc.b   105,105,115,115,115,128,128,128,128,144,144,144,144,165,165,165
       dc.b   165,193,193,193,193,193,232,232,232,232,232,232,232,232,232,232
       dc.b   232,232,232,232,232,232,232,232,232,232,232,232,232,232,232,232

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Reset Vector
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       org    $3ffc
Security:
       dc.b   $FF
       jmp    main
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

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

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

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

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

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

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

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

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

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

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

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

DRM079
Rev. 0, 5/2006

freescale™
semiconductor

Freescale Semiconductor
Application Note

Document Number: AN3409
Rev. 0, 01/2007

Basic Refrigerator Control Using the MC9RS08KA2

by: Jose Ruiz
RTAC Americas

1 Introduction

Some refrigerators still have a basic electromechanical circuit that controls the temperature. This application shows how to implement a low-cost, basic temperature control for refrigerators using the MC9RS08KA2. This method can be implemented to control the temperature of any device using a thermoresistor, a potentiometer, resistors and a capacitor.

2 Requirements

- MC9RS08KA2 microcontroller (MCU)
- One potentiometer
- One thermoresistor
- One ceramic capacitor
- Two ¼ watt resistors
- CodeWarrior™ 5.1 development tool

Contents

© Freescale Semiconductor, Inc., 2007. All rights reserved.

freescale™ semiconductor

3 Implementation

The temperature control is implemented with a single potentiometer and a capacitor connected to one MC9RS08KA2 MCU I/O pin. The temperature sensor is a basic voltage divider formed by a resistor and a thermistor. The output is an I/O pin connected to a relay that switches the supply of the refrigerator.

The flow of the program consists of reading the control wheel value followed by reading the sensor voltage and, finally, switching the output ON or OFF according to the control and sensor values.

3.1 Control Value

The refrigerator temperature control is a basic RC network connected to an I/O pin. By measuring the charging time of the RC network, we can determine the potentiometer resistance, and therefore, the value you entered. The charge curve of the RC network is used to determine the time the curve takes to go from 0 V to the input-high voltage (V_{IH}). This method is used because the MC9RS08KA2 MCU does not have an integrated analog-to-digital converter (ADC).

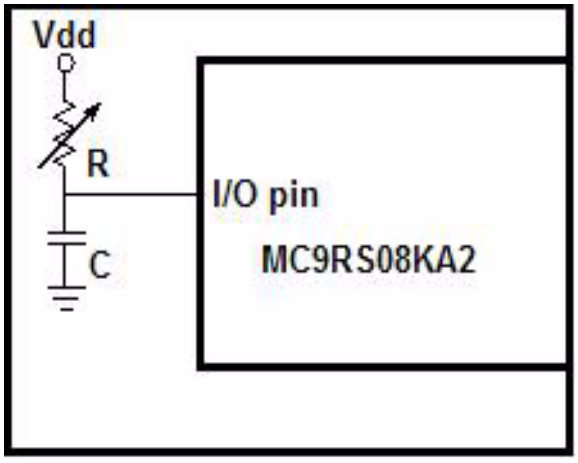


Figure 1. Temperature Control Implementation

The first step is configuring the control pin as output. Set the pin value to 0 to discharge the capacitor. After the capacitor is fully discharged, change the control pin direction to an input. The capacitor starts charging to V_{DD}.

When the voltage of the capacitor gets to V_{IH}, the pin state changes from 0 to 1.

A variable resistor (potentiometer) is used to modify the time the capacitor takes to reach V_{IH}. Adjusting its resistance varies that time.

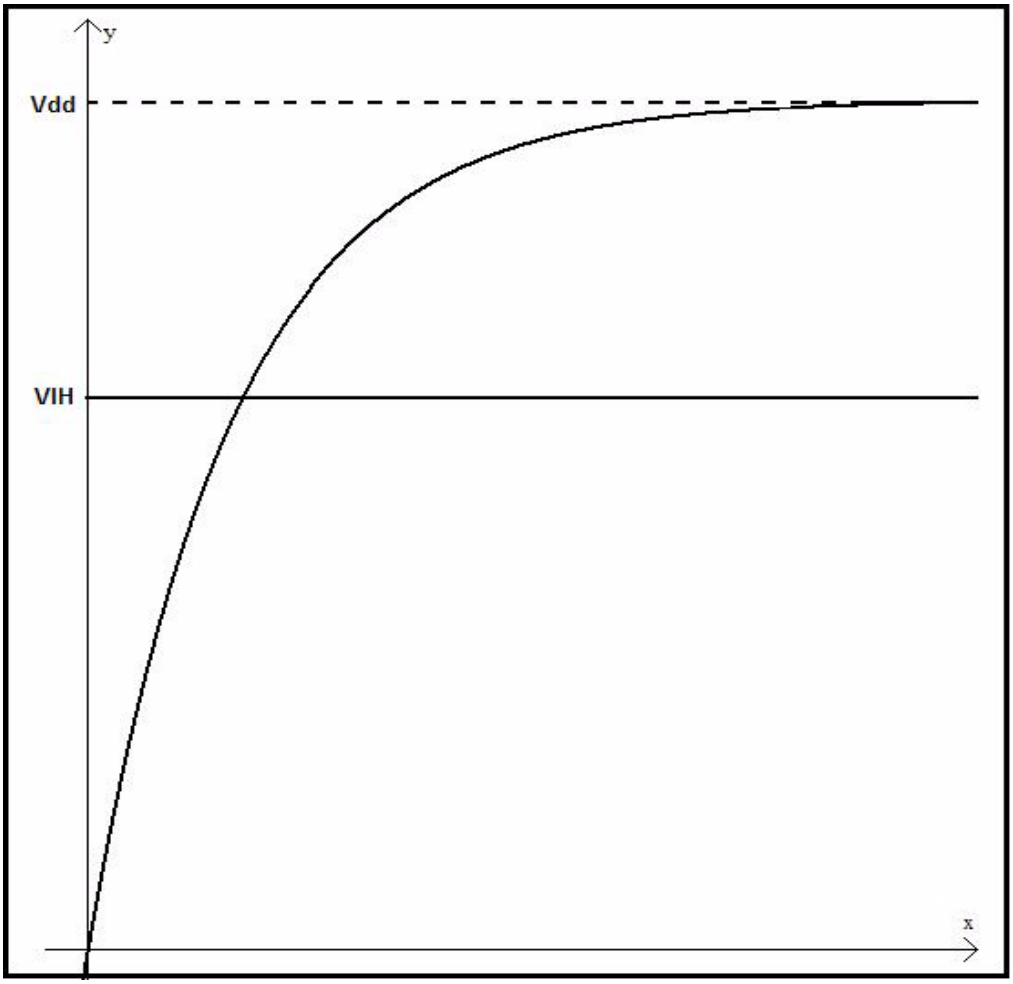


Figure 2. Charge Capacitor Waveform

The capacitor voltage is given by the following equation:

$$Vc = Vdd\left(1 - e^{-\frac{t}{rc}}\right)$$

Solving for time

$$t = -rc\ln\left(1 - \frac{Vc}{Vdd}\right)$$

- Vc — Voltage of the capacitor
- V_{DD} — Supply voltage of RC network
- t — Time (seconds)
- r — Resistance
- c – Capacitor

A 10 kΩ potentiometer and 33 nF capacitor were used in this application note.

From the MC9RS08KA2 datasheet, we know that when $V_{DD} > 2.3$ V, the V_{IH} for the inputs is 0.70 x V_{DD}.

If the MC9RS08 MCU is supplied with 3.3 V then:

$V_{IH} = 0.70 \times V_{DD} = (0.70 \times 3.3.) = 2.31$ V

Table 1 shows the difference in time using the above with different resistance commercial values.

Table 1. Time Result According Resistance Values

V_{DD}	V_{IH}	R	C	time
3.3	2.31	1k	33nF	3.973E-05
3.3	2.31	3k	33nF	0.0001192
3.3	2.31	5k	33nF	0.0001987
3.3	2.31	7k	33nF	0.0002781
3.3	2.31	10k	33nF	0.0003973

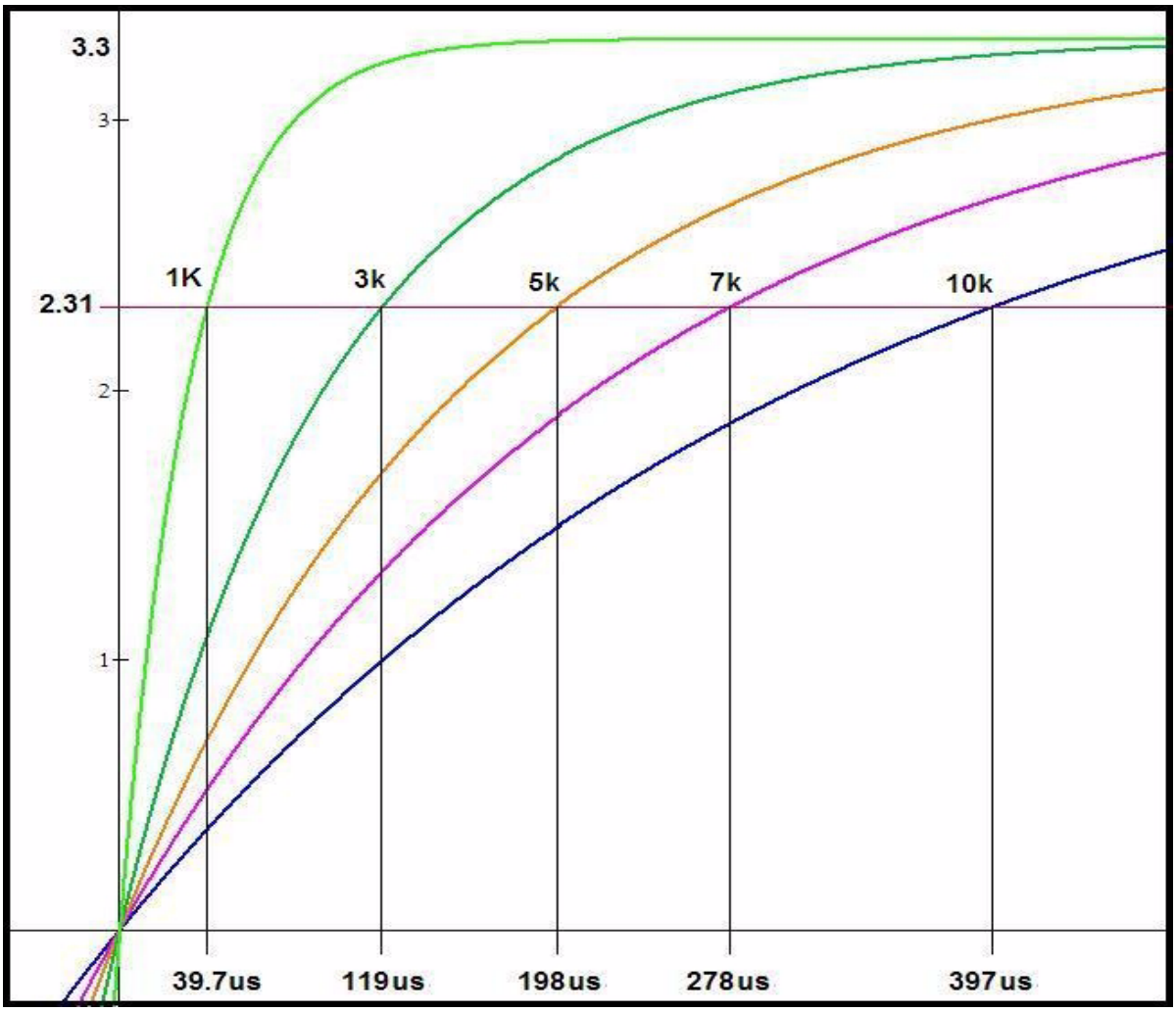


Figure 3. Charge Curve with Different Resistor

NOTE

The resistor value must not reach 0, or a short circuit can occur.

3.1.1 Code Implementation

The first step is to configure the control pin as output, and setting a low level on it, (0). Then wait for the RC network to discharge completely;

```
Pin_Measure:
        bset control,PTADD        ; Set control pin as Output
        bclr control,PTAD         ; Discharge RC network
        clr ControlValue
        lda  #$FE
Discharge2:
        dbnza Discharge2
```

The following step is to configure the control pin as input and increment a counter while pin state is 0:

```
        bclr control,PTADD                ; Set Control pin as Input
measure_pin:
        inc ControlValue
        brclr control,PTAD,measure_pin; Inc value while pin is in low state
        rts
```

The `ControlValue` variable represents the time taken for the capacitor to reach V_{IH}.

After the pin reaches the high level, we know the approximate position of the potentiometer entered by the user.

3.2 Temperature Sensor

A basic voltage divider with one resistor and one thermoresistor is used to implement the temperature sensor. The thermoresistor resistance depends on the temperature. For each temperature, we have a different voltage in the divider. This value is effectively measured with the ADC implemented by software that uses one resistor, one capacitor, and the analog comparator included in the MC9RS08KA2 MCU.

The voltage divider is composed of the thermoresistor NCP18WB333J03RB and a 82 ohms resistor. It is better to have a big variation in the output voltage of the sensor with a little variation in the temperature.

The supply voltage of the RC network in this application note is 3.3 V and the output voltage of the sensor can be calculated with the next equation.

$$Vout = Vdd\left(\frac{NTC}{NTC + R}\right) = 3.3\left(\frac{NTC}{NTC + 82}\right)$$

According to the thermoresistor specifications, the resistor range is between 89.61 Ω to 116.16 Ω in a range of 4 °C to –0.5 °C (Section Appendix A, “NCP18WB333J03RB Thermistor Range Table”). With those values the following data is calculated:

Table 2. Sensor Output Voltage

Temperature	NTC Value	Resistor	V_{DD}	Sensor Output
– 1	119.11	82	3.3	1.9544677
– 0.5	116.16	82	3.3	1.9344368
0	113.21	82	3.3	1.9138005
0.5	110.26	82	3.3	1.8925309
1	107.31	82	3.3	1.8705985
1.5	104.36	82	3.3	1.8479717
2	101.41	82	3.3	1.824617
2.5	98.46	82	3.3	1.8004987
3	95.51	82	3.3	1.7755788
3.5	92.56	82	3.3	1.7498167
4	89.61	82	3.3	1.7231688

Instead of having an ADC module, the MC9RS08KA2 MCU has a basic ADC implemented by software using the analog comparator module. This software ADC is basically composed by a RC network and the analog voltage to be measured. The software measures the time taken by the RC network to reach the sensor input voltage. This ADC by software is fully detailed in the R*S08 Quick Reference Guide* (RS08QRUG). Download the document at http://www.freescale.com

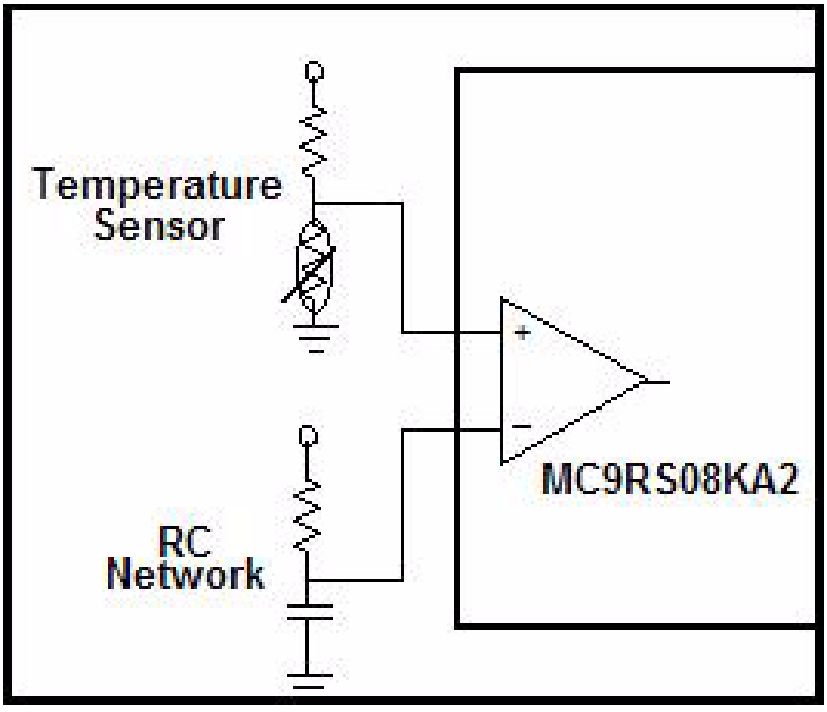


Figure 4. Sensor Value Input (ADC by Software)

The formula to calculate the time taken for the capacitor to charge is the same as the temperature control formula :

$$Vc = Vdd\left(1 - e^{-\frac{t}{rc}}\right)$$

Solving for time

$$t = -rc\ln\left(1 - \frac{Vc}{Vdd}\right)$$

But, for the ADC by software the RC network is fixed. In this case, the resistor value is 10 kΩ. The capacitor is 0.1 μF.

If the sensor values and the capacitor charging curve are graphed together the result is the time the RC network takes to reach the sensor output voltage.

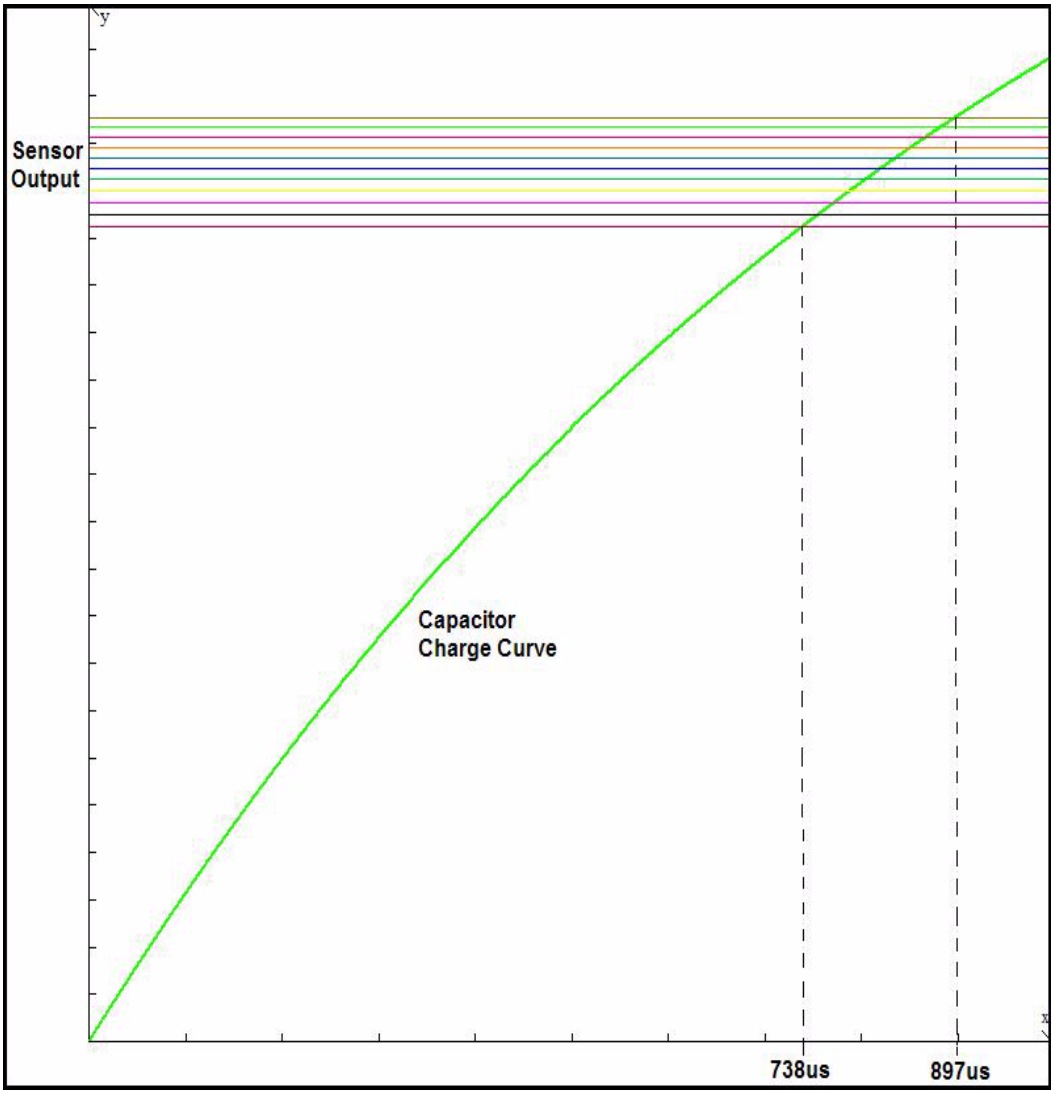


Figure 5. Capacitor Charge Versus Sensor Output Voltage

Based on the bus speed (8 MHz for this application), it is effective to build a table with the timer value according the sensor voltage.

To calculate the timer counts of each sensor voltage the next formula must be applied:

$$TimerCounts = VIH\ \ time \left(\frac{BusClock}{prescaler} \right)$$

Table 3. Temperature, Sensor Output, and Microcontroller Counts

Temperature	V sensor	V_{IH} Time	Timer counts (Bus/32)	Timer counts (Bus/32)
-0.5°C	1.93444	0.0008824	220.5889	110.2944
0°C	1.9138	0.0008674	216.8392	108.4196
0.5°C	1.89253	0.0008521	213.0323	106.5162
1°C	1.8706	0.0008367	209.1667	104.5833
1.5°C	1.84797	0.000821	205.2403	102.6201
2°C	1.82462	0.000805	201.2512	100.6256
2.5°C	1.8005	0.0007888	197.1975	98.59874
3°C	1.77558	0.0007723	193.0769	96.53846
3.5°C	1.74982	0.0007555	188.8873	94.44366
4°C	1.72317	0.0007385	184.6263	92.31315

3.2.1 Code Implementation:

```
ADC_Single_conversion:

        ; Discharge Capacitor

bset 1,PTADD
        bclr 1,PTAD
        lda  #$FE
waste:
        dbnza waste
        mov #ACMP_ENABLE,ACMPSC          ; ACMP Enabled
        mov #MTIM_ENABLE,MTIMSC          ; Timer Counter Enabled
        wait                             ; Wait for Analog Comparator Interrupt
        bset 4,MTIMSC                    ; Stop MTIM
        lda MTIMCNT                      ; read counter timer value
        sta ADCValue                     ;store counter value
        mov #MTIM_STOP_RESET,MTIMSC      ;Stop and reset counter

        mov #HIGH_6_13(SIP1), PAGESEL
        brset 3, MAP_ADDR_6(SIP1),Conv_OK ; branch if ACMP interrupt arrives
        bra ADC_Single_conversion

Conv_OK:
        mov #ACMP_DISABLED, ACMPSC       ; ACMP Disabled, Clear Interrupt flag
        rts
```

3.3 Temperature Control Application

The refrigerator's temperature control has four positions, the range of each one is:

- Position 4: 0 °C – 1 °C
- Position 3: 1 °C – 2 °C
- Position 2: 2 °C – 3 °C
- Position 1: 3 °C – 4 °C

The control switches on the relay when the temperature is over range. It switches it off when the temperature reaches the window value.

Because of temperature inertia, the window temperature is 1.5 °C. Figure 6 shows the window and the values from it.

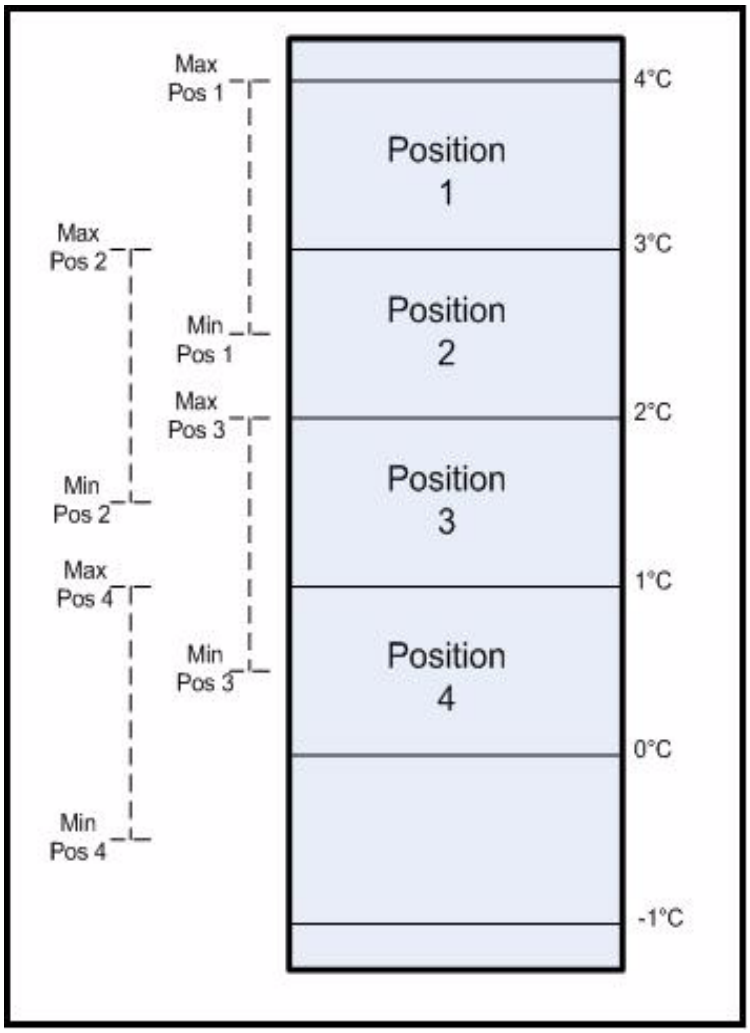


Figure 6. Temperature Control Range

For example, when the temperature position is 1, if the temperature is higher than 4 °C, the relay is closed, and the refrigerator compressor is on. Next, when the temperature reaches 2.5 °C, the application opens the relay and the compressor stops.

This guarantees that the temperature is stable for long periods of time between the ranges and, no matter what; the temperature is never more than 4 °C.

Each temperature limit can be easily changed in the definition part of the main code.

```
; Variable definitions
; Prescaler /64
TEMP1_ON          SET 92
TEMP1_OFF         SET 99
TEMP2_ON          SET 97
TEMP2_OFF         SET 103
TEMP3_ON          SET 100
TEMP3_OFF         SET 107
TEMP4_ON          SET 105
TEMP4_OFF         SET 111
```

The `definition_ON` is the value that closes the relay, and `definition_OFF` opens the relay. And the resolution of these values can be adjusted with the timer prescaler.

3.3.1 Code Implementation

```
;*************************************************************
;*             Comparation (Control vs Temp)                 *
;*************************************************************
comparation:
         lda ControlValue
cmp #65
blo Temp1_4
         cmp #130
blo Temp2_4
         cmp #195
         blo Temp3_4


         mov #04,ControlValue       ; selector = 4 (Coldest)
         lda ADCValue
         cmp #TEMP4_ON
         blo Compresor_ON
         cmp #TEMP4_OFF
         bhs Compresor_OFF
         rts

Temp3_4:
         mov #03,ControlValue       ; selector = 3 (Mid-Low)
         lda ADCValue
         cmp #TEMP3_ON
         blo Compresor_ON
         cmp #TEMP3_OFF
         bhs Compresor_OFF
         rts

Temp2_4:
```

```
        mov #02,ControlValue      ; selector = 2 (Mid-High)
        lda ADCValue
        cmp #TEMP2_ON
        blo Compresor_ON
        cmp #TEMP2_OFF
        bhs Compresor_OFF
        rts

Temp1_4:
        mov #01,ControlValue      ; selector = 1 (Hot)
        lda ADCValue
        cmp #TEMP1_ON
        blo Compresor_ON
        cmp #TEMP1_OFF
        bhs Compresor_OFF
        rts

Compresor_ON:                                   ; Compresor ON
        bset output,PTAD
        rts
Compresor_OFF:                                  ; Compresor OFF
        bclr output,PTAD
        rts
```

3.4 Schematic

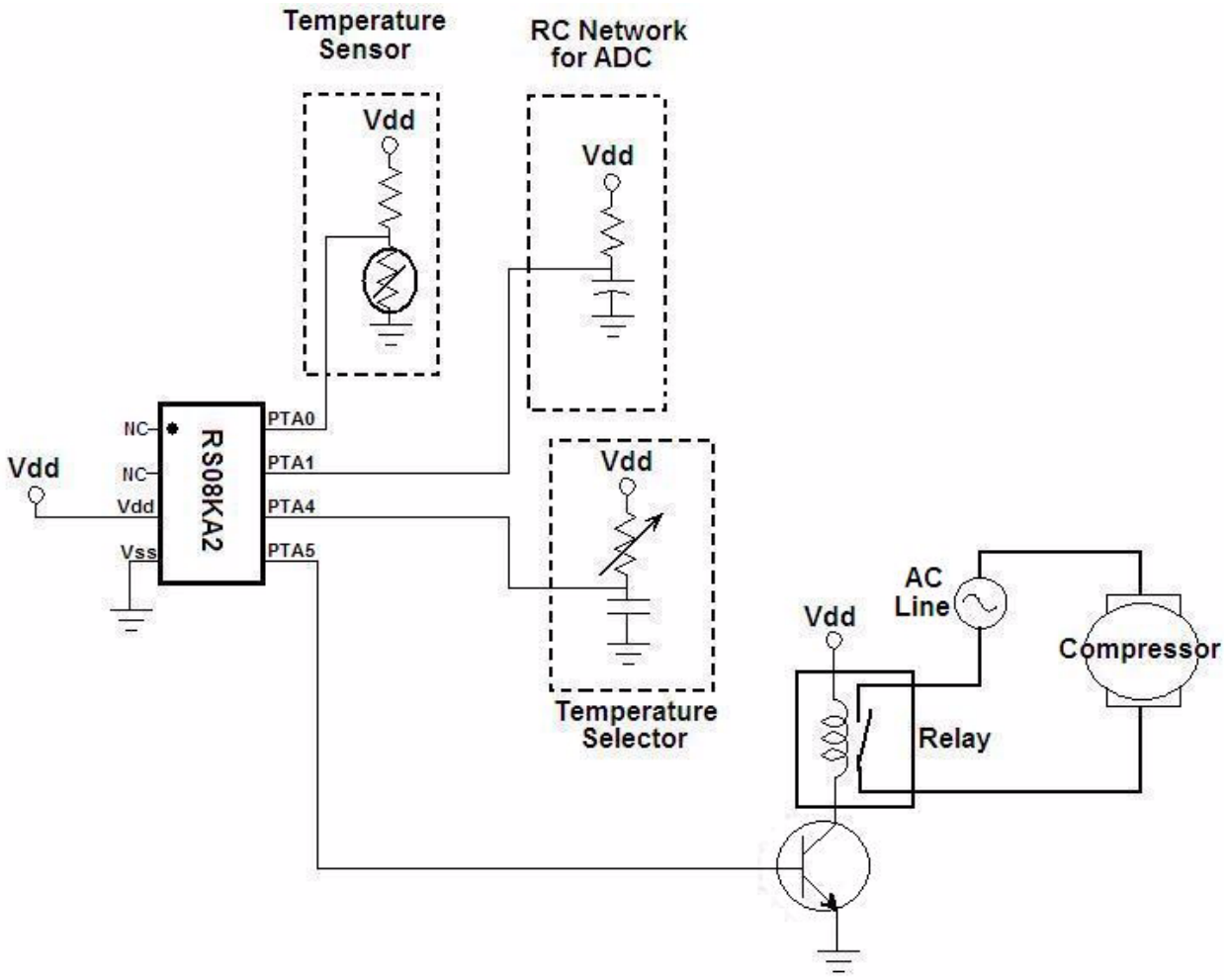


Figure 7. Hardware Schematic

4 Conclusion

This application note shows how to implement a simple on-off control system with a low-end 8-bit microcontroller.

Appendix A NCP18WB333J03RB Thermistor Range Table

Temp (°C)	Resistance (K)
-40	1227.263
-35	874.449
-30	630.851
-25	460.457
-20	339.797
-15	253.363
-10	190.766
-5	144.964
0	111.087
5	85.842
10	66.861
15	52.470
20	41.471
25	33.000
30	26.430
35	21.298
40	17.266
45	14.076
50	11.538
55	9.506
60	7.870
65	6.549
70	5.475
75	4.595
80	3.874
85	3.282
90	2.789
95	2.379
100	2.038
105	1.751
110	1.509
115	1.306
120	1.134
125	0.987

Appendix B Code Implementation

```
 INCLUDE 'derivative.inc'               ; Include derivative-specific definitions

; export symbols
            XDEF _Startup
            ABSENTRY _Startup


; Variable declarations
ACMP_ENABLE      SET   $92
ACMP_DISABLED    SET   $20
MTIM_INIT        SET   $50
MTIM_ENABLE      SET   $40
MTIM_STOP_RESET  SET   $30
MTIM_64_DIV      SET   $06
FREE_RUN         SET   $00
DEBUG_MODE       SET   $00
RUN_MODE         SET   $01


control                  SET $04
output                   SET $05

TEMP1_ON         SET 94
TEMP1_OFF        SET 100
TEMP2_ON         SET 98
TEMP2_OFF        SET 104
TEMP3_ON         SET 102
TEMP3_OFF        SET 108
TEMP4_ON         SET 106
TEMP4_OFF        SET 112

MODE:            EQU    DEBUG_MODE

; variable/data section
    ORG    RAMStart

ADCValue:        DS.B 1
counter          DS.B 1
ControlValue     DS.B 1


; code section
     ORG    ROMStart


;*************************************************************
;*                      MACRO DECLARATION                    *
;*************************************************************

TRIM_ICS: MACRO             ; Macro used to configure the ICS with TRIM
    mov #$FF,PAGESEL                ;change to last page
    lda  #$FA                       ; load the content which TRIM value is store
    tax                             ; move A content to X
    lda  ,x                         ; read D[X]
    sta  ICSTRM                     ; Store TRIM value
```

```
    ENDM

ACK_RTI: MACRO
    mov #HIGH_6_13(SRTISC), PAGESEL
    bset 6,MAP_ADDR_6(SRTISC)
    ENDM



;*************************************************************
;*             Comparation (Control vs Temp)                 *
;*************************************************************
comparation:
    lda ControlValue
    cmp #65
    blo Temp1_4
    cmp #130
    blo Temp2_4
    cmp #195
    blo Temp3_4


   mov #04,ControlValue ; selector = 4 (Coldest)
   lda ADCValue
   cmp #TEMP4_ON
   blo Compresor_ON
   cmp #TEMP4_OFF
   bhs Compresor_OFF
   rts

Temp3_4:
   mov #03,ControlValue; selector = 3 (Mid-Low)
   lda ADCValue
   cmp #TEMP3_ON
   blo Compresor_ON
   cmp #TEMP3_OFF
   bhs Compresor_OFF
   rts

Temp2_4:
   mov #02,ControlValue   ; selector = 2 (Mid-High)
   lda ADCValue
   cmp #TEMP2_ON
   blo Compresor_ON
   cmp #TEMP2_OFF
   bhs Compresor_OFF
   rts

Temp1_4:
   mov #01,ControlValue; selector = 1 (Hot)
   lda ADCValue
   cmp #TEMP1_ON
   blo Compresor_ON
   cmp #TEMP1_OFF
   bhs Compresor_OFF
   rts
```

```
Compresor_ON:              ; Compresor ON
   bset output,PTAD
   rts
Compresor_OFF:             ; Compresor OFF
   bclr output,PTAD
   rts

;************************************************************
;*                     CONFIGURES SYSTEM CONTROL             *
;************************************************************
Init_mc:
   mov #HIGH_6_13(SOPT), PAGESEL
    mov #$E3, MAP_ADDR_6(SOPT)      ;Enable STOP mode and COP with long timeout period

    clr  ICSC1                      ; FLL is selected as Bus Clock
    TRIM_ICS
    clr  ICSC2
    bset output,PTADD     ; Enable PTA5 as output
    rts

;************************************************************
;*            Modulus Timer Configuration for ADC            *
;************************************************************
 MTIM_ADC_Init:
    mov #MTIM_64_DIV,MTIMCLK         ; Select bus clock as reference, Set prescaler with 64
    mov #FREE_RUN,MTIMMOD             ; Configure Timer as free running
    mov #MTIM_STOP_RESET,MTIMSC
    rts

;************************************************************
;*                  ADC Single Conversion                    *
;************************************************************
ADC_Single_conversion:

   ; Discharge Capacitor
   bset 1,PTADD
   bclr 1,PTAD
    lda  #$FE
waste:
    dbnza waste

   ; Start Conversion
    mov #ACMP_ENABLE,ACMPSC           ; ACMP Enabled, ACMP+ pin active, Interrupt enabled,
Rising edges detections
    mov #MTIM_ENABLE,MTIMSC           ; Timer Counter Enabled
   wait                                     ; Wait for Analog Comparator Interrupt (match
signals)
    bset 4,MTIMSC                  ; Stop MTIM
    lda MTIMCNT                       ; read counter timer value
    sta ADCValue                   ; store counter value
    mov #MTIM_STOP_RESET,MTIMSC       ; Stop and reset counter

    mov #HIGH_6_13(SIP1), PAGESEL
    brset 3, MAP_ADDR_6(SIP1),Conv_OK ; branch if ACMP interrupt arrives
    bra ADC_Single_conversion

    ; Comparator Interrupt OK
```

```
Conv_OK:
    mov #ACMP_DISABLED, ACMPSC        ; ACMP Disabled, Clear Interrupt flag
    rts

;************************************************************
;*                     Control Value                        *
;************************************************************
Pin_Measure:
   bset control,PTADD     ; Set control pin as Output
   bclr control,PTAD               ; Discharge RC network
   clr ControlValue
    lda  #$FE
Discharge2:
    dbnza Discharge2
    bclr control,PTADD    ; Set Control pin as Input
measure_pin:
   inc ControlValue
   brclr control,PTAD,measure_pin; Inc value while pin is in low state
   rts


;************************************************************
;*               RTI Module Configuration                   *
;************************************************************
Init_RTI:
   mov #HIGH_6_13(SRTISC), PAGESEL
   mov #$37, MAP_ADDR_6(SRTISC)      ; Enable RTI (1 sec period)
   rts

;************************************************************
;*                          MAIN                            *
;************************************************************
_Startup:
   bsr Init_mc
   bsr Init_RTI
   bsr MTIM_ADC_Init                 ; Configure MITM for ADC module
    ; Application Loop
mainLoop:
   feed_watchdog                    ; Clear COP timer
   bsr ADC_Single_conversion  ; ADC Conversion
   bsr Pin_Measure                  ; Control Measure
   jsr comparation                  ; Comparation
   stop                                    ; Enter in STOP mode
   ACK_RTI                          ; Ack for RTI Interrupt
   bra mainLoop

;************************************************************
;*                 Startup Vector                           *
;************************************************************
      ORG   $3FFD
      JMP _Startup                     ; Reset
```

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: AN3409
Rev. 0
01/2007

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

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

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

© Freescale Semiconductor, Inc. 2007. All rights reserved.

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

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

freescale™
semiconductor

Freescale Semiconductor
Application Note

Document Number: AN3410
Rev. 1, 02/2007

Implementing a Sewing Machine Controller with an MC9RS08KA2 Microcontroller

by: Ulises Corrales Salgado
Applications Engineer
RTAC Americas

1 Introduction

This application note explains how to control motor speed using an MC9RS08KA microcontroller.

Many machines or devices are made entirely with mechanical parts, making maintenance a problem. Electronic devices are easier to maintain and cheaper to produce than mechanical devices.

This document explains how to design a low-cost, digital motor-speed control using an accelerometer and an MC9RS08KA2 microcontroller. This option can replace the pulleys and gears on ordinary mechanical devices, reduce cost, and increase functionality.

1.1 Description

Sewing machines can be classified into three types, depending on their build. The first type is made with mechanical parts and is difficult to use because, to control the motor's speed, you must use both feet to move the pedal (Figure 1).

Contents

© Freescale Semiconductor, Inc., 2007. All rights reserved.

freescale™ semiconductor

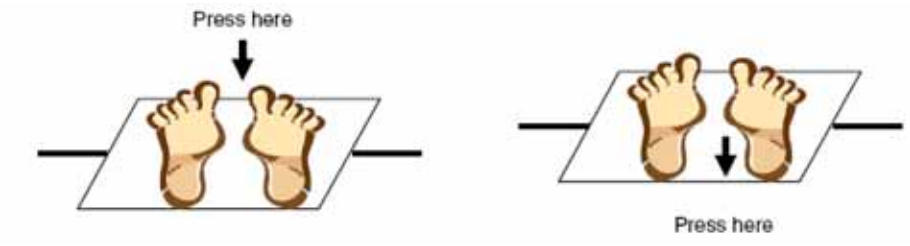


Figure 1. Using a Manual Sewing Machine

The second type has a pedal made with gauges that function like a variable resistor. The resistor values vary depending of the information of the gauge. When you press the gauge, the motor speeds up. When you release it, it slows down (Figure 2).

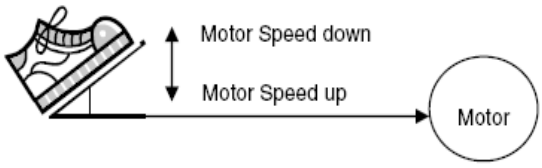


Figure 2. Using an Electric Sewing Machine

The third type of sewing machine has an accelerometer sensor[1] in its pedal. The accelerometer measures the tilt of the pedal. With this, you can speed up or speed down the motor (Figure 3).

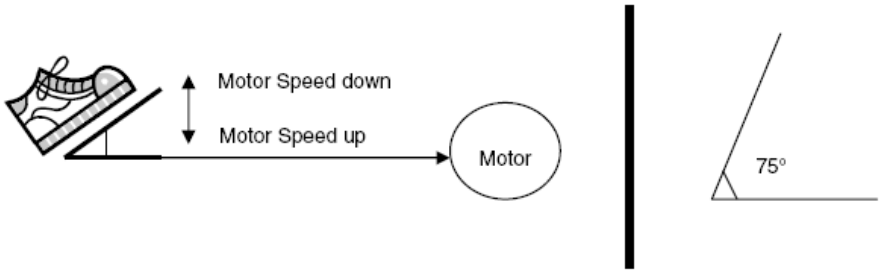


Figure 3. Using a Sewing Machine with an Accelerometer

The accelerometer measures acceleration and angles. In this application, it measures angles from 0° to 75°.

Figure 4 shows the blocks that make up the three types of sewing machines. The yellow blocks (also outlined with thicker lines) indicate the application note's focus.

1. http://www.freescale.com/files/sensors/doc/data_sheet/MMA7260QT.pdf

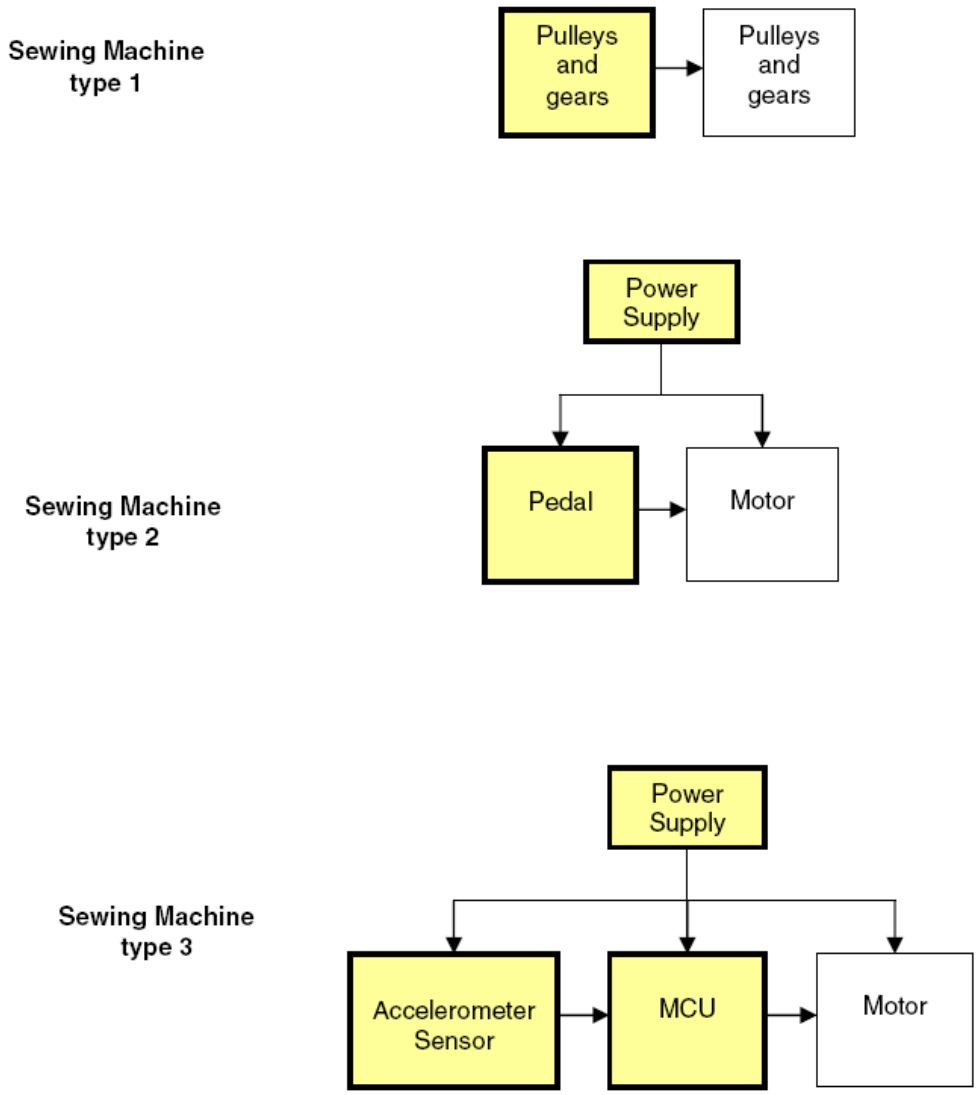


Figure 4. Block Diagrams for Three Sewing Machine Types

See 2.1, "Solution Benefits," for the advantages of our solution.

1.2 Design Requirements

To create the accelerometer controller, you need:

- One MC9RS08KA2 microcontroller
- One MMA7260QT accelerometer sensor
- The power stage from the MCU to the DC motor, depending on the motor characteristics

2 Solution

Our solution depends on the accelerometer sensor's data. The main program reads the analog-to-digital converter (ADC). After the KA2 acquires the data, the microcontroller processes it and generates a pulse-width modulation (PWM), which speeds up or slows the motor. The accelerometer sensor is located in the foot pedal. The motor speed depends on the degree of tilt (Figure 5).

Figure 5. Accelerometer Location

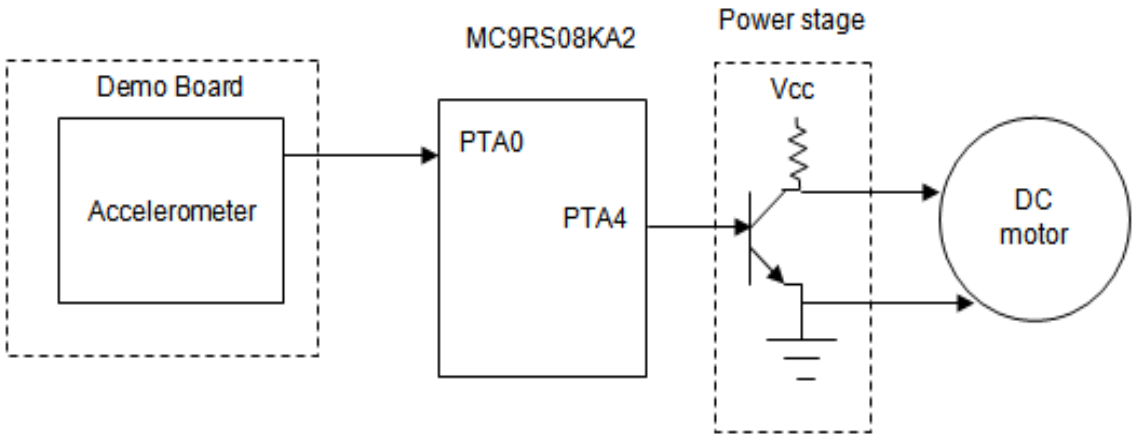


Figure 6. Sewing Machine Block Diagram

NOTE

This device was made using a DC motor. This device will not work using an AC motor. The power-stage block varies depending on the motor characteristics. Also, the MC9RS08KA2 microcontroller does not have an ADC module, so an RC circuit and software obtains the ADC values.

The accelerometer demo board was used in this application note. The accelerometer demo board has an MC9S08QG that configures the sensitivity sensor and also keeps the accelerometer working in active mode. There is no documentation regarding this board except for a schematic (Appendix A, “ Firmware”). You can solve this problem by using hardware such as in Figure 7.

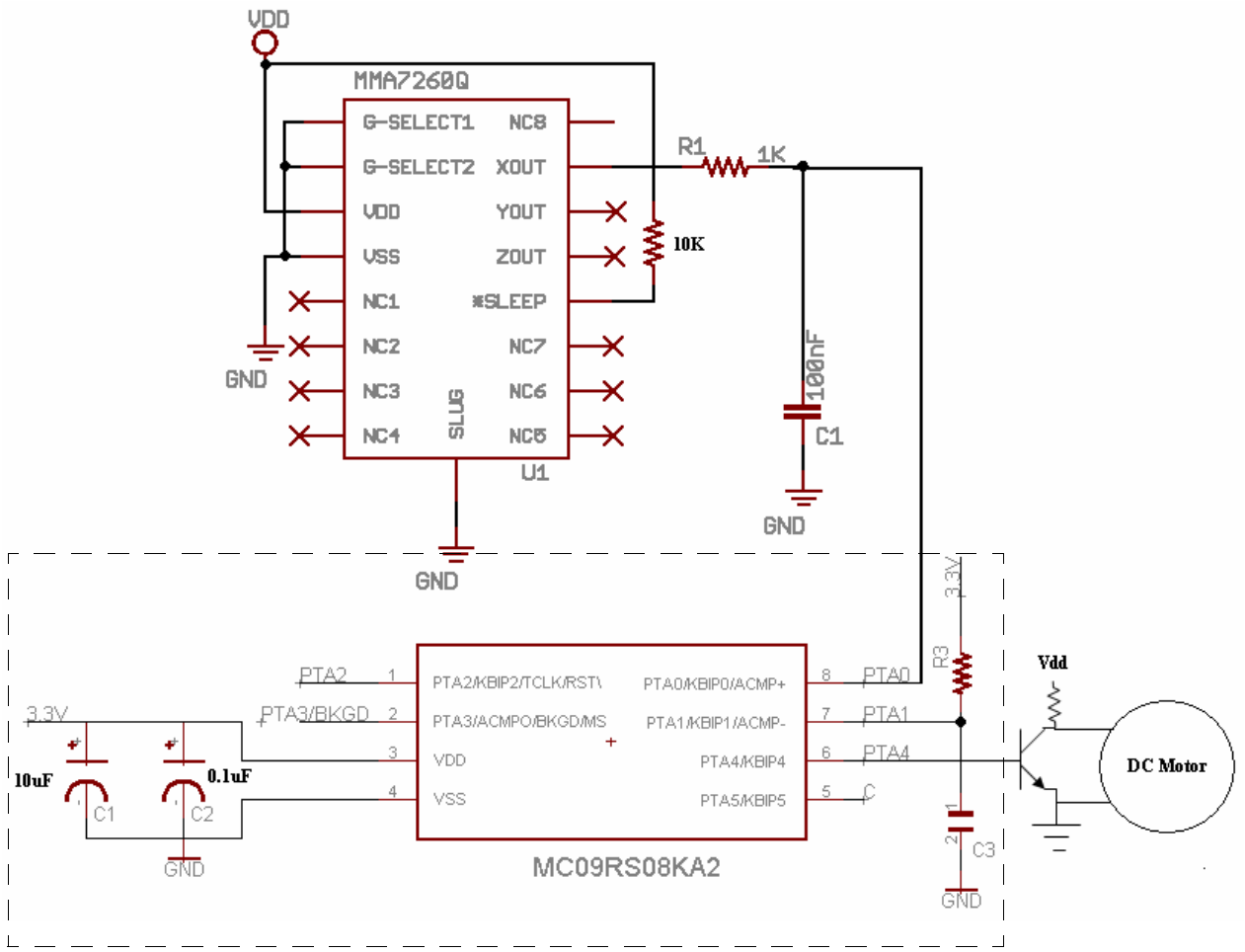


Figure 7. Sewing Machine Schematic Diagram

NOTE

The MCU connection inside the dashed square are part of the KA2 evaluation board. You should make the DC motor connections and accelerometer connections.

2.1 Solution Benefits

Device benefits:

- Smaller foot pedal
- Pedal is not temperature-sensitive. If the temperature varies, the ADC reading is the same.
- The foot pedal is cheaper than a gauge pedal.
- Detects tilt changes more precisely, allowing better control over motor speed.

3 Detailed Description

The MCU firmware first configures the microcontroller (this routine configures the bus clock and disables the COP). After that, the modulo timer is configured. Next, the discharge capacitor is configured.

Data table — This part of the code is used to get values between 0 and 255. The values the SensorReading variable obtains are between 40 and 75. These values cannot be used in the PWM routine because the range values are too short, so adjustments are made to these values.

Lookup table —This part of the code does a page calculation to obtain the data from the data table. In this project, the ADC reads 35 values approximately between 0° to 75°. The values must be extrapolated between 0 and 255 because the lowest value the ADC obtains (0° of tilt) is 40, this value has to extrapolate to 255. You can only obtain the extrapolation values by subtracting seven from the previous value. The maximum extrapolation value is 255.

For example, if you want to use eight values rather than all the values the ADC obtained, extrapolate new values and enter them in the data table in the firmware project. Table 1 and Table 2 show the extrapolation values made in the project and the example extrapolation values obtained.

NOTE

The ADC values are always the same, but you can use the values you want. The ADC values you do not choose are obtained by the extrapolated value of its predecessor. In the example, the values between 40 and 44 are extrapolated to the 255 value.

Table 1. Values from the Project

Values Obtained by ADC	Degree of Tilt (Approximately)	Extrapolation
40	0	255
41	2.2	248
42	4.4	241
43	6.6	234
…	…	…
71	68.2	38

Table 1. Values from the Project (continued)

Values Obtained by ADC	Degree of Tilt (Approximately)	Extrapolation
72	70.4	31
73	72.6	24
74	74.8	17
75	75	0

Table 2. Example Values

Desired ADC values	Extrapolation
40	255
45	220
50	185
55	150
60	115
65	80
70	45
75	0

```
ORG

dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,255,248,241,234,227,220,213,206
dc.b
199,192,185,178,171,164,157,150,143,136,129,122,115,108,101,94
dc.b 87,80,73,66,59,52,45,38,31,24,17,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Figure 8. Extrapolated Values from the Data Table

PWM — The motor speed is controlled in this section of code. There are two main routines: PWMLoopOn and PWMLoopOff.

- The PWMLoopOn routine activates the PTA4 pin, which causes the motor to spin.
- The PWMLoopOff deactivates the PTA4 pin, causing the motor to stop.

The PWM value determines the highest speed a motor can reach and, in this case, the ConvertedValue value.

Configure MTIM — In this section, the MTIM module is configured, and the ACMP module is disabled. A subtraction is executed in this section: 255 – ConvertedValue. The result is saved in the complement variable. This variable is used to turn off the motor in the PWM routine.

User constants — Reserved memory sections that cannot be modified.

User variables — Include values, erase, etc., that can be changed during the program.

Macro definitions — Here is where the internal clock source (ICS) is configured.

Configures port A — This section configures the port A pins. In this program PTA5, PTA4, and PTA1 are configured as outputs.

To understand the project code better, see the program-flow diagram (Figure 9).

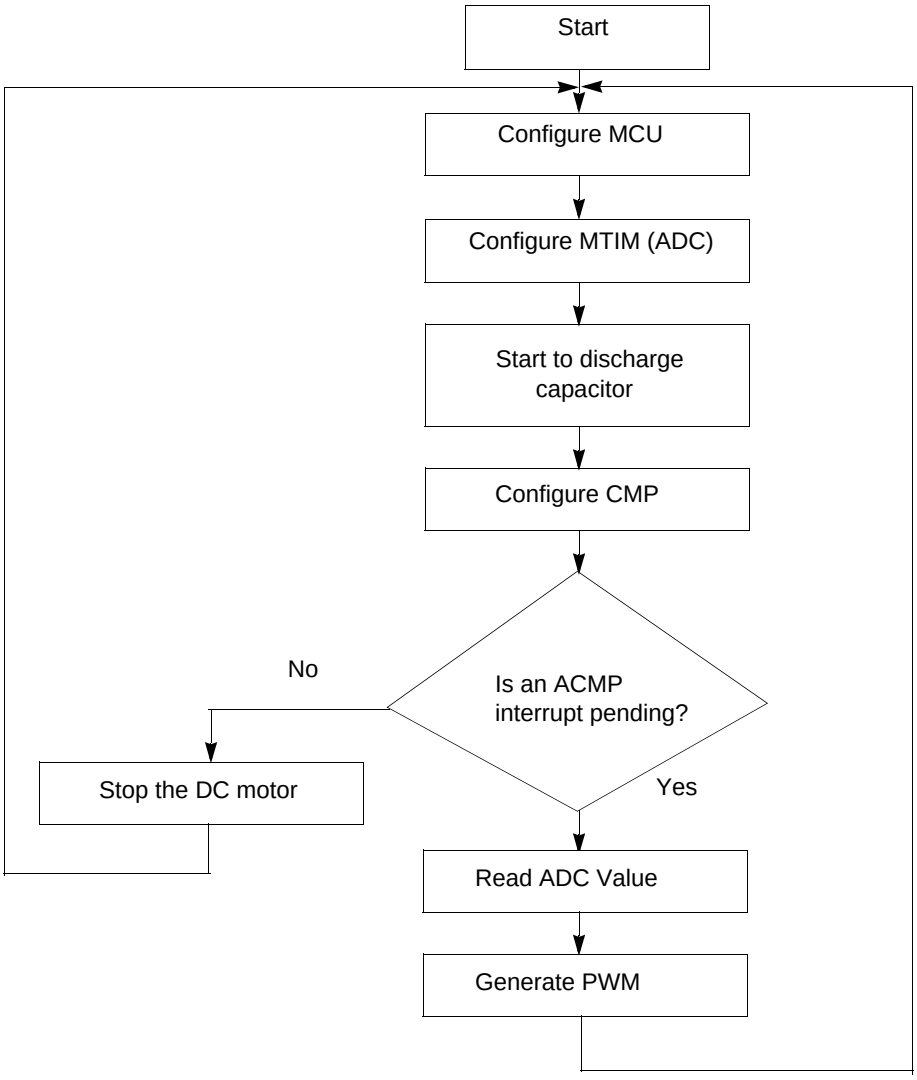


Figure 9. Program Flow

4 How to Download the Program to Flash Memory

To download this project:

1. Open the CodeWarrior™ version 5.1 development tool.
2. Open AppNote.mcp file.
3. Select the option SofTec RS08.
4. Click on the make button.
5. Click on the debug button.
6. Select the DEMO9RS08KA2 board.
7. Click OK.

5 Conclusion

Electronic devices are easier to maintain and support than mechanical or gauge devices. You have better control with an electronic device than a mechanical one. This application note explained how to control a sewing machine's motor speed with an MC9RS08KA microcontroller.

6 References

http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9RS08KA2.pdf

http://www.freescale.com/files/sensors/doc/app_note/AN3107.pdf?fsrch=1

http://www.freescale.com/files/sensors/doc/data_sheet/MMA7260QT.pdf

7 Glossary

AC — Alternating current

ACMP — Analog comparator

ADC — Analog-to-digital converter

COP — Computer operating properly (watchdog)

DC — Direct current

MCU — Microcontroller unit

MTIM — Modulo timer

PWM — Pulse-width modulation

RC — Resistor capacitor

Appendix A
Firmware

```
;*********************************************************************
;*                            MAIN.ASM                               *
;*********************************************************************
;*          Copyright (c) 2007 Freescale Semiconductor             *
;*          http://www.freescale.com/                              *
;*********************************************************************
;* Implementing a Sewing machine controller with a MC9RS08KA2      *
;* microcontroller. The speed motor depends of tilt pedal.         *
;*********************************************************************
;* Ulises Corrales Salgado                                          *
;* Applications Engineer                                            *
;* RTAC Americas                                                    *
;*********************************************************************

; Include derivative-specific definitions
            INCLUDE 'derivative.inc'
;
; export symbols
;
            XDEF _Startup
            ABSENTRY _Startup

;*****************************************************************************
;*                             User Constants                                *
;*****************************************************************************

Table_Data EQU $3E00

; Variable declarations
ACMP_ENABLE      SET   $92
ACMP_DISABLED    SET   $20
MTIM_INIT        SET   $50
MTIM_ENABLE      SET   $60
MTIM_STOP_RESET  SET   $30
MTIM_128_DIV     SET   $07
FREE_RUN         SET   $00
```

```
DEBUG_MODE       SET   $00
RUN_MODE         SET   $01

MODE:            EQU   DEBUG_MODE


;*******************************************************************************
;*                            User Variables                                   *
;*******************************************************************************
;
; variable/data section
;
            ORG    RAMStart         ; Insert your data definition here

ConvertedValue: DS.B   1            ; This varible store converted value
Complement      DS.B   1
Temp_Page       DS.B   1            ; Temporal backup Page
SensorReading   DS.B   1
PcBuffer        DS.B   2            ; Temporal backup SPC

            ORG    ROMStart


;*************************************************************
;*                       MACRO DEFINITIONS                   *
;*************************************************************
TRIM_ICS: MACRO                   ; Macro used to configure the ICS with TRIM
  MOV   #$FF,PAGESEL              ; change to last page
  LDX   #$FA                      ; load the content which TRIM value is store
  LDA   ,x                        ; read D[X]
 CBEQA #$FF,No_Trim    ;Omit the 0xFF value if $3FFA location content

                                   ; the value
  STA   ICSTRM                    ; Store TRIM value into ICSTRM register
No_Trim:
  ENDM

ENTRY_SUB: MACRO                   ; Macro for "stacking" SPC
            SHA
```

```
            STA PcBuffer + 2*(\1)
            SHA
            SLA
            STA PcBuffer + 2*(\1) +1
            SLA
  ENDM

  NOP                           ; Needs to separate MACROS

EXIT_SUB: MACRO                  ; Macro for restore SPC
            SHA
            LDA PcBuffer + 2*(\1)
            SHA
            SLA
            LDA PcBuffer + 2*(\1) +1
            SLA
  ENDM


;*************************************************************
;*                       CONFIGURES PORT A                   *
;*************************************************************
PortA:
  MOV #HIGH_6_13(PTAPE), PAGESEL
  MOV #$FE, MAP_ADDR_6(PTAPE)    ; Enables internal Pulling device

  MOV #HIGH_6_13(PTAPUD), PAGESEL
  MOV #$04,MAP_ADDR_6(PTAPUD)    ;Configures Internal pull up device in PTA  ; 5

  MOV #$32, PTADD                ; PTA5, PTA4 and PTA3 as outputs
  MOV #$00, PTAD

  RTS

;*************************************************************
;*                     CONFIGURES SYSTEM CONTROL             *
;*************************************************************
Init_Config:
  IFNE  MODE
```

```
  MOV #HIGH_6_13(SOPT), PAGESEL
   MOV  #$01,  MAP_ADDR_6(SOPT)       ;  Disables  COP  and  enables  RESET  (PTA2)  pin

  ELSE
  MOV #HIGH_6_13(SOPT), PAGESEL
  MOV #$03, MAP_ADDR_6(SOPT)      ; Disables COP, enables BKGD (PTA3) and RESET
                                  ; (PTA2) pins
  ENDIF
  CLR  ICSC1                      ; FLL is selected as Bus Clock
  TRIM_ICS
  CLR  ICSC2
  RTS

;*************************************************************
;*          Analog Comparator Initial Configuration          *
;*************************************************************
ACMP_Conf:
      MOV #ACMP_ENABLE,ACMPSC     ; ACMP Enabled, ACMP+ pin active, Interrupt
                                  ;enabled, Rising edges detections
      RTS

;*************************************************************
;*            Modulus Timer Configuration for ADC            *
;*************************************************************
 MTIM_ADC_Init:
      MOV #MTIM_128_DIV,MTIMCLK     ; Select bus clock as reference, Set prescaler
                                   ; with 64
      MOV #FREE_RUN,MTIMMOD        ; Configure Timer as free running
      MOV #MTIM_STOP_RESET,MTIMSC
      RTS



;*************************************************************
;*                    Discharge Capacitor                    *
;*************************************************************
Discharge_Cap:
      BSET  1,PTADD                ; Configure PTA1 as Output
      BCLR  1,PTAD                 ; Start Capacitor discharging
      LDA   #$FE                   ; Set delay time
```

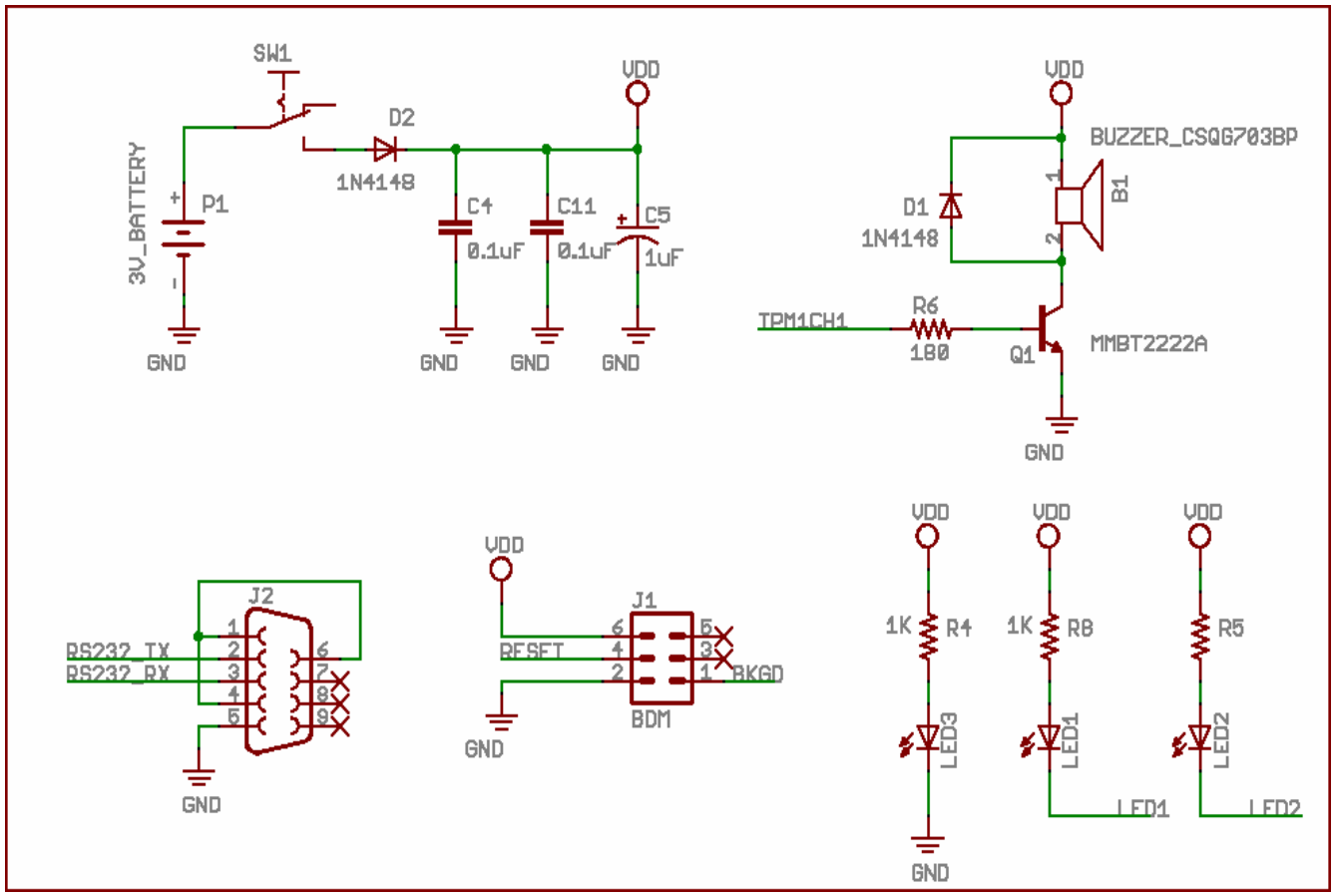
```
Waste_time:
      DBNZA Waste_time              ; Wait until Delay = 0
      RTS


_Startup:

  BSR Init_Config
  BSR PortA
NextValue:
  BSR MTIM_ADC_Init
  BSR Discharge_Cap
  BSR ACMP_Conf                     ; Configure ACMP+ and ACMP-
  MOV #MTIM_ENABLE,MTIMSC           ; Timer Counter Enabled

mainLoop:
  WAIT                              ; Wait for ACMP interrupt
  BSET 4,MTIMSC
  LDA MTIMCNT
  STA SensorReading                 ; Store counter value
  MOV #HIGH_6_13(SIP1), PAGESEL
  BRSET 3, MAP_ADDR_6(SIP1),PWM     ; Branch if ACMP interrupt arrives
  BCLR 7,ACMPSC
  BRA NextValue


PWM:

  MOV #MTIM_STOP_RESET,MTIMSC       ; Stop and reset counter
  MOV #ACMP_DISABLED, ACMPSC        ; ACMP Disabled, Clear Interrupt flag

  LDA SensorReading
  CMP #75
  BHS NextValue
  JSR LookupTable


;*************************************************************
;*                     Configure MTIM                        *
```

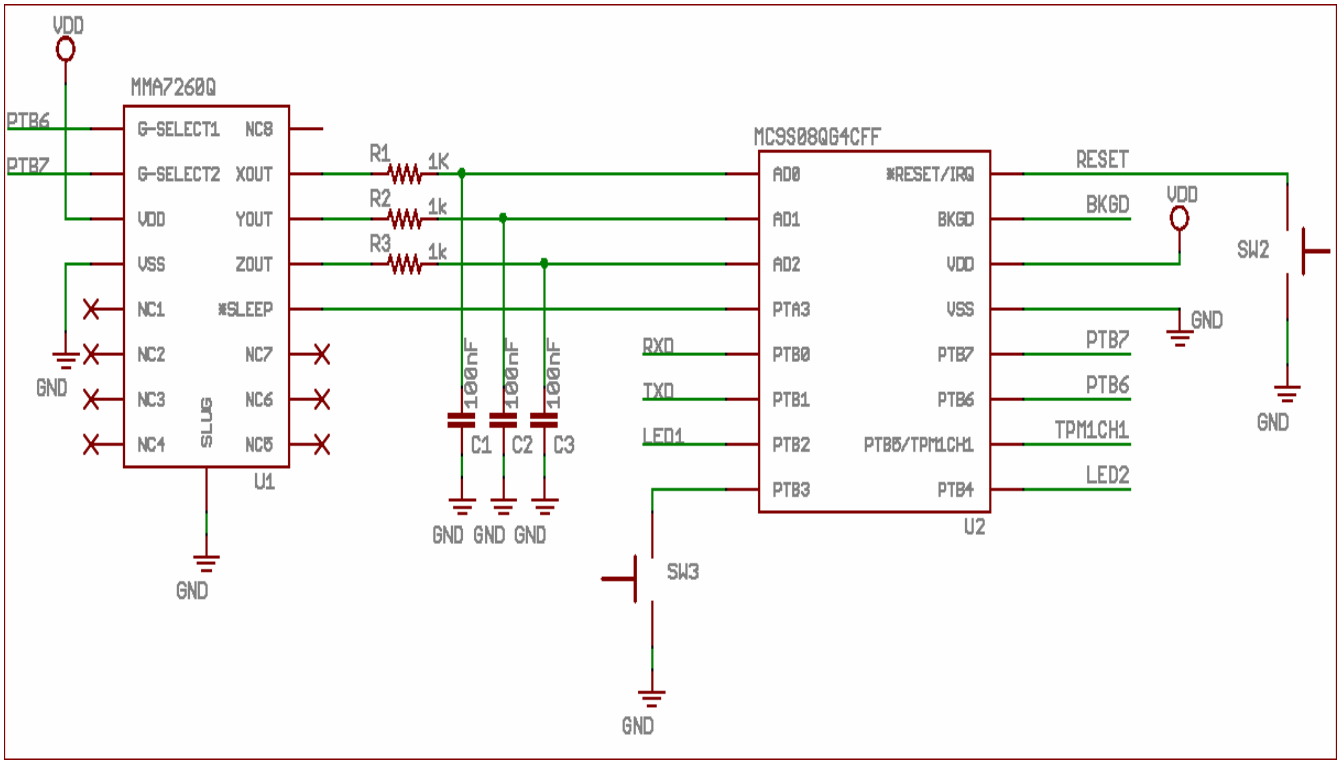
```
;*************************************************************
Activecounter:
  MOV #$00,MTIMCLK          ; Enables interrupt, stops and resets timer counter
  MOV #$01,MTIMMOD          ; MTIM modulo with 1 counts before interrupt.
  MOV #$70,MTIMSC            ; Selects internal clock as reference bus clock (4 MHz)
                            ; with prescaler 1

  MOV #MTIM_STOP_RESET,MTIMSC       ; Stop and reset counter
  MOV #ACMP_DISABLED, ACMPSC        ; ACMP disabled, Clear interrupt flag

  LDA #$FF
  SUB ConvertedValue
  STA Complement

  BCLR 4,MTIMSC                     ; MTIM counter is Active
  BRA PWMLoopOn

  ;*************************************************************
;*                           PWM                               *
;*************************************************************

PWMLoopOn:
BRSET 7,MTIMSC,PWM_Isr_D          ; Branch if timer interrupt pending
BRA PWMLoopOn
PWM_Isr_D:
BSET 5,MTIMSC                     ; Reset MTIM Counter, Clear overflow flag
BRA PWM_Set_D
PWM_Set_D:
BSET 4,PTAD                       ; Turn on led 4
DBNZ ConvertedValue,PWMLoopOn

PWMLoopOff:
BRSET 7,MTIMSC,PWM_Isr_D1         ; Branch if timer interrupt pending
BRA PWMLoopOff
PWM_Isr_D1:
BSET 5,MTIMSC                     ; Reset MTIM Counter, Clear  overflow flag
BRA PWM_Clear
PWM_Clear:
BCLR 4,PTAD                       ; Turn off led 4
```

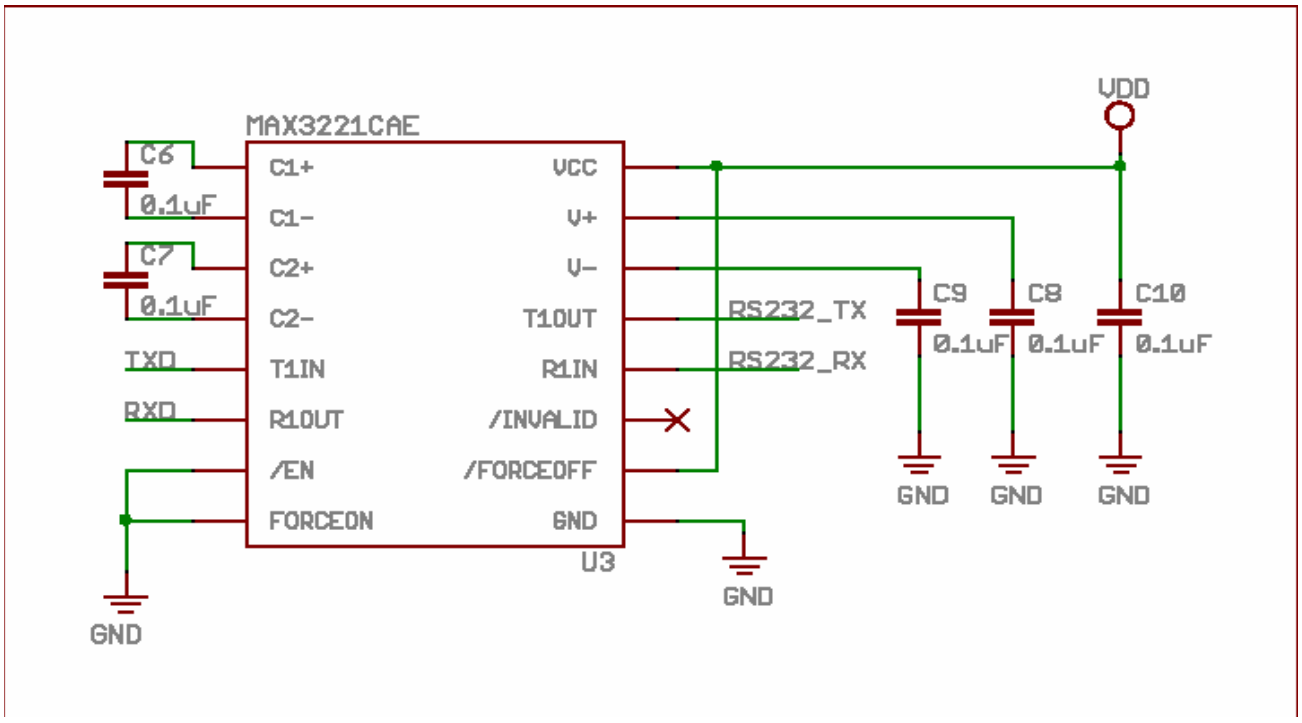
```
DBNZ Complement,PWMLoopOff

BRA NextValue


LookupTable:
      LDA SensorReading
      ROLA                          ; Getting 2 MSB
      ROLA
      ROLA
      AND #$03
      ADD #(Table_Data>>6)          ; Page Calculating
      MOV #PAGESEL,Temp_Page        ; Backup actual page
      STA PAGESEL                   ; Page Change
      LDA SensorReading
      AND #$3F                      ; Extract 6 LSB
      ADD #$C0                      ; Index to paging window
      TAX
      LDA ,x                        ; Load table result
      STA ConvertedValue            ; Store result
      MOV #Temp_Page, PAGESEL        ; Back Page
      RTS

;*************************************************************
;*                  Startup Vector                           *
;*************************************************************
            ORG   $3FFD

            JMP _Startup          ; Reset









;*************************************************************
;*                     Data Table                            *
;*************************************************************
 ORG Table_Data
```

```
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,255,248,241,234,227,220,213,206
dc.b 199,192,185,178,171,164,157,150,143,136,129,122,115,108,101,94
dc.b 87,80,73,66,59,52,45,38,31,24,17,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Appendix B Accelerometer Demo Board Schematic




VDD
MMA7260Q
PTB6
PTB7
G-SELECT1
G-SELECT2
VDD
VSS
NC1
NC2
NC3
NC4
NC8
XOUT
YOUT
ZOUT
*SLEEP
NC7
NC6
NC5
SLUG
U1
GND
R1
R2
R3
1K
1k
1k
100nF
100nF
100nF
C1
C2
C3
GND GND GND
MC9S08QG4CFF
AD0
AD1
AD2
PTA3
PTB0
PTB1
PTB2
PTB3
*RESET/IRQ
BKGD
VDD
VSS
PTB7
PTB6
PTB5/TPM1CH1
PTB4
U2
RESET
BKGD
VDD
SW2
GND
PTB7
PTB6
TPM1CH1
LED2
GND
RXD
TXD
LED1
SW3
GND


MAX3221CAE
C6
0.1uF
C7
0.1uF
C1+
C1-
C2+
C2-
T1IN
R1OUT
/EN
FORCEON
VCC
V+
V-
T1OUT
R1IN
/INVALID
/FORCEOFF
GND
U3
TXD
RXD
GND
VDD
RS232_TX
RS232_RX
C9
C8
C10
0.1uF
0.1uF
0.1uF
GND
GND
GND
GND

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 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: AN3410
Rev. 1
02/2007

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.

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

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

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. 2007. All rights reserved.

freescale™
semiconductor

Piano and Dimming Light Using Ultra Low-End MCU and Electric-Field Sensor

Designer Reference Manual

RS08
Microcontrollers

DRM085
Rev. 0
10/2006

freescale.com

freescale™
semiconductor

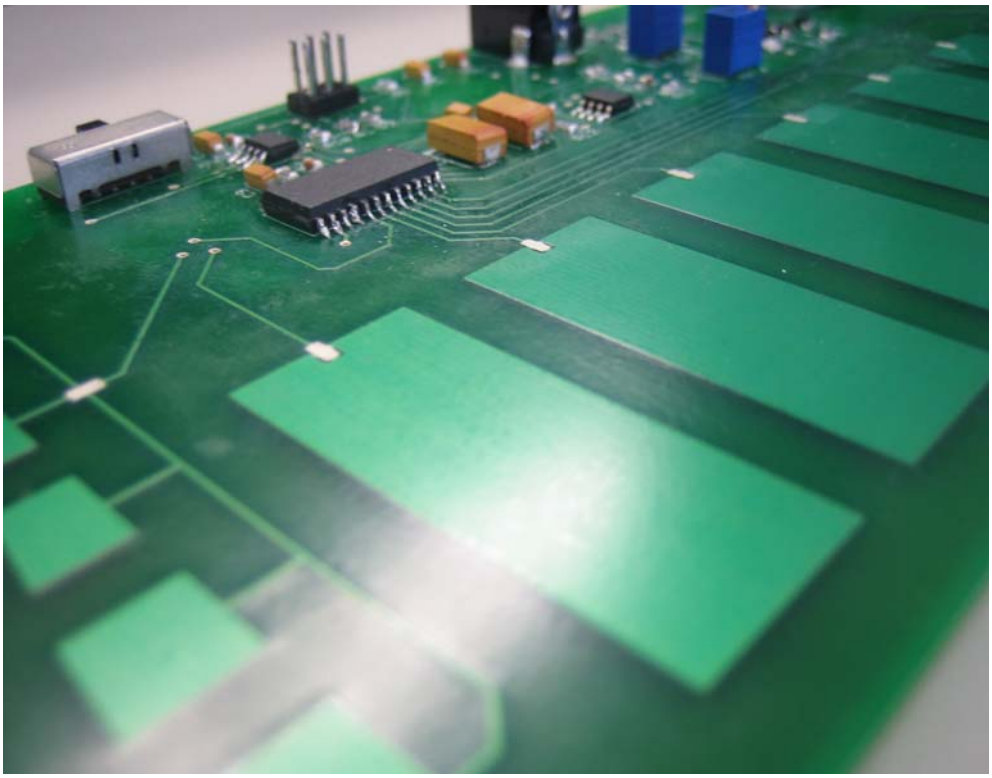
Piano and Dimming Light Using Ultra Low-End MCU and Electric-Field Sensor

Designer Reference Manual

by: Ulises Corrales
Manuel Davalos
Allan Led Collins

RTAC Americas

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 that you have the latest information available, refer to http://www.freescale.com

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

Revision History

Date	Revision Level	Description	Page Number(s)
07/2006	0	Initial release	N/A

Chapter 1
Designer Reference Manual Usage Notes

Chapter 2
Hardware Description

Chapter 3
Firmware Description

Appendix A BOM and Schematics

Appendix B Glossary

Chapter 1
Designer Reference Manual Usage Notes

1.1 Intended Application Functionality

This reference design demonstrates the application of Freescale's MC9RS08KA2 microcontroller and the MC34940 electrical field. This demonstration shows the connection between MC9RS08KA2 and the electrical-field sensor.

This demonstration can play seven musical notes (C, D, E, F, G, A, B) and be heard in a buzzer. A switch can change a piano function to a dimmer-light function with one electrode divided into five parts of varying intensity. One electrode connects to the electrical field when the demonstration works on a dimmer function. When the demonstration works in piano function, the electrical field is connected to seven electrodes.

The MC9RS08KA2 PADL reference design is intended for the toy industry and for use in some lighting devices (for example, lamps).

1.2 Quick Start

The demonstration shows the connection between the MC9RS08KA2 ultra low-end, 8-bit microcontroller and the MC34940, an electrical-field imaging device.

1.2.1 System Requirements

This demonstration needs only a 12 V DC external power supply to work properly.

1.2.2 MC9RS08KA2 PADL Setup

The MC9RS08KA2 PADL demonstration requires no setup. The board is distributed with the application loaded in the MC9RS08KA2 flash memory. Changing between functions depends on the switch position. Figure 1-1 and Figure 1-2 are pictures of the reference design board.

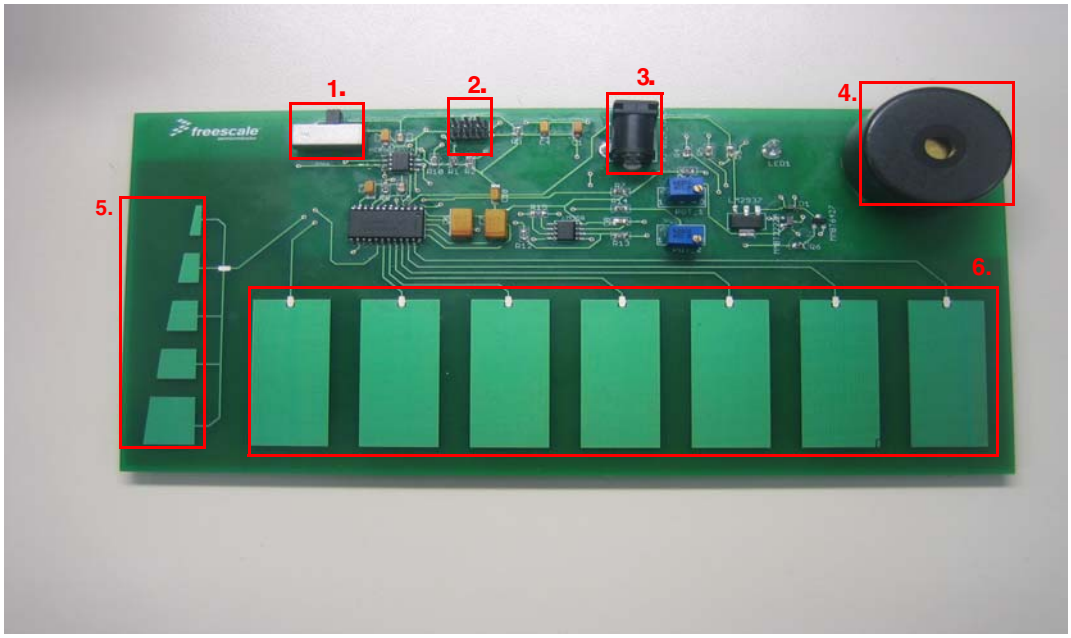


Figure 1-1. MC9RS08KA2 PADL Board (Front)

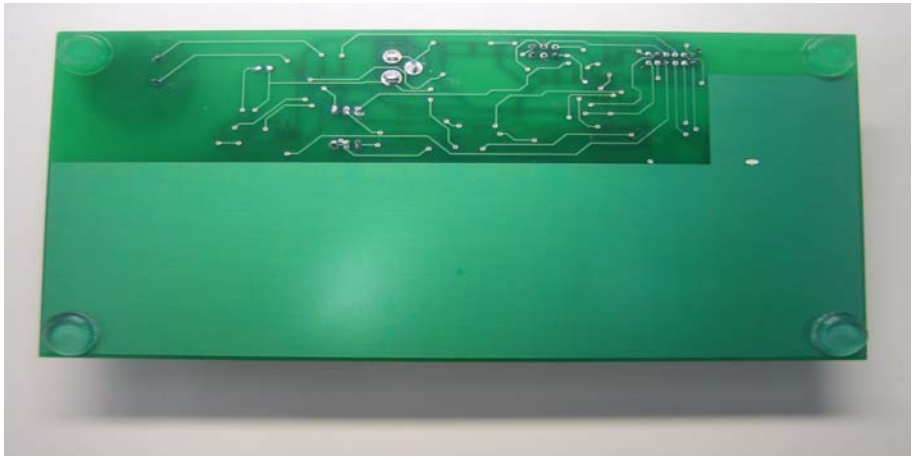


Figure 1-2. MC9RS08KA2 PADL Board (Back)

1. Switch — Selects between piano and dimmer light.
2. BDM connector
3. Power supply
4. Buzzer
5. Dimmer electrodes
6. Piano electrodes

To run the demonstration:

1. Put the switch in position 1 or position 2.
   a. Position 1 selects dimming mode.
   b. Position 2 selects piano mode.
2. Connect the 12 V DC power supply to the demo board.
3. Start pressing the electrodes.

Chapter 2 Hardware Description

2.1 Introduction

This section describes the module design and features.

Figure 2-1 shows the basic blocks of each reference design component.

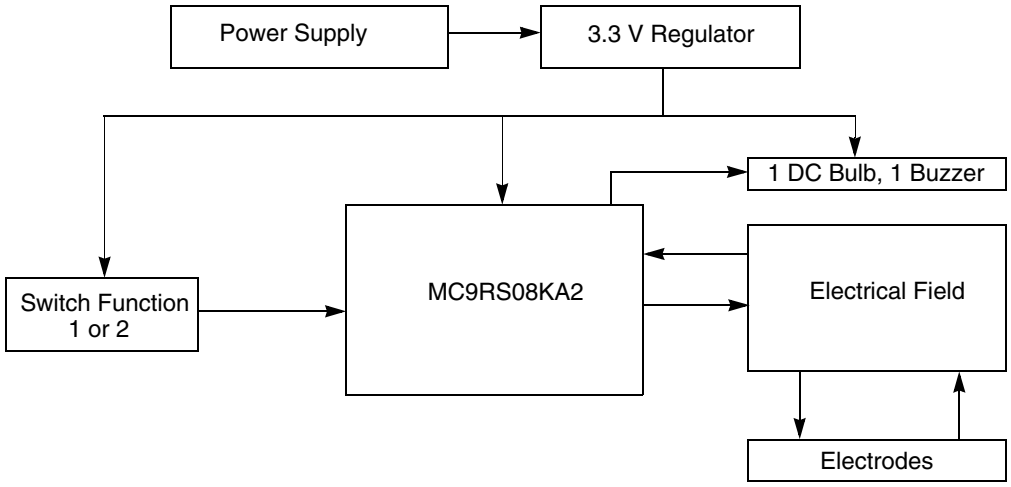


Figure 2-1. MC9RS08KA2 PADL Building Block

This demo has two functions (both use the electrical-field sensor):

- Piano
- Light dimmer

2.1.1 Piano Function

In the piano function, MC9RS08KA2 is connected to the electrical field (MC34940), the seven electrodes (from the seven different musical notes), and a buzzer (Figure 2-2).

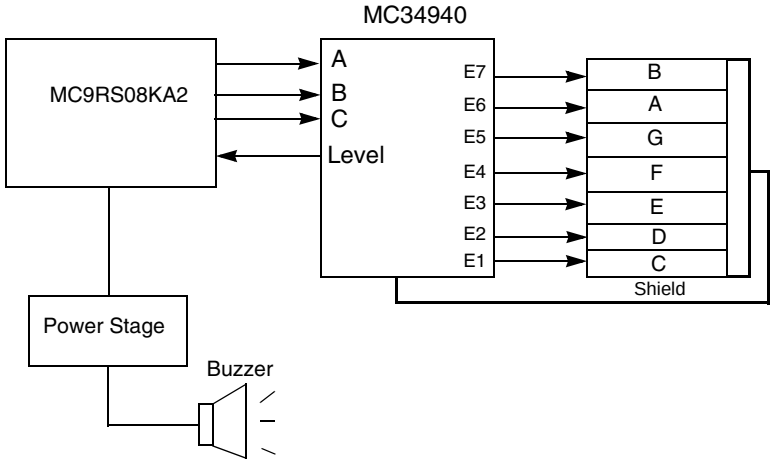


Figure 2-2. MC9RS08KA2 PADL Function 1 Logic Diagram

2.1.2 Dimmer Light Function

When the demo runs in dimmer light function, the MC9RS08KA2 MCU is connected to an electrical field and DC bulb. The electrical field is also connected to a divided electrode. Figure 2-3 shows the logic block diagram for function 2.

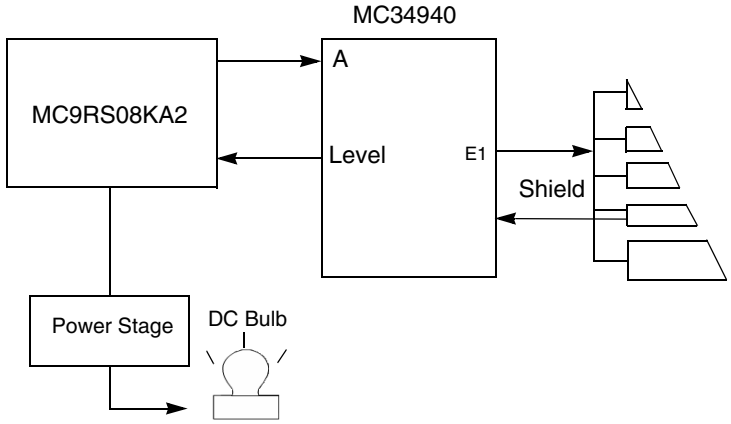


Figure 2-3. MC9RS08KA2 PADL Function 2 Logic Diagram

2.2 Technical Data

This section provides technical details of the MC9RS08KA2 PADL and components in this reference design.

2.2.1 MC9RS08KA2 Microcontroller Unit

The MC9RS08KA2 is the control unit for the PADL and part of the RS08 family.

MC9RS08KA2 features:

- 8-bit RS08 CPU
  – Analog comparator (ACMP)

  - Keyboard interrupt (KBI)
  - Pending interrupt indication
  - Background debug mode (BDM)
- Reset, clock, COP watchdog
- 6-bit port with digital filtering and programmable rising- or falling-edge trigger
- Memory
  - 2K flash EEPROM
  - 63 bytes RAM

MC9RS08KA2 functions:

- Controls the MC34940 (electrical field)
- Generates the pulse width modulation (PWM) for the dimming light and musical notes
- Switches between PADL functions
- Processes data

2.2.2 Electrical-Field Imaging Device (MC34940)

MC34940 is a sensor that detects objects in an electrical field. In this demonstration, it detects the user's hands.

Features:

- Supports up to nine electrodes and two references or electrodes
- Shield driver for driving remote electrodes through coaxial cables
- Lamp driver output
- Watchdog and power-on reset timer
- High-purity sine wave generator tunable with external resistor

Function:

- Relays electrode status to microcontroller

2.3 MC9RS08KA2 PADL Reference Design Architecture

2.3.1 MC9RS08KA2

The MC9RS08KA2 controls the application.

The application occupies these modules:

- General-purpose input/output pins (GPIO)
- Analog comparator (ACMP)
- Modulo timer (MTIM)

2.3.2 Electrical Field (MC34940)

The MC34940 generates a low radio frequency sine wave with nominal 5 V peak-to-peak amplitude. An internal multiplexer routes the signal to one of seven terminals under the ABC input terminals. A receiver multiplexer connected to the selected electrode simultaneously routes its signal to a detector that converts the sine wave to a DC level. The DC level is filtered by an external capacitor, and that value is multiplied so the value offset is less sensitive to objects near the signal.

A capacitor is formed between the driving electrode and the object, each forming a plate that holds the electrical charge. The voltage measured is an inverse function of the capacitance among the electrode being measured, the surrounding electrodes, and other objects in the electrical field surrounding the electrode. Increasing capacitance decreases voltage.

The basic building block for these devices is found in Figure 2-4 and shows its connections and functionality, the shield module, and the electrode select.

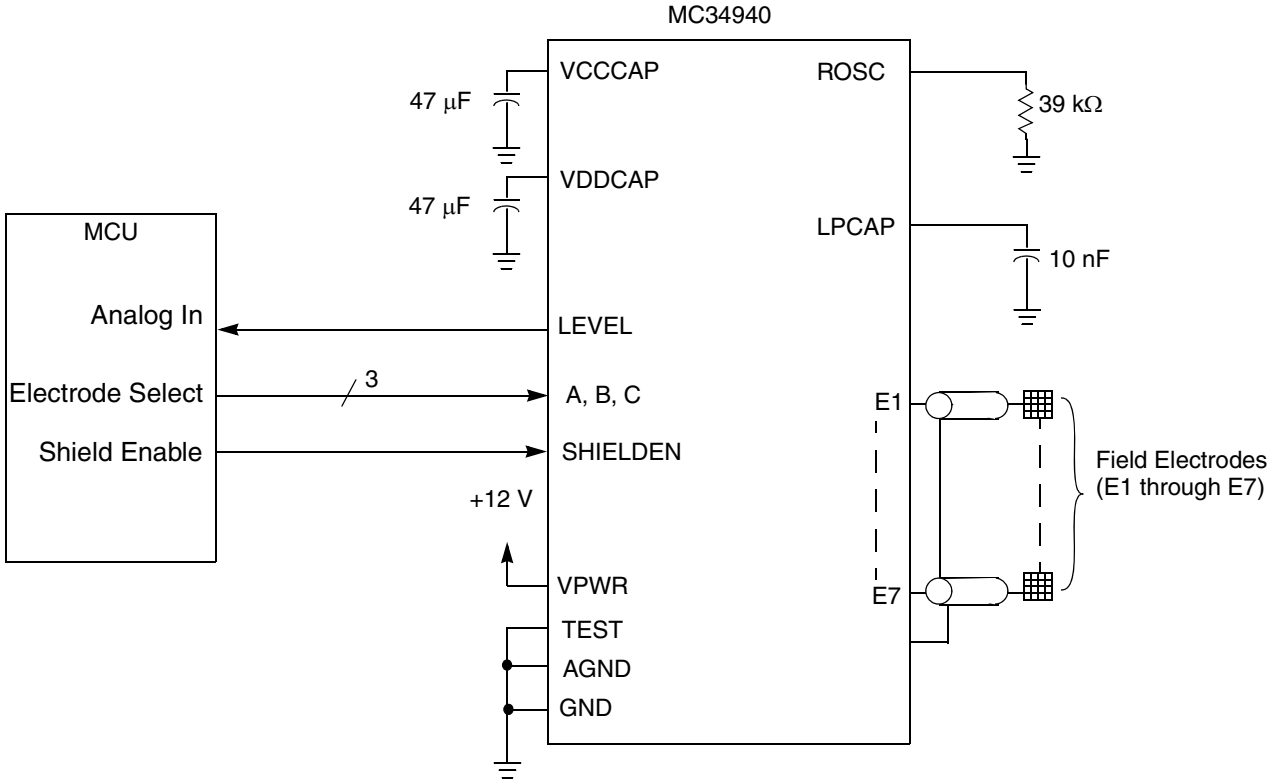


Figure 2-4. Electrical-Field Building Block

2.4 Board Layout

The detailed layout, schematic, and bill of materials (BOM) of the MC9RS08KA2 PADL reference design are shown in this section.

2.4.1 PADL Components Board

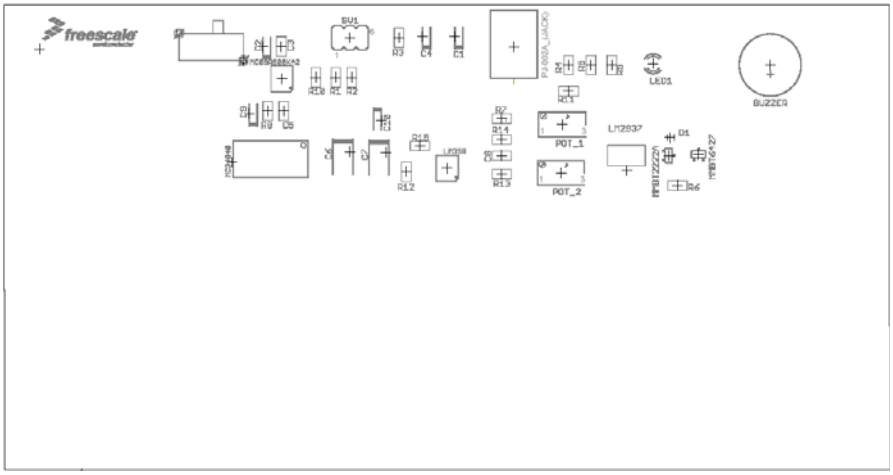


Figure 2-5. PADL Component Side

2.4.2 General Layout Explanation

The circuit board layout considerations are dominated by:

- Minimizing size
- The need to conduct high currents into the module, to the power drivers, and then out of the module

2.4.3 PCB Board Size

The circuit board size is 7.1 x 3.8 inches.

2.4.4 Component Placement

Before laying out the PCB, components were placed on the PCB. Figure 2-6 shows the separated main blocks. Low-level analog, digital circuitry, and high-current switches were separated to maintain signal integrity.

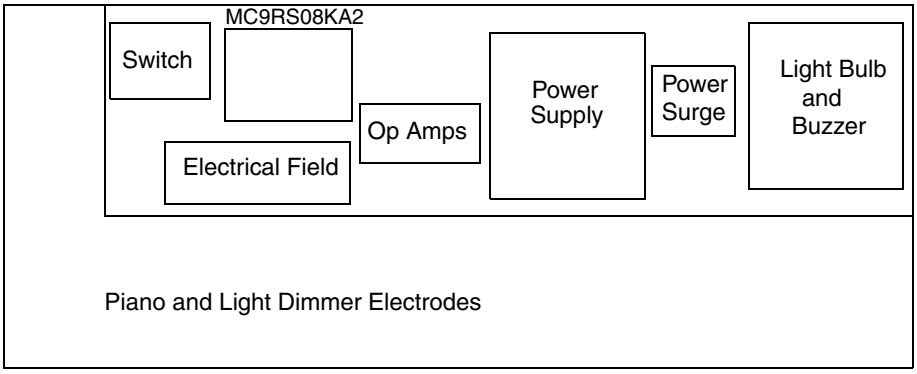


Figure 2-6. PADL Board and Main Blocks

- BDM Connector—Near the MC9RS08KA2 microcontroller.
- Power Stage Block—Digital devices are separated from the high-current management blocks.
- Electrical-Field Footprints—Electrodes must be near the electrical-field electrodes to avoid parasite capacitance.
- Via Dimensions and Spacing—Electrode spacing is 244 mil. The electrode width is 636 mil, and the electrode height is 1476 mil. The vias spacing is 35 mil.

2.4.5 Layout Layers

2.4.5.1 Top Layer

The low-analog and digital signals are separated from battery and high current traces to avoid interference between the signals; the signals from the piano and dimmer electrodes are near the electrical-field device.

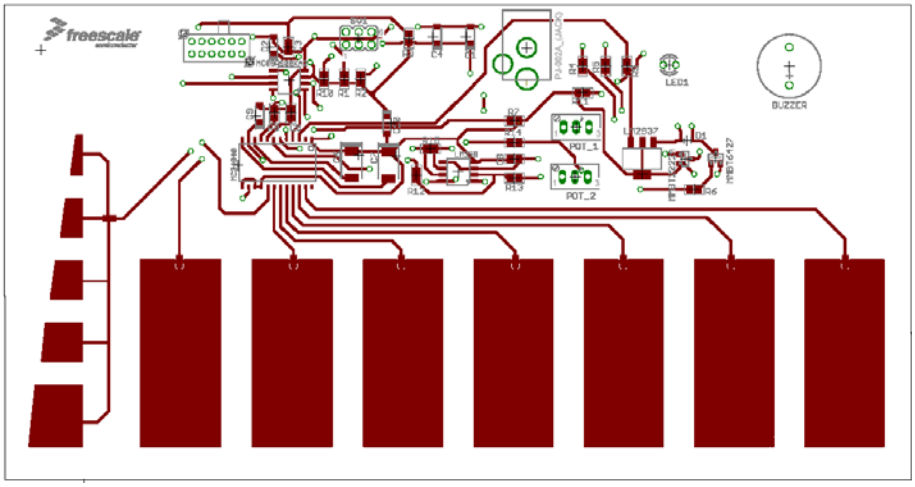


Figure 2-7. Top Layer

2.4.5.2 Bottom Layer

Similar to the top layer, the low-analog and digital signals are separated from battery and high current traces to avoid interference between the signals (Figure 2-8).

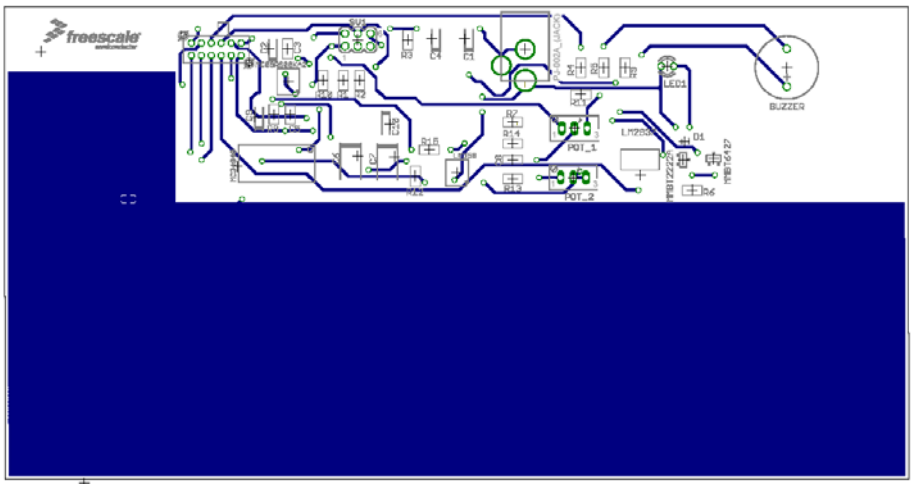


Figure 2-8. Bottom Layer

Chapter 3 Firmware Description

3.1 Introduction

This section describes MC9RS08KA2 PADL firmware.

3.1.1 Firmware Basics

This project was written using the CodeWarrior™ V 5.1 development tool. The application source file for this demo was written in assembly language.

3.1.2 Application Basics

The application demonstrates how to connect the MC9RS08KA2 to the MC34940.

3.2 Project Introduction

This section introduces and describes firmware implementation of the PADL project.

3.2.1 Coding Convention

All source code was written using guidelines to make the final product more readable.

The most important guidelines:

- Variables—Begin in uppercase to distinguish between words.
- Subroutines—In uppercase only at the beginning and are underscored to distinguish between words.
- Macros—Uppercase; underscores distinguish between words.
- Tags—In the first column of every row.

3.2.2 List of Project Files

These files are required for the project:

- Project files—PADL.mcp is the CodeWarrior Project file.
- Configuration files—MC9RS08KA2.inc contains definitions. Derivative.inc contains watchdog feed macro.
- Application source files—Main.asm contains the demonstration's main application.

3.2.3 MCU Peripherals

This section briefly describes the RS08 peripherals used in the project and summarizes the necessary microcontroller resources.

The MC9RS08KA2 MCU is used in this demonstration with the SOIC 8-pin package.

3.2.3.1 GPIO Module

If the demonstration is in piano mode, the GPIO module is configured as follows:

- Outputs—PTA1 (pin 7), PTA3 (pin 2), PTA4 (pin 6), and PTA5 (pin 5).
- Inputs—PTA0 (pin 8) and PTA2 (pin 1).

PTA3, PTA4, and PTA5 are the selectors of electrodes.

PTA1 is the PWM output (this signal goes to a buzzer create a sound).

PTA2 selects between function 1 and function 2.

PTA0 goes to the level signal; it indicates whether an electrode was pressed.

If the demonstration is in dimmer mode, the GPIO module is configured:

- Outputs—PTA1 (pin 7) and PTA4 (pin 4).
- Inputs—PTA0 (pin 8) and PTA2 (pin 1).

PTA1 is needed to discharge a capacitor. This is necessary to emulate an ADC.

PTA4 is the PWM output (this signal goes to a DC bulb).

PTA0 is an input for the level signal; it goes to the ACMP.

PTA2 selects between function 1 (piano) and function 2 (dimmer).

3.2.3.2 ACMP Module

The ACMP module is configured for only the dimmer light. When the ACMP— is greater than ACMP+, an interrupt generates. With this we can determine the ADC value.

ACMP+ is located in pin number 8 (PTA0).

ACMP– is located in pin number 7 (PTA1).

3.2.3.3 MTIM Module

The MTIM module emulates the ADC. The MTIM interrupts every 1 ms; this configuration works only for function 2. In function 1, the module interrupts every 40 counts.

3.2.4 Software Interrupts

See Table 3-1 for software interrupts used in the MC9RS08KA2 PADL.

Table 3-1. Interrupts

Module	Type of Interrupt	Purpose
ACMP	Asynchronous	Verify whether ACMP– is above ACMP+
MTIM	Asynchronous	Generate PWM

3.2.5 Main Variables of the Project

These variables control module functionality:

- Var
- Times
- SensorReading
- C1
- C2
- CountsSet
- CounterClr
- CounterL
- Counts
- PCBuffer

3.2.6 Memory Usage

Table 3-3 shows software memory usage of the MC9RS08KA2 PADL.

Table 3-2. Memory Usage

Memory Type	Total Size (Bytes)	Used Memory (Bytes)	Free Memory (%)
Flash	2048	491	Approx. 77%
RAM	63	11	Approx. 82%

3.3 Firmware Implementation

This section contains the complete description of the firmware included with the board.

3.3.1 Application

When the switch is in position 1 (logic level 0), the application works on the piano demonstration. This function selects one of the seven electrodes and knows which was pressed (Figure 3-1and Figure 3-2).

The application switches three pins (PTA3, PTA4, and PTA5) from 001 to 111 (see Figure 3-1, Figure 3-2, and Table 3-3); this signal is sent to the electrical field and then selects an electrode to read the level voltage. This is the main loop from this firmware section.

NOTE

The signals A, B, and C can be switched at least 2 ms after the last changes were made.

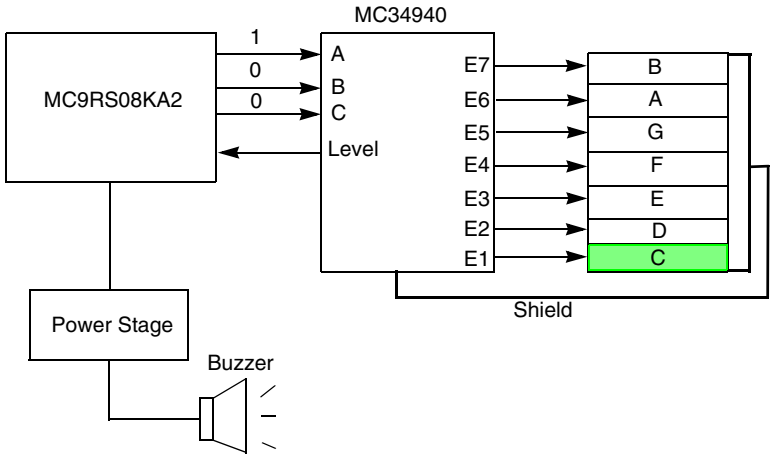


Figure 3-1. MC9RS08KA2 Piano and Dimming Switches Signals

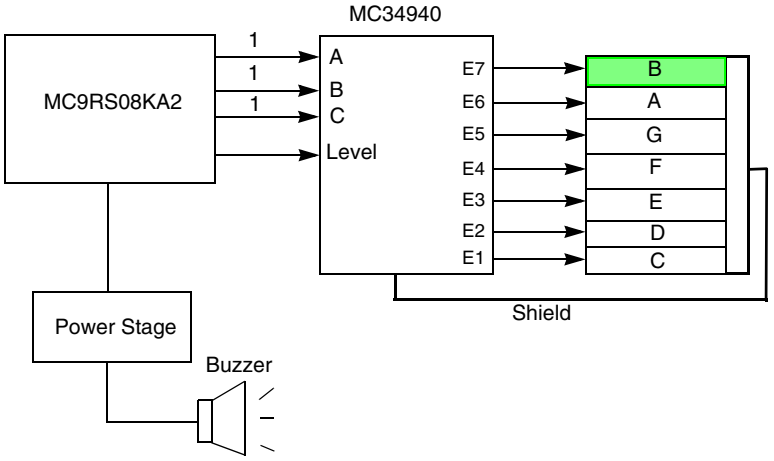


Figure 3-2. MC9RS08KA2 Piano and Dimming Switches Signals

Table 3-3. Signals Switch to Select a Pad

Microcontroller Signals			Electrical-Field Signals	
A	**B**	**C**	**Selected Pad**	
0	0	1	E1	C
0	1	0	E2	D
0	1	1	E3	E
1	0	0	E4	F
1	0	1	E5	G
1	1	0	E6	A
1	1	1	E7	B

After the electrode is selected, it is assigned the number of counts necessary to generate a musical note. The counts and the MTIM (configured with a prescaler of 1 and 40 counts before interrupts) generate the PWM. The PWM subroutine requires the pulse in the middle of the period (Figure 3-2). If the pulse is not generated at the middle of the period, the resulting frequency is incorrect for the required note.

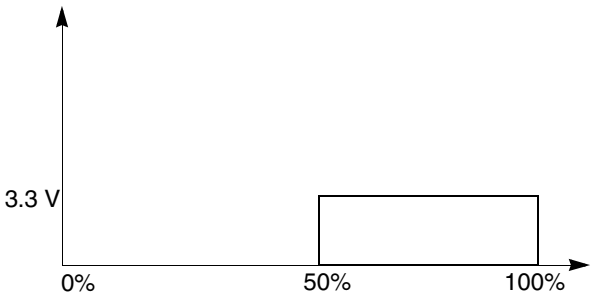


Figure 3-3. MC9RS08KA2 Piano and Dimming Light PWM

After the sound is generated, the application reads PTA2 and executes the function depending on the switch position.

If the switch changes to position 2 (logical level 1), the application works on the dimmer light. For this part of the firmware only one electrode is selected (signal A is active, signal C is set to GND). When the electrode is pressed, the ACMP detects the voltage change. Then a value is read from the ADC. This value is used to make a PWM and reveal the intensity light in a DC bulb. This application has only five light intensities.

For more information about the firmware, refer to Figure 3-3.

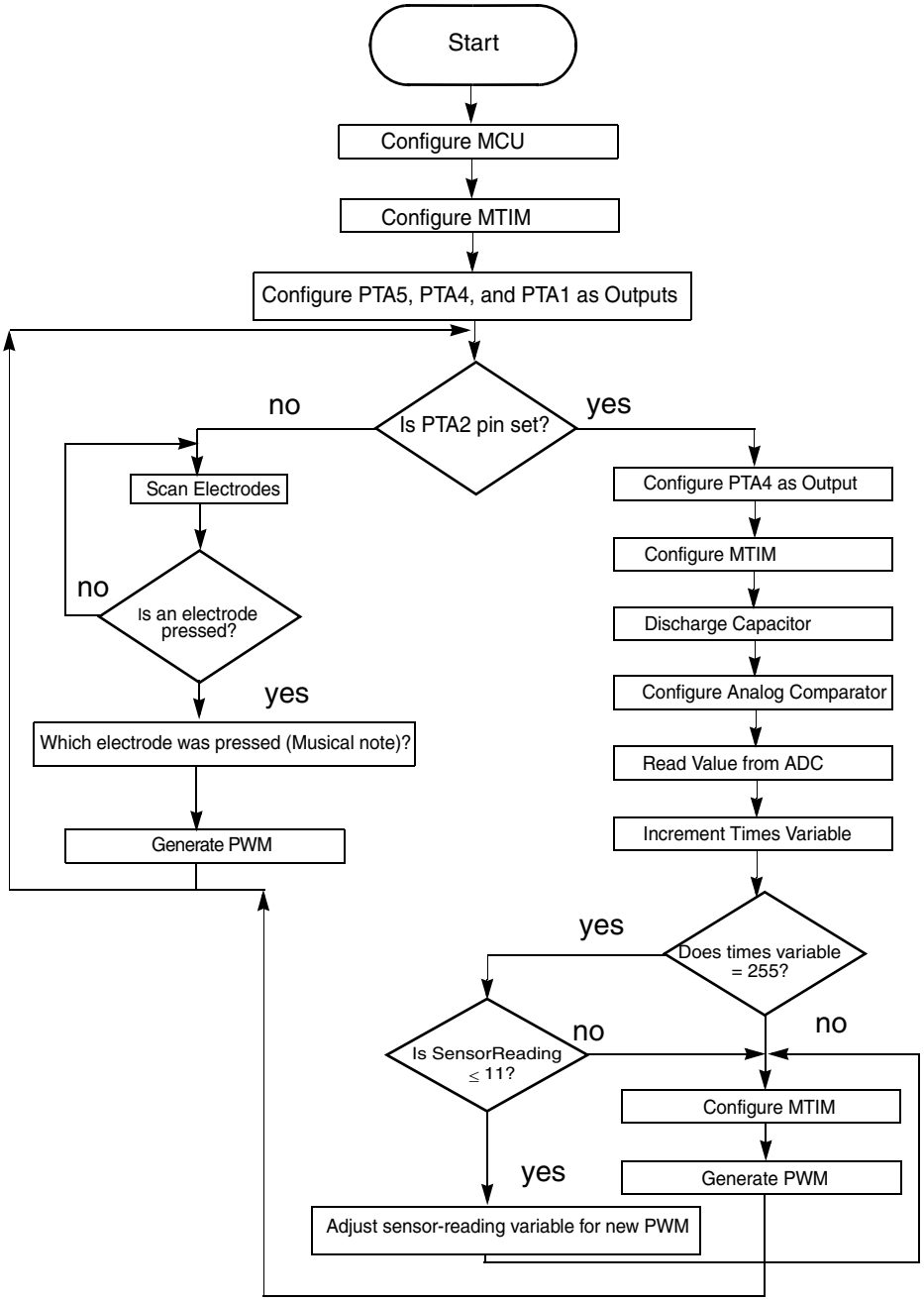


Figure 3-4. MC9RS08KA2 PADL Flowchart Diagram

Appendix A. BOM and Schematics

Table A-1. MC9RS08KA2 PADL Reference Design Bill of Materials

Item	QTY	Reference	Description	Manufacturer	Part Number
1	3	C3,C5,C8	0.01μ	KEMET	399-1158-1-ND
2	3	C1,C4,C9	0.1 μ	KEMET	399-3676-1-ND
3	2	C2,C10	10 μ	KEMET	399-3687-1-ND
4	2	C6,C7	47 μ	KEMET	7343-31(D-EIA)
5	1	R5	120 Ω	Rohm	RHM120CC-ND
6	3	R4,R6,R7	1 kΩ	Panasonic	P1.0KACT-ND
7	2	R10,R11	2.2 kΩ	Yageo	311-2.2KARCT-ND
8	1	R8	39 kΩ	Yageo	311-39.0KCRCT-ND
9	2	R2,R9	47 kΩ	Yageo	311-47KARCT-ND
10	6	R1,R3,R12,R13, R14,R15	10 kΩ	Yageo	311-10.0KCRCT-ND
11	2	NU	Pot 100 kΩ	Bourns	3266W-104-ND
12	1	NU	Lm358DR – Op Amp	3-M	296-1014-1-ND
13	1	NU	EG4208 – Switch	E-Switch	EG1914-ND
14	1	NU	MMBT6427 – NPN Darlington	Fairchild Semiconductor	MMBT6427-FSCT-ND
15	1	NU	Microcontroller	Freescale Semiconductor	MC9RS08KA2
16	1	NU	Electrical Field	Freescale Semiconductor	MC34940
17	1	NU	MMBT2222A – BJT	Fairchild	MMBT2222A-FDICT-ND
18	1	NU	B0520WS-7-F – Diode	Taiwan Semiconductor	B0520WS-FDICT-ND
19	1	NU	Buzzer	Mallory	458-1064-ND
20	1	NU	7219 – Bulb	Chicago Miniature	CM7219-ND
21	1	NU	LM2937IMP-3.3 – 3.3V Regulator	National Semiconductor	LM2937IMP-3.3CT-ND
22	1	NU	PJ-002A – DC Power Jack	CUI Inc	CP-002A-ND

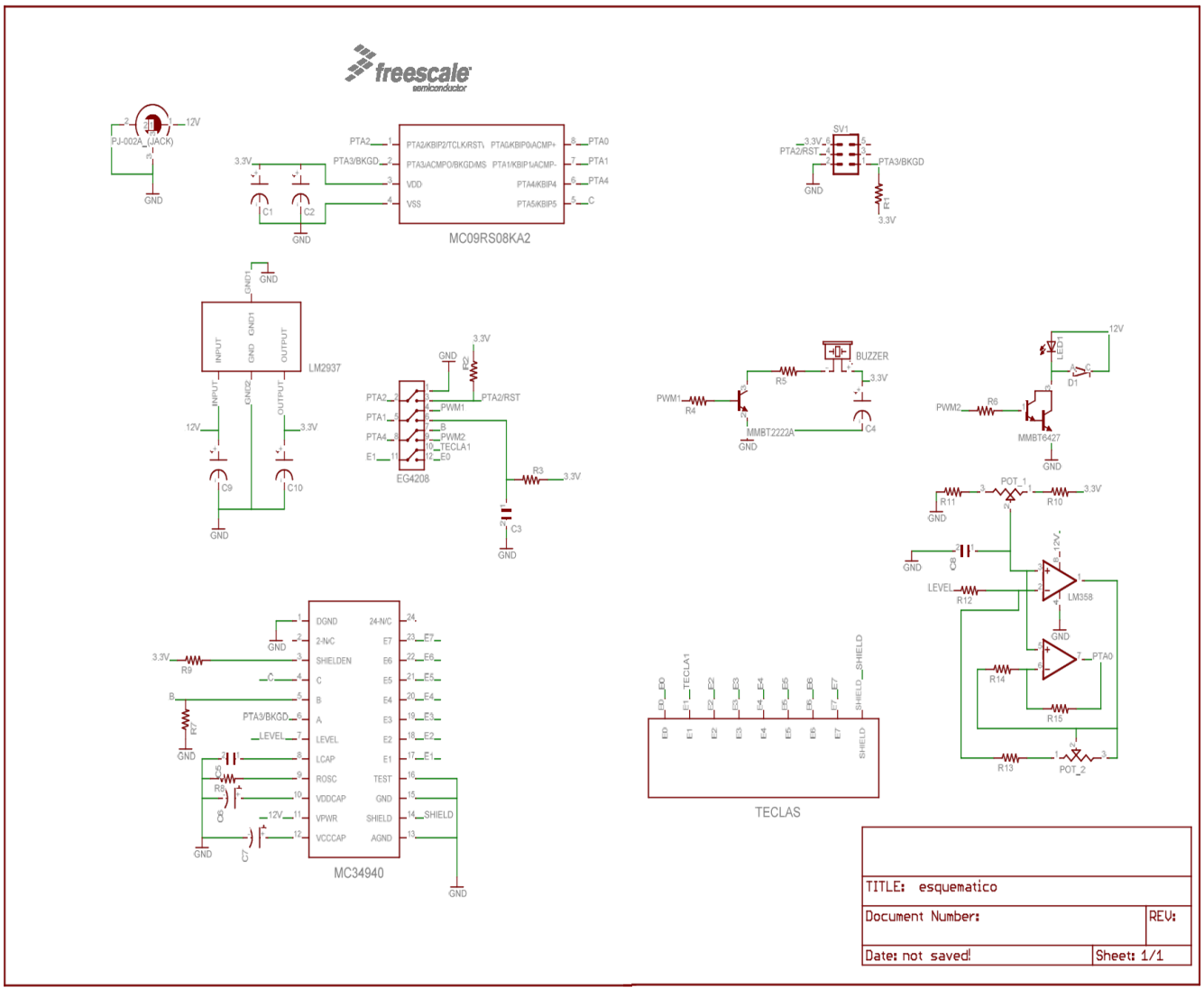


Figure A-1. PADL Block Diagram Schematic

Appendix B. Glossary

- ACMP – Analog comparator module. It provides a circuit to compare two analog inputs' voltage or one analog input's voltage to an internal reference.
- ADC – Analog to digital converter.
- Binary – Relating to the base 2 number system.
- Bit – A binary digit that has a value of logic 0 or logic 1.
- Byte – A set of 8 bits.
- DC – Direct current.
- EEPROM – Electrically erasable and programmable, read-only memory.
- Firmware – Instructions and data that control the operation of a microcontroller.
- MCU – Microcontroller unit, which is a complete computer system, including a CPU, memory, a clock oscillator, and Input/Output (I/O) on a single integrated circuit.
- PADL – Piano and dimming light.
- Port – A set of wires for communicating with off-chip devices.
- Prescaler – A circuit that generates an output signal related to the input signal by a fractional scale such as 1/2, 1/8, 1/10, etc.
- PWM – Pulse width modulation.
- Toggle – To change the state of an output from logic 0 to logic 1 or from logic 1 to logic 0.
- VBatt – Battery voltage.

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

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

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

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. The ARM POWERED logo is a registered trademark of ARM Limited. ARM7TDMI-S is a trademark of ARM Limited. Java and all other Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. The Bluetooth trademarks are owned by their proprietor and used by Freescale Semiconductor, Inc. under license.

© Freescale Semiconductor, Inc. 2004. All rights reserved.

DRM085
Rev. 0
10/2006

Freescale Semiconductor
Application Note

Document Number: AN3413
Rev. 0, 02/2007

Low-Cost Digital Timer

General-Purpose Digital Timer Using the MC9RS08KA2

by: Manuel Dávalos
8-Bit Microcontrollers Applications Engineer
RTAC Americas

1 Introduction

Many mechanical devices continue to function in our daily tasks. In this application note, you will see how a digital device (digital timer) could replace a mechanical device (mechanical timer) in a more precise, low-cost design using the MC9RS08KA.

Contents

© Freescale Semiconductor, Inc., 2007. All rights reserved.

freescale™ semiconductor

2 Description of Mechanical and Digital Timers

2.1 Mechanical Timers

Mechanical timers turn on devices at the end of a time period. They do not require electrical power and can be stored for a long period of time. There are some different types of mechanical timers such as clock timers, spring-driven timers, and dashpoint timers. Clock timers open or close a circuit based on the position of internal or external mechanisms. Spring-driven timers use a spring and trip lever to generate the mechanical action. Dashpoint timers pass compressed air or fluid into or out of a contained space through an opening with a fixed or variable diameter (smaller openings are used for longer time delays).

Timing ranges are measured in seconds or hours. Some mechanical timers are compact, rugged, or resistant to corrosion. Others provide varying degrees of resistance to environmental factors such as temperature, vibration, and shock of operation.

Most mechanical timers are used in applications where checking the completion of an operation causes the start of another process. Common applications include automatic presses, refrigerators, and industrial washing machines. Figure 1 shows some typical mechanical timer's gears. Figure 2 shows a simple gear system of a mechanical timer.

Figure 1. Mechanical Timers' Gears

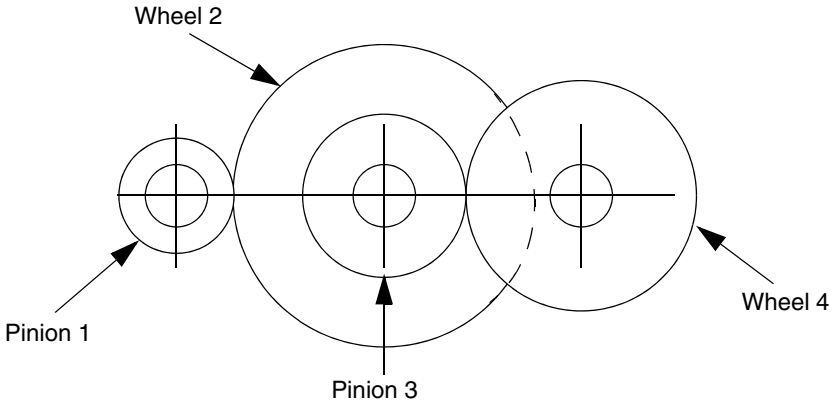


Figure 2. Gear System of a Mechanical Timer

2.2 Digital Timers

When using an L-CDT, press a pushbutton and one light-emitting diode (LED) turns on; then, tune a potentiometer to configure a hold time. The hold time could be in hours or minutes and depends on the software configuration changes. The next time you press the pushbutton, a second LED turns on. After that, tune the potentiometer again to configure another hold time. Finally, press the pushbutton one more time and a third LED turns on to indicate the module is working (two hours off and three hours on) within a finite loop that finishes when the pushbutton is pressed again to configure another on/off time. Figure 3 shows the basic hardware configuration for this application note.

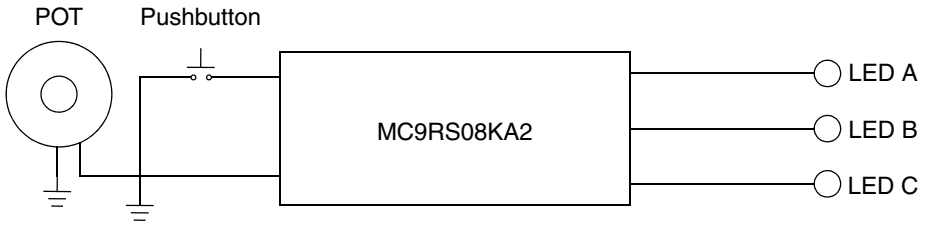


Figure 3. Digital Timer Hardware Configuration

3 Design Requirements

The digital timer's device must have:

- One MC9RS08KA2 MCU
- Three LEDs
  - LED A indicates the device is waiting for the new configuration of time the relay must remain off before activating the device
  - LED B indicates the device is waiting for the configuration of time the relay must remain on
  - LED C indicates the module is programmed and activated. (disables when times are being programmed)
- 1 M Ω potentiometer — When turning the potentiometer right or left, you can select the times
- Pushbutton — To start a new cycle within the states machine

4 Solution

The L-CDT solution works with the MC9RS08KA2 MCU in stop mode, waiting for a keyboard interrupt (KBI) to occur to go to one of the four states of the state machine. When a KBI is pending and the state machine begins, the MCU wakes up, turns on the LED A, and enters into stop mode again to wait for the first configuration (the delay time before actuating the device in hours/minutes). When the pushbutton is pressed again, the MCU wakes up, saves the value coming from the analog-to-digital converter (ADC), turns on the LED B, and enters into stop mode to wait for the last configuration (the delay time the relay must remain on in hours/minutes). After the MCU receives another KBI, it goes to the next state that saves the value coming from the ADC, turns on the LED C to indicate the L-CDT is working, and enters stop mode again. However, while the MCU is in stop mode, the output values from the MC9RS08KA2 port

remains. If you press the pushbutton again in this state, the MCU turns off all the outputs and enters stop mode. The MCU then waits for another KBI to start the state machine.

NOTE

The MCU does not have an ADC module. The ADC was designed with a resistor capacitor (RC) circuit. The capacitor is charging and discharging by software to obtain a digital value using the modulo timer (MTIM) and the analog comparator (ACMP) modules.

4.1 Benefits of a Digital Timer

The benefits of using a digital timer over a mechanical timer are:

- Digital timer costs less than mechanical timers
- The digital timing precision is better than in the mechanical timers
- Friction is not present in digital timers
- Ease of use
- Digital timers are reconfigurable to the user's convenience

5 Tutorial

The program starts in the _Startup vector, where in the first code lines, the program calls three different subroutines for configuring the MCU. The first subroutine (Init_Conf) disables the computer operating properly (COP), enables the stop mode, disables or enables the background (BKGD) mode (depending on the MODE value), and configures the MCU to run with its bus clock frequency at 8 MHz trimmed. The second subroutine (Init_PTA) configures the general-purpose input/output (GPIO) port and the pull-up/pull-down internal resistors. Finally, the third subroutine (Init_KBI) configures PTA2 as the keyboard interrupt (KBI).

```
;*****************************************************************************
;*                              Main Program                                 *
;*****************************************************************************
;
_Startup:
      jsr Init_Conf
      jsr Init_PTA
      jsr Init_KBI
```

The next code lines are where the MCU enters into stop mode and waits for a KBI interrupt. After the MCU receives a KBI interrupt, the LED A in the demo board turns on and the MCU is waits for the configuration time the device must remain off. When the MCU receives the second KBI interrupt, it turns on the LED B and calls the ADC_RECEIVE subroutine to obtain a digital value of eight bits of range. There is a delay subroutine called before entering into stop mode and before reading an ADC value because of the debounce time. After that, the MCU calls one of two subroutines (in this case, the TIME_HOURS subroutine) to divide the ADC range value by 12 different values. If you call the TIME_HOURS subroutine, time increments by one hour (1, 2, 3, 4,...,12). If you call the TIME_MINUTES subroutine, time increments by five minutes (5, 10, 15, 20,…, 60 minutes).

```
main:
      mov #(mKBISC_KBACK | mKBISC_KBIE),KBISC
      clr PTAD
      jsr delay
      stop
      bset 3,PTAD
      mov #(mKBISC_KBACK | mKBISC_KBIE),KBISC

      jsr delay
      stop
      bset 4,PTAD
      mov #(mKBISC_KBACK | mKBISC_KBIE),KBISC
      jsr delay
      jsr ADC_RECEIVE
      jsr TIME_HOURS
      mov timec,temp
```

NOTE

There is a variable called timec that must be stored into flags called temp and temp2. The first time one of the TIME_HOURS and TIME_MINUTES subroutines are used, timec must be stored into the temp flag. The second time you use one of these subroutines, timec must be stored into the temp2 flag for future loops functions.

The MCU enters into stop mode again to wait for the next configuration time and another KBI interrupt. When this KBI interrupt arrives, the MCU turns on the LED C and retrieves the ADC value from the potentiometer (POT). Store the timec value in the temp2 flag because it is the second time a TIME_HOURS or TIME_MINUTES subroutine is called. After this action is taken, the MCU is ready to configure the modulo timer module (MTIM) to retrieve time interrupts with counts to achieve one second.

```
      stop
      bset 5,PTAD
      mov #(mKBISC_KBACK | mKBISC_KBIE),KBISC
      jsr delay
      jsr ADC_RECEIVE
      jsr TIME_HOURS
      mov timec,temp2
      MTIM1SConfig
```

Finally, the MCU loads the temp flag into the timec variable to enter a finite loop that finishes when the timec variable is 0 (time the actuator is off). Then, the MCU loads the temp2 flag into the timec variable to enter another finite loop that finishes when the timec variable is 0 again (time the actuator is on). To show the actuator is working, the actuator’s outport pin from the demo board is the same as the LED A (PTA3).

NOTE

You do not see the actuator work in debug mode. The MCU must be working in run mode to show the actuator changes (MODE EQU 1).

```
loop:
      mov temp,timec
      bclr 3,PTAD
loop1:
      jsr HOUR
      lda temp
      cbeqa #$01,main
      dbnz timec,loop1

      mov temp2,timec
      bset 3,PTAD
loop2:
      jsr HOUR
      dbnz timec,loop2
      bra loop
```

NOTE

A KBI interrupt could also break these loops. When this happens, it is possible to configure another on/off time. If using the TIME_HOURS or TIME_MINUTES subroutines, the corresponding HOUR or MINUTE subroutines must be used too.

Figure 4 shows the basic code functions of the application note in a flow chart.

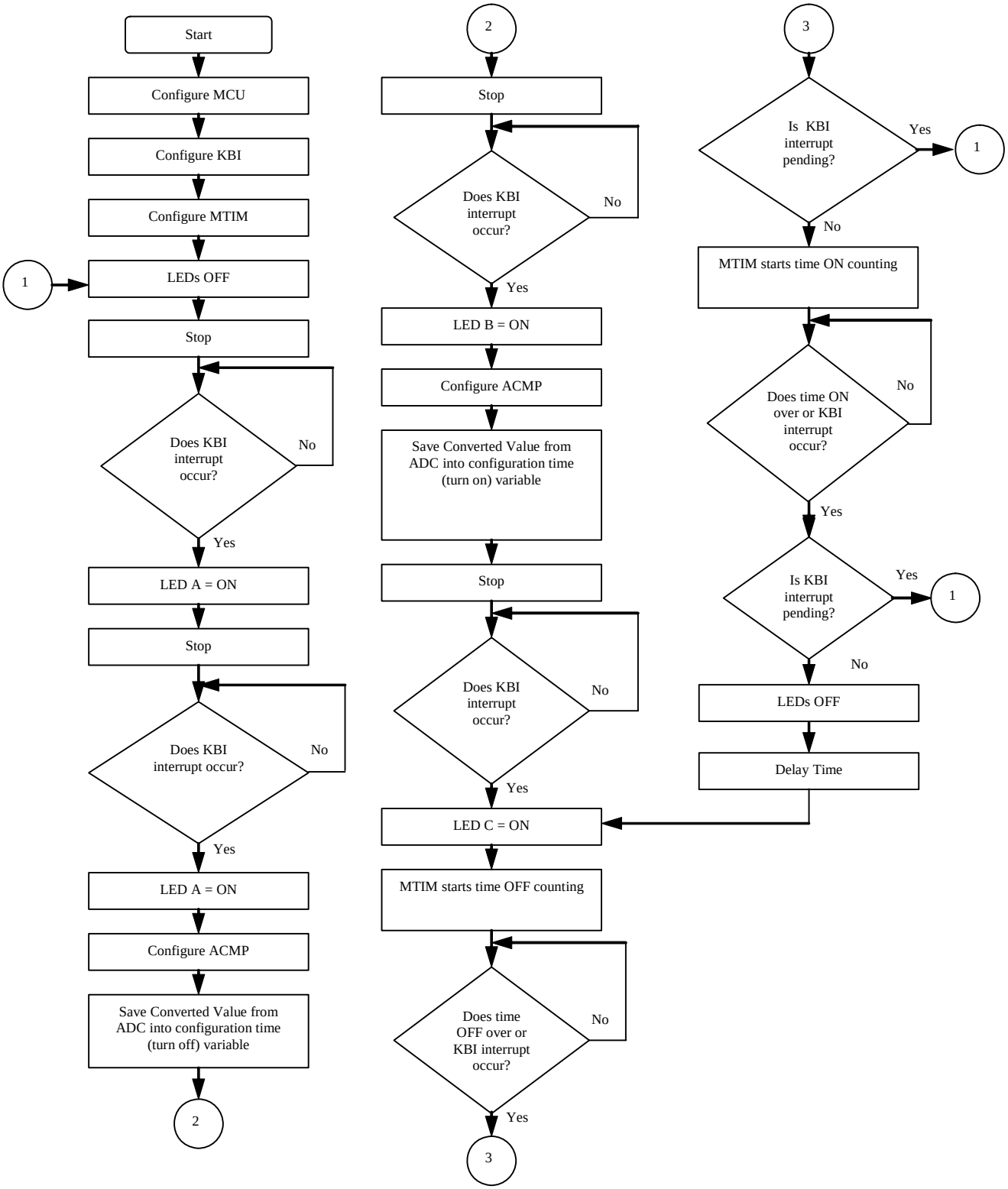


Figure 4. Basic Code Functions Flow Chart

NOTE

When LED C turns on, LED A turns on and off to indicate when the device is on and off.

6 Instructions

To achieve the solution, do this:

1. Open CodeWarrior version 5.1
2. Open the ETimerR.mcp file from the ETimerR folder
3. Change the MCU connections for SofTec RS08
4. Press the F7 function key for making the project
5. Press the F5 function key to download the file into the flash memory
6. In the MCU configuration, select the DEMO9RS08KA2 and press the OK button
7. The CodeWarrior 5.1 compiler downloads the file into flash

7 References

Freescale MC9RS08KA2 Data Sheet available from Freescale.com

About Mechanical Timers

Appendix A Firmware

```
;*********************************************************************
;*                           DISCLAIMER                              *
;* Services performed by FREESCALE in this matter are performed      *
;* AS IS and without any warranty. CUSTOMER retains the final decision   *
;* relative to the total design and functionality of the end product.    *
;* FREESCALE neither guarantees nor will be held liable by CUSTOMER      *
;* for the success of this project. FREESCALE disclaims all warranties,  *
;* express, implied or statutory including, but not limited to,          *
;* implied warranty of merchantability or fitness for a particular       *
;* purpose on any hardware, software ore advise supplied to the project  *
;* by FREESCALE, and or any product resulting from FREESCALE services.   *
;* In no event shall FREESCALE be liable for incidental or consequential *
;* damages arising out of this agreement. CUSTOMER agrees to hold        *
;* FREESCALE harmless against any and all claims demands or actions      *
;* by anyone on account of any damage, or injury, whether commercial,    *
;* contractual, or tortuous, rising directly or indirectly as a result   *
;* of the advise or assistance supplied CUSTOMER in connection with      *
;* product, services or goods supplied under this Agreement.             *
;*********************************************************************
;*****************************************************************
;* This stationery serves as the framework for a user application. *
;* For a more comprehensive program that demonstrates the more     *
;* advanced functionality of this processor, please see the        *
;* demonstration applications, located in the examples             *
;* subdirectory of the "Freescale CodeWarrior for HC08" program    *
;* directory.                                                      *
;*****************************************************************

; Include derivative-specific definitions
            INCLUDE 'derivative.inc'

; export symbols

            XDEF _Startup
            ABSENTRY _Startup

; variable/data section


      ORG    RAMStart

;*************************************************************************
;*                           ADC Constant                                *
;*************************************************************************
Table_Data EQU $3E00

;*************************************************************************
;*                           ADC definitions                             *
;*************************************************************************
ACMP_ENABLE          SET   $92
ACMP_DISABLED        SET   $20
MTIM_ENABLE          SET   $40
MTIM_STOP_RESET      SET   $30
MTIM_128_DIV         SET   $07
FREE_RUN             SET   $00
```

```
;*****************************************************************************
;*                            Time definitions                                *
;*****************************************************************************
;
RTIMES               SET   $F5
RTIMEM               SET   $3C
RTIMEH               SET   $3C


MAXlevel           EQU   1
MODE               EQU   1         ; Operation Mode (1:Run Mode, 0:Background Mode)
;*****************************************************************************
;*                             ADC Variables                                 *
;*****************************************************************************
;
SensorReading        DS.B  1         ; Store ACMP read value
ConvertedValue       DS.B  1         ; This varible store converted value
Temp_Page            DS.B  1         ; Temporal backup Page

pcBuffer             DS.B  MAXlevel

;*****************************************************************************
;*                               Variables                                   *
;*****************************************************************************
;
time                 DS.B  1
timec                DS.B  1
seconds              DS.B  1
minutes              DS.B  1
hours                DS.B  1
temp                 DS.B  1
temp2                DS.B  1
counter              DS.B  1

      ; Insert your data definition here


; code section

            ORG    ROMStart

;*****************************************************************************
;*                            MACRO declarations                             *
;*****************************************************************************
;
TRIM_ICS: MACRO                      ; Macro used to configure the ICS with TRIM
      mov   #$FF,PAGESEL             ; change to last page
      ldx   #$FA                     ; load the content which TRIM value is store
      lda   ,x                       ; read D[X]
      sta   ICSTRM                   ; Store TRIM value into ICSTRM register
      mov   #$00,ICSSC               ; Fine TRIM
      ENDM

ENTRY_SUB: MACRO              ;Macro for "stacking" SPC
             sha
             sta pcBuffer + 2*(\1)
             sha
             sla
             sta pcBuffer + 2*(\1) +1
             sla
```

```
      ENDM

      NOP                          ;needs to separate MACROS

EXIT_SUB: MACRO        ;Macro for restore SPC
            sha
            lda pcBuffer + 2*(\1)
            sha
            sla
            lda pcBuffer + 2*(\1) +1
            sla
      ENDM

MTIM1MS: MACRO
      mov #$70,MTIMSC
      mov #$7D,MTIMMOD
      mov #$06,MTIMCLK
      ENDM

MTIM1SConfig: MACRO
      mov #$70,MTIMSC
      mov #$FE,MTIMMOD
      mov #$07,MTIMCLK
      bclr 4,MTIMSC
      ENDM

MTIM1S: MACRO
      mov #RTIMES,seconds
MTIMIsr:
      brclr 7,MTIMSC,MTIMIsr
      bset 5,MTIMSC
      dbnz seconds,MTIMIsr
      brset 3,KBISC,reset
      clr temp
      bra e_
reset:
      mov #$01,temp
e_
      ENDM

;*****************************************************************************
;*                            Init Microcontroller                           *
;*****************************************************************************
Init_Conf:
      IFNE  MODE
      mov #HIGH_6_13(SOPT), PAGESEL
     mov #$20, MAP_ADDR_6(SOPT)       ; Disables COP and enables Stop
      ELSE
      mov #HIGH_6_13(SOPT), PAGESEL
    mov #$22, MAP_ADDR_6(SOPT)    ; Disables COP, enables BKGD (PTA3) and Stop

      ENDIF
      clr  ICSC1                     ; FLL is selected as Bus Clock
      TRIM_ICS                       ; Trim MCU to work at 8MHz
      clr  ICSC2
      clr  temp
      rts
```

```
;*****************************************************************************
;*                                 Init PTA                                  *
;*****************************************************************************
Init_PTA:
      mov #HIGH_6_13(PTAPE),PAGESEL
      mov #$FE, MAP_ADDR_6(PTAPE); Enables internal Pulling device

      mov #HIGH_6_13(PTAPUD),PAGESEL
      clr MAP_ADDR_6(PTAPUD)         ; Configures Internal pull up device in PTA

      mov #$FA,PTADD
      clr PTAD
      rts


;*****************************************************************************
;*                                 Init KBI                                  *
;*****************************************************************************
Init_KBI:
      mov #(mKBIPE_KBIPE2),KBIPE
      mov #(mKBISC_KBIE | mKBISC_KBACK),KBISC
      rts

;*****************************************************************************
;*                            ADC Receive Function                           *
;*****************************************************************************
ADC_RECEIVE:
      bra MTIM_ADC_Init                ; Configure MITM
next:
      bra Discharge_Cap                ; Discharge Capacitor
next2:
      bra ACMP_Conf                    ; Configure ACMP+ and ACMP-
next3:
      mov #MTIM_ENABLE,MTIMSC          ; Timer Counter Enabled
      wait                             ; Wait for ACMP interrupt
      bset 1,MTIMSC
      lda MTIMCNT                      ; read counter timer value
      sta SensorReading                ; store counter value
      mov #HIGH_6_13(SIP1), PAGESEL
      brset 3, MAP_ADDR_6(SIP1),ReadVal ; branch if ACMP interrupt arrives
      bra next
ReadVal:
      MOV #MTIM_STOP_RESET,MTIMSC      ; Stop and reset counter
      MOV #ACMP_DISABLED, ACMPSC
LookupTable:
      lda SensorReading
      rola                             ; Getting 2 MSB
      rola
      rola
      and #$03
      add #(Table_Data>>6)             ; Page Calculating
      mov #PAGESEL,Temp_Page           ; Backup actual page
      sta PAGESEL                      ; Page Change
      lda SensorReading
      and #$3F                         ; Extract 6 LSB
      add #$C0                         ; Index to paging window
```

```
      tax
      lda ,x                          ; Load table result
      sta ConvertedValue              ; Store result
      mov #Temp_Page, PAGESEL         ; Back Page
      rts
MTIM_ADC_Init:
      mov #MTIM_128_DIV,MTIMCLK
      mov #FREE_RUN,MTIMMOD           ; Configure Timer as free running
      mov #MTIM_STOP_RESET,MTIMSC
      bra next
Discharge_Cap:
      bset 1,PTADD                    ; Configure PTA1 as Output
      bclr 1,PTAD                     ; Start Capacitor discharging
      lda  #$FE                       ; Set delay time
waste_time:
      dbnza waste_time                ; wait until Delay = 0
      bra next2
ACMP_Conf:
      MOV #ACMP_ENABLE,ACMPSC
      bra next3
      rts


;*****************************************************************************
;*                            Times Functions                                *
;*****************************************************************************
MINUTE:
      mov #RTIMEM,minutes
TimeIsrM:
      lda temp
      cbeqa #$01,_en
      MTIM1S
      dbnz minutes,TimeIsrM
_en
      rts

HOUR:
      mov #RTIMEH,hours
TimeIsrH:
      ENTRY_SUB 0
      jsr MINUTE
      EXIT_SUB 0
      lda temp
      cbeqa #$01,en_
      dbnz hours,TimeIsrH
en_
      rts

;*****************************************************************************
;*                              TIME_HOURS                                   *
;*****************************************************************************
TIME_HOURS:
      lda ConvertedValue
      sub #241
      bhs One
      lda ConvertedValue
      sub #220
```

```
      bhs Two
      lda ConvertedValue
      sub #199
      bhs Three
      lda ConvertedValue
      sub #178
      bhs Four
      lda ConvertedValue
      sub #157
      bhs Five
      lda ConvertedValue
      sub #136
      bhs Six
      lda ConvertedValue
      sub #115
      bhs Seven
      lda ConvertedValue
      sub #94
      bhs Eight
      lda ConvertedValue
      sub #73
      bhs Nine
      lda ConvertedValue
      sub #52
      bhs Ten
      lda ConvertedValue
      sub #31
      bhs Eleven
      mov #12,timec
      bra _End
One:
      mov #1,timec
      bra _End
Two:
      mov #2,timec
      bra _End
Three:
      mov #3,timec
      bra _End
Four:
      mov #4,timec
      bra _End
Five:
      mov #5,timec
      bra _End
Six:
      mov #6,timec
      bra _End
Seven:
      mov #7,timec
      bra _End
Eight:
      mov #8,timec
      bra _End
Nine:
      mov #9,timec
      bra _End
```

```
Ten:
      mov #10,timec
      bra _End
Eleven:
      mov #11,timec
_End
      rts

;*****************************************************************************
;*                              TIME_MINUTES                                 *
;*****************************************************************************
TIME_MINUTES:
      lda ConvertedValue
      sub #241
      bhs One_
      lda ConvertedValue
      sub #220
      bhs Two_
      lda ConvertedValue
      sub #199
      bhs Three_
      lda ConvertedValue
      sub #178
      bhs Four_
      lda ConvertedValue
      sub #157
      bhs Five_
      lda ConvertedValue
      sub #136
      bhs Six_
      lda ConvertedValue
      sub #115
      bhs Seven_
      lda ConvertedValue
      sub #94
      bhs Eight_
      lda ConvertedValue
      sub #73
      bhs Nine_
      lda ConvertedValue
      sub #52
      bhs Ten_
      lda ConvertedValue
      sub #31
      bhs Eleven_
      mov #60,timec
      bra End_
One_:
      mov #5,timec
      bra End_
Two_:
      mov #10,timec
      bra End_
Three_:
      mov #15,timec
      bra End_
Four_:
```

```
      mov #20,timec
      bra End_
Five_:
      mov #25,timec
      bra End_
Six_:
      mov #30,timec
      bra End_
Seven_:
      mov #35,timec
      bra End_
Eight_:
      mov #40,timec
      bra End_
Nine_:
      mov #45,timec
      bra End_
Ten_:
      mov #50,timec
      bra End_
Eleven_:
      mov #55,timec
End_
      rts

;*****************************************************************************
;*                            Main Program                                   *
;*****************************************************************************
_Startup:
      jsr Init_Conf
      jsr Init_PTA
      jsr Init_KBI
main:
      mov #(mKBISC_KBACK | mKBISC_KBIE),KBISC
      clr PTAD
      jsr delay
      stop
      bset 3,PTAD
      mov #(mKBISC_KBACK | mKBISC_KBIE),KBISC

      jsr delay
      stop
      bset 4,PTAD
      mov #(mKBISC_KBACK | mKBISC_KBIE),KBISC
      jsr delay
      jsr ADC_RECEIVE
      jsr TIME_HOURS
      mov timec,temp
      stop
      bset 5,PTAD
      mov #(mKBISC_KBACK | mKBISC_KBIE),KBISC
      jsr delay
      jsr ADC_RECEIVE
      jsr TIME_HOURS
      mov timec,temp2
      MTIM1SConfig
```

```
loop:
      mov temp,timec
      bclr 3,PTAD
loop1:
      jsr HOUR
      lda temp
      cbeqa #$01,main
      dbnz timec,loop1

      mov temp2,timec
      bset 3,PTAD
loop2:
      jsr HOUR
      dbnz timec,loop2
      bra loop

delay:
      mov #24,counter
      MTIM1MS
      bclr 4,MTIMSC

MTIMIsr1ms:
      brclr 7,MTIMSC,MTIMIsr1ms
      bset 5,MTIMSC
      dbnz counter,MTIMIsr1ms
      bset 4,MTIMSC
      rts

;*****************************************************************************
;*                            Startup Vector                                 *
;*****************************************************************************

      ORG   $3FFD
      JMP _Startup          ; Reset

;*****************************************************************************
;*                            Data Table                                     *
;*****************************************************************************
 ORG Table_Data
 dc.b 0,5,10,14,19,23,28,32,36,40,44,48,52,56,60,63
 dc.b 67,71,74,78,81,84,87,91,94,97,100,103,106,108,111,114
 dc.b 117,119,122,124,127,129,132,134,136,139,141,143,145,147,149,151
 dc.b 153,155,157,159,161,162,164,166,168,169,171,173,174,176,177,179
 dc.b 180,182,183,184,186,187,188,190,191,192,193,194,196,197,198,199
 dc.b 200,201,202,203,204,205,206,207,208,209,210,211,211,212,213,214
 dc.b 215,215,216,217,218,218,219,220,221,221,222,222,223,224,224,225
 dc.b 226,226,227,227,228,228,229,229,230,230,231,231,232,232,233,233
 dc.b 234,234,234,235,235,236,236,236,237,237,237,238,238,238,239,239
 dc.b 239,240,240,240,241,241,241,241,242,242,242,243,243,243,243,244
 dc.b 244,244,244,244,245,245,245,245,245,246,246,246,246,246,247,247
 dc.b 247,247,247,247,248,248,248,248,248,248,249,249,249,249,249,249
 dc.b 249,249,250,250,250,250,250,250,250,250,250,251,251,251,251,251
 dc.b 251,251,251,251,251,252,252,252,252,252,252,252,252,252,252,252
 dc.b 252,252,253,253,253,253,253,253,253,253,253,253,253,253,253,253
 dc.b 253,253,253,253,254,254,254,254,254,254,254,254,254,254,254,254
```

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 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: AN3413
Rev. 0
02/2007

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.

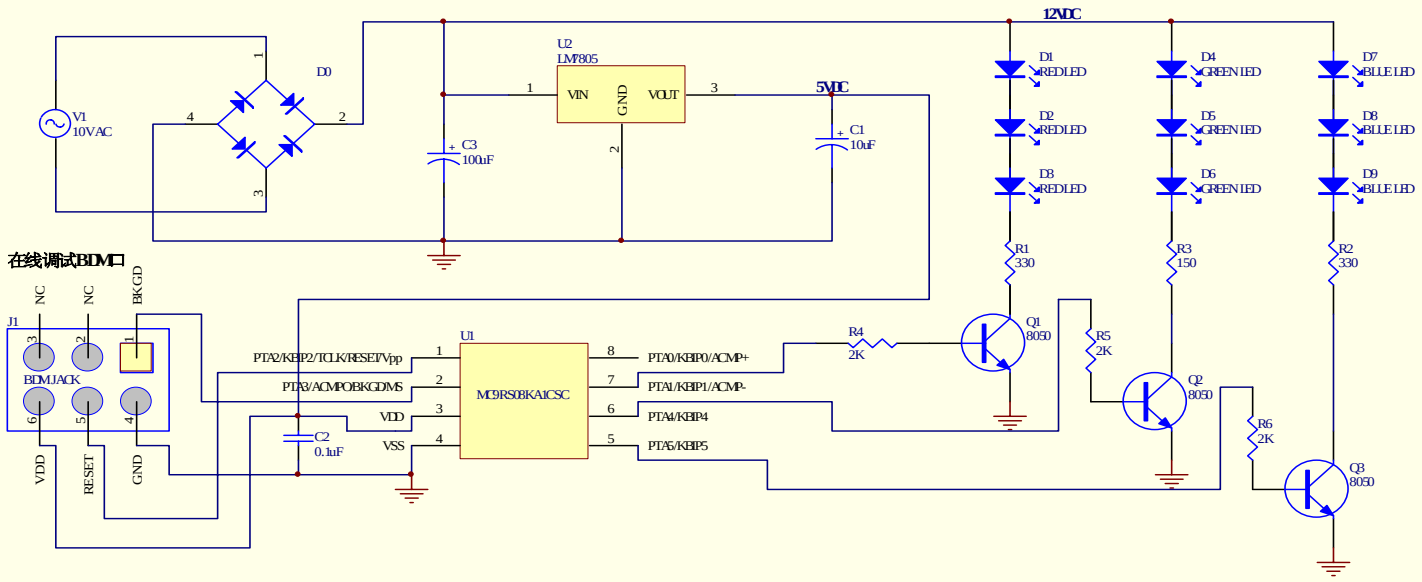
Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability 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.

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

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

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. 2007. All rights reserved.

freescale™
semiconductor



MC9RS08KA1在地埋灯上的应用

By.Johnsonhan 2007-8-31

Title		
Size: A4	Number	Revision
Date: 31-Aug-2007		Sheet of
File: E:\客户案例\地埋灯\地埋灯.Ddb		Drawn By:

软件参考代码：

```
;*******************************************************************
;* This stationery serves as the framework for a user application. *
;* For a more comprehensive program that demonstrates the more     *
;* advanced functionality of this processor, please see the         *
;* demonstration applications, located in the examples              *
;* subdirectory of the "Freescale CodeWarrior for HC08" program     *
;* directory.                                                       *
;*******************************************************************

; export symbols
                XDEF _Startup, main
               ; we export both '_Startup' and 'main' as symbols. Either can
               ; be referenced in the linker .prm file or from C/C++ later on

; Include derivative-specific definitions
               INCLUDE 'derivative.inc'
;===================================================================
====
; ICS Definition
;===================================================================
====
ICS_DIV_1          equ  $00
ICS_DIV_2          equ  $40
ICS_DIV_4          equ  $80
ICS_DIV_8          equ  $c0

;===================================================================
====
; MTIM Definition
;===================================================================
====
MTIM_DIV_1         equ   $00
MTIM_DIV_2         equ   $01
MTIM_DIV_4         equ   $02
MTIM_DIV_8         equ   $03
MTIM_DIV_16        equ   $04
MTIM_DIV_32        equ   $05
MTIM_DIV_64        equ   $06
MTIM_DIV_128       equ   $07
```

```
MTIM_DIV_256            equ     $08

MTIM_BUS_CLK            equ   $00
MTIM_XCLK               equ   $10
MTIM_TCLK_FALLING       equ  $20
MTIM_TCLK_RISING        equ   $30


;application definition
RED       equ     PTAD_PTAD1
mRED      EQU     mPTAD_PTAD1
BLUE      equ     PTAD_PTAD4
mBLUE     EQU     mPTAD_PTAD4
GREEN     EQU     PTAD_PTAD0
mGREEN EQU     mPTAD_PTAD0

;=============================================================================
====
; Application Macro
;=============================================================================
====
Delay_1_cycle: macro
        nop                                                     ;1 cycles
                endm
Delay_2_cycles: macro
        tsta                                                    ;2 cycles
                endm
Delay_3_cycles: macro
        brn       *                                             ;3 cycles
                endm
Delay_4_cycles: macro
        brn       *                                             ;3 cycles
        nop                                                     ;1 cycles
                endm
Delay_5_cycles: macro
        tstx                                                    ;5 cycles
                endm
                                                ;5 cycles
```

```
; variable/data section
TINY_RAM_VARS: SECTION    RS08_SHORT              ; Insert here your data definition
Counter:       DS.B     1
FiboRes:       DS.B     1
tmpCounter: DS.B     1
tmp:           DS.B     1
ONTIME         DS.B     1
OFFTIME        DS.B     1
timers         ds.b     1



; code section
MyCode:        SECTION
main:
_Startup:


          ;cofig ICS
          mov     #HIGH_6_13(NV_ICSTRM), PAGESEL
          mov     MAP_ADDR_6(NV_FTRIM), ICSSC                    ; $3FFB
          mov     MAP_ADDR_6(NV_ICSTRM), ICSTRM                   ; $3FFA
        ;   bclr    ICSC1_CLKS,ICSC1
              mov     #ICS_DIV_1, ICSC2
            ; Use 10MHz

              ;config ARM
              clr    timers
              ;-----------------------------------------------------
              ;Config System
              ;-----------------------------------------------------
              mov       #HIGH_6_13(SOPT),  PAGESEL                          ;  Init
Page register
              mov       #(mSOPT_COPT|mSOPT_STOPE|mSOPT_BKGDPE),
MAP_ADDR_6(SOPT)                    ; COP disabled
      ;       Mov       #(mSOPT_STOPE),MAP_ADD_6(SOPT)                      ;
BKGD disable
              mov       #(mSPMSC1_LVDE|mSPMSC1_LVDRE),  MAP_ADDR_6(SPMSC1) ;
LVI enable
```

```
                    ;Config GPIO
                ; PWM - init H
                ;-------------------------------------------------------
    ;           mov                                         #(mRED|mGREEN|mBLUE),
PTAD                                        ; Initial port
                mov     #(mRED|mGREEN|mBLUE),  PTADD                    ;
DIM0 and PWM Output pins
                ;-------------------------------------------------------
                ;Config KBI
                ;-------------------------------------------------------
;               lda     #mBUTTON
;               sta   MAP_ADDR_6(PTAPE)                         ;Enable Pullup
;               sta     KBIPE                                   ;KBI Enable
;       bset    KBISC_KBACK, KBISC



                ;-------------------------------------------------------
                ;Config ACMP
                ;-------------------------------------------------------
    ;           mov     #(mACMPSC_ACME), ACMPSC                 ;       Enable
ACMP

                ;-------------------------------------------------------
                ;Config MTIM
                ;
                ;Timer prescalar=1 -> Timer clk = 1MHz
        ;Bus = 1MHz
        ;Max OF period = 256us
        ;Timer resolution = 1000ns
                ;-------------------------------------------------------
                mov     #(MTIM_BUS_CLK|MTIM_DIV_64),
MTIMCLK                 ;MTIMCLK=#00000000B
                mov     #255, MTIMMOD
                mov     #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC
MAIN1:
            ;   brset MTIMSC_TOF, MTIMSC, REDLED
            ;   JMP MAIN1
              JMP REDLED
```

```
PWMGEN:
        MOV     #240,MTIMMOD
        MOV     MTIMMOD,ONTIME
PWMON:
        LDA     ONTIME
        CBEQA #0,PWMEND
        EOR     #11111111B
        STA     OFFTIME
        CLR     PTAD
        MOV     ONTIME,MTIMMOD
        DEC     ONTIME
PWM1:
        mov         #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC
LOOP1:
        brset MTIMSC_TOF, MTIMSC, TIMERS_LOOP
        JMP     LOOP1
TIMERS_LOOP:
        lda     timers
        cbeqa #04,PWMOFF
        inca
        sta     timers
        JMP     PWM1

 PWMOFF:
        MOV    #(mRED|mBLUE|mGREEN),PTAD
 ;       bset RED,BLUE,GREEN,PTAD
        MOV    OFFTIME,MTIMMOD
 PWM2:
        mov         #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC
 LOOP2:
        brset MTIMSC_TOF, MTIMSC, TIMERS_LOOP2

        JMP LOOP2

 TIMERS_LOOP2:
        lda    timers
        cbeqa #00,PWMON
        Deca
        sta    timers
        JMP    PWM2
```

```
PWMEND:
        RTS

PWMGEN_N:
        MOV     #240,MTIMMOD
        MOV     MTIMMOD,ONTIME
PWMON_N:
        LDA     ONTIME
        CBEQA #0,PWMEND_N
        EOR     #11111111B
        STA     OFFTIME
        MOV #(mRED|mBLUE|mGREEN),PTAD
        MOV     ONTIME,MTIMMOD
        DEC     ONTIME
PWM1_N:
        mov         #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC
LOOP1_N:
        brset MTIMSC_TOF, MTIMSC, TIMERS_LOOP_N
        JMP     LOOP1_N
TIMERS_LOOP_N
        lda     timers
        cbeqa #04,PWMOFF_N
        inca
        sta     timers
        JMP     PWM1_N

 PWMOFF_N:
        CLR    PTAD
;        bset RED,BLUE,GREEN,PTAD
        MOV    OFFTIME,MTIMMOD
 PWM2_N:
        mov         #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC
 LOOP2_N:
        brset MTIMSC_TOF, MTIMSC, TIMERS_LOOP2_N

        JMP LOOP2_N

 TIMERS_LOOP2_N:
        lda    timers
        cbeqa #00,PWMON_N
```

```
        Deca
        sta    timers
        JMP    PWM2_N
PWMEND_N:
        RTS
REDLED:
                mov     #(mRED) , PTADD
          JSR PWMGEN
        JSR PWMGEN_N

GREENLED:
                mov     #(mGREEN),    PTADD
        JSR PWMGEN
        JSR PWMGEN_N
BLUELED:
                mov     #(mBLUE) ,    PTADD
        JSR PWMGEN
        JSR PWMGEN_N
BGLED:
                mov     #(mRED|mGREEN) , PTADD
        JSR PWMGEN
                JSR PWMGEN_N
RBLED:
                mov     #(mRED|mBLUE) ,    PTADD
        JSR PWMGEN
        JSR PWMGEN_N
GBLED:
                mov     #(mGREEN|mBLUE),PTADD
        JSR PWMGEN
        JSR PWMGEN_N
RGBLED:
                mov     #(mRED|mBLUE|mGREEN) , PTADD
        JSR PWMGEN
        JSR PWMGEN_N
                JMP    REDLED

                org     $3ffc
                dc.b    $00
           ;    JMP REDLED
```

Open Source BDM Interface Users Manual

1.0 Introduction to Open Source BDM

This document describes an Open Source programming and debugging development tool designed to work with Freescale HCS08 microcontrollers. Called Open Source BDM, it can be obtained from the 8-bit message board at http://forums.freescale.com. While there is no support for Open Source BDM from Freescale, the Open Source BDM is provided with all required source code for both hardware and software components. Because it is open source, the source code can be used and/or modified from its original design free of charge. Figure 1 provides a pictorial overview of the typical connections required for programming and debugging using the Open Source BDM. A PC connects to the Open Source BDM PCB, in turn the PCB is connected to a Programming/Debug target. In this example, the GB60 demonstration boards are being programmed and debugged.

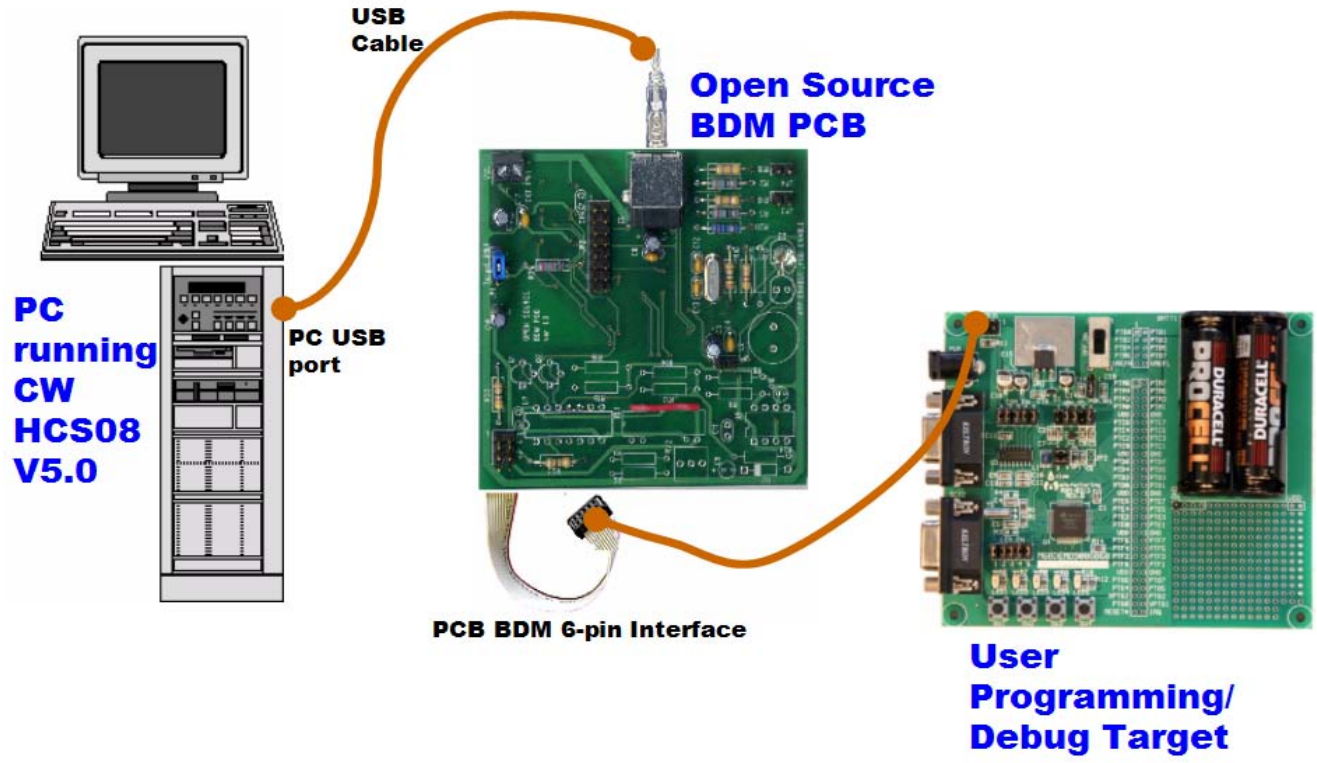


Figure 1. Debugging with Open Source BDM

Open Source BDM, with its hardware and software components, provides a transparent connection between a computer running CodeWarrior Development Studio for HCS08 version 5.0 to a Freescale HCS08 microcontroller via the microcontrollers BKGD pin. With a connection to the BKGD pin, the Open Source BDM enables debuggers and other software tools to communicate with the microcontroller including downloading of user code into the

microcontroller's on-chip flash. Programming and debugger functionality is made possible by the HCS08 microcontroller's Background Debug Controller (BDC) and In-Circuit Emulator (ICE) Debug (DBG) modules.

1.1 About the HCS08 BDC/ICE Debug Module

HCS08 microcontrollers contain a single-wire background debug interface, supporting in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. The BKGD pin on HCS08 devices provides this single-wire background debug interface to the on-chip BDC and ICE Debug modules. See the Development Tools chapter of any HCS08 data sheet for more information about both the BDC and ICE Debug modules. While the interface is single wire, typically a 6-pin connector, a BDM port is used to interface with the target. Figure 2 depicts the 6-pin BDM port.

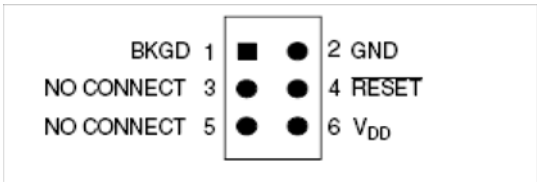


Figure 2 . Target 6-Pin Connector for BKGD Pin

The primary function of this pin is for bidirectional serial communication of active background mode commands and data. During reset, this pin is used to select between starting in active background mode or starting the user's application program. Additionally, this pin requests a timed sync response pulse, allowing a host development tool to determine the correct clock frequency for background debug serial communications.

BDC commands are sent serially from a host computer to the BKGD pin of the target HCS08 MCU. All commands and data are sent MSB-first using a custom BDC communications protocol. Two groups of BDC commands are listed below. A complete list of these commands is provided in the Development Tools chapter.

1. Active Background Mode Commands: These commands allow the CPU registers to be read or written. Users can also trace one user instruction at a time, or begin the user program in this mode.

2. Non-intrusive Commands: These commands permit read or write of MCU memory locations, or access status and control registers within the background debug controller. Non-intrusive commands can be executed at any time.

With a single-wire background debug interface, a relatively simple interface pod is used to translate commands from a host computer into commands for the BDC. In the case of the Open Source BDM, a Low-speed (LS) universal serial bus (USB) interface is used to communicate

between the host PC and the pod. Functionality provided by the BDC and ICE DBG modules include:

- Single pin for mode selection and background communications
- BDC registers not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from Stop or Wait modes
- One hardware address breakpoint built into BDC
- Oscillator runs in Stop mode when BDC is enabled
- COP watchdog disabled while in Active Background mode
- Features of the debug module (DBG) include:
  - Two trigger comparators: Two address + read/write (R/W) or One full address + data + R/W
  - Flexible 8-word by 16-bit first-in, first-out (FIFO) buffer for capture information: change-of-flow addresses or event-only data
  - Two types of breakpoints: Tag breakpoints for instruction opcodes and force breakpoints for any address access
  - Nine trigger modes

1.2 Scope of the User Manual

This documentation describes the Open Source BDM solution. The description provides sufficient detail, enabling the Open Source community to update, change, and maintain the Open Source BDM. This user manual describes the the organization and operation of the following:

- Open Source BDM Generic Debug Instrument (GDI) DLL plug-in for the CodeWarrior hiwave debugger
- Open Source BDM interface DLL and USB driver (libusb.lib)
- HC908JB16 USB/BDM firmware
- Open Source BDM PCB hardware

1.3 Open Source BDM Overview

While using the Open Source BDM, supplemental understanding of a debugger pod may be required with possibly more user maintenance to provide a low-cost alternative for development tools. The Open Source BDM software and hardware were developed to support existing and future HCS08 devices. It is designed to provide maximum accuracy and performance for HCS08 devices. Open Source BDM is compatible with CodeWarrior Studio HCS08 version 5.0, and currently supports the following HCS08 families:

- GB60
- AW6
- QG8
- RCx

As other HCS08 families are developed and released, the open source community may have to modify the Open Source BDM along with installation of CodeWarrior Studio HCS08 version 5.0 service packs. Other features of the Open Source BDM are:

- Open source distribution
- Low-cost development tools
- Designed for the modern and widely available USB interface
- Firmware developed with the HCS08 CodeWarrior Special Edition version
- Supports targets with operating voltages from 1.8 to 5.0V
- Supports target bus speeds from 1.0 to 20MHz
- Double-side PCB uses mostly though-hole design for ease of assembly
- PCB can optionally supply 5.0V power to the MCU target board
- PCB has an optional 16-pin MONO8 header to program the JB16
- PC USB DLL drivers source code is provided in ANSI C
- JB16 firmware source is mostly C with some assembly for time critical operations
- Firmware can be flash programmed using USB and Freescale interface. See AN2399 - these procedures are documented within this document.
- Windows USB drivers included using Open Source LIBUSB USB drivers
- Uses a JB16 MCU with 12MHz crystal clock source
- Documented API for easy integration into debuggers

As indicated, all the Open Source BDM source code is provided without charge. In addition, the Open Source BDM hardware uses low-cost components and is very simple to assemble. Unassembled PCB cost approximately under five U.S. dollars and the components can also be obtained for less than U.S. five dollars, depending on volume. Subsequent sections provide a detailed description of both the Open Source BDM hardware and software.

1.4 Open Source BDM Block Diagram

To better understand the implementation of the Open Source BDM, Figure 3 delineates the Open Source BDM solution into its most basic components. The diagram illustrates part of the Open Source BDM IP residing on the PC host for the debugging software and some resides on the Open Source BDM PCB.

At the time of release, the Open Source BDM is supported by the version 5.0 release of CodeWarrior Development Studio for Freescale HC(S)08 Microcontrollers. The Studio provides both a software development IDE and a debugger.

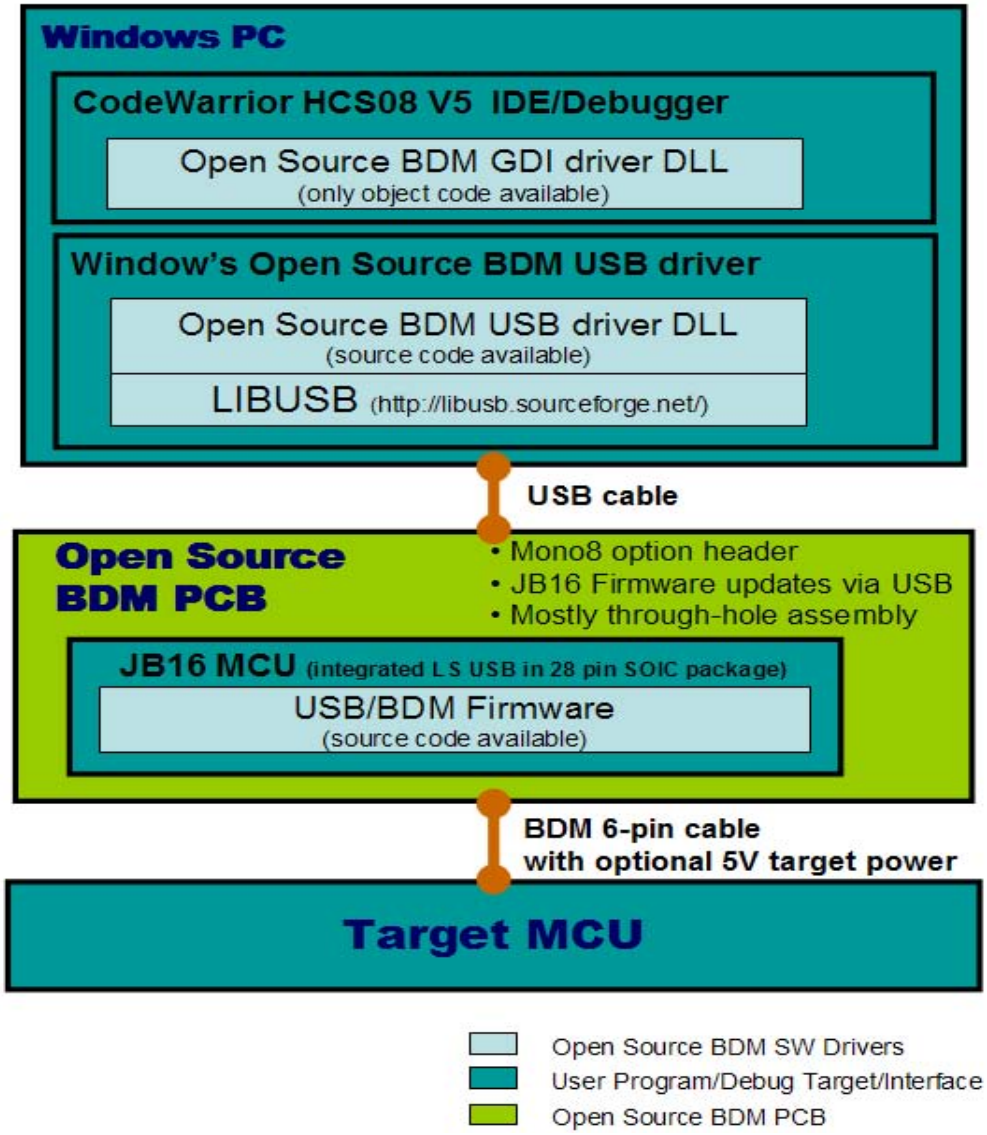


Figure 3. Block Diagram

Figure 3 illustrates five primary components, four software in nature and one hardware; these are itemized below. All of the software components are intended to be used as binaries by majority of users. For those who would like to look deeper, the source code of the JB16 firmware and the Open Source BDM interface DLL is provided in Table 1.

Table 1 Open Source BDM IP Components

Component	Description/Interfaces	Available/Comments
Open Source BDM GDI driver	File - OpenSourceBDM_gdi.dll Software interfacing with the CW V5.0 debugger GDI interface, providing function call to the Open Source BDM USB driver.	This software is available only as object code in the form of a DLL.
Open Source BDM USB driver	File - OpenSourceBDM.dll Software interfacing with the Open Source BDM GDI driver, providing function calls to the Windows USB drivers. This file is also required for the PC when it detects the Open Source BDM PCB as a new USB device on the PC USB port.	This software is available as both object (DLL) and source code. The DLL is provided in an Open Source BDM USB install package. Point the PC Hardware Wizard to the USB install package when the PC detected the Open Source BDM PCB.
Windows USB drivers	File - libusb.lib Software used by the Open Source BDM USB driver it interfaces to PC USB ports.	This USB library is compiled with the Open Source BDM USB driver and is also part of the Open Source BDM USB install package. The source code for libusb can be found at (http://libusb.sourceforge.net/)
JB16 USB/BDM firmware	File - OpenSourceBDM.s19 Software running on the JB16 that receives commands via USB from the PC and converts them into commands as defined by the BDC. These command are serial outputted - “bit - banged” - by the JB16 using port pins to drive the BKGD pin on the user’s target.	This file is provide as a S-recorded and need to be programmed the the JB16 on-chip flash
Open Source BDM PCB hardware	PCB: OpenSourceBDM Pod ver. 1.0 This hardware contains the JB16 and circuitry, clock, and power to provide the interface to the program/debug target.	All schematics and gerber files for the Open Source BDM PCB is provided along with a Bill of Materials (BOM). Also included is a complete HW description which enables you to build the interface

1.5 Open Source BDM Package

This section describes the contents of the Open Source BDM package. The package is distributed in zip file and includes both release and development folders. The development files folder contains source files while the release files folder contains only binary release files. A detailed discussion is provided for each of these items in the next sections. An illustration of the unzipped Open Source BDM package directory structure is provided in Figure 4.

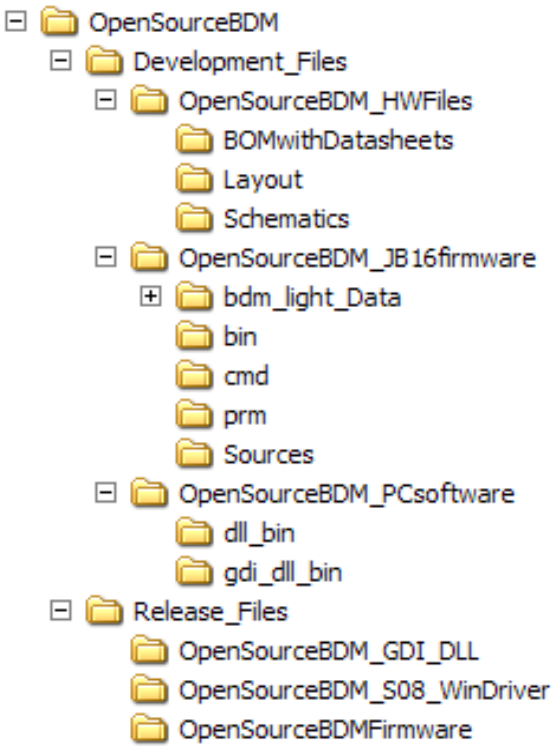


Figure 4. Unzipped Open Source BDM Package

1.6 Support and Licensing

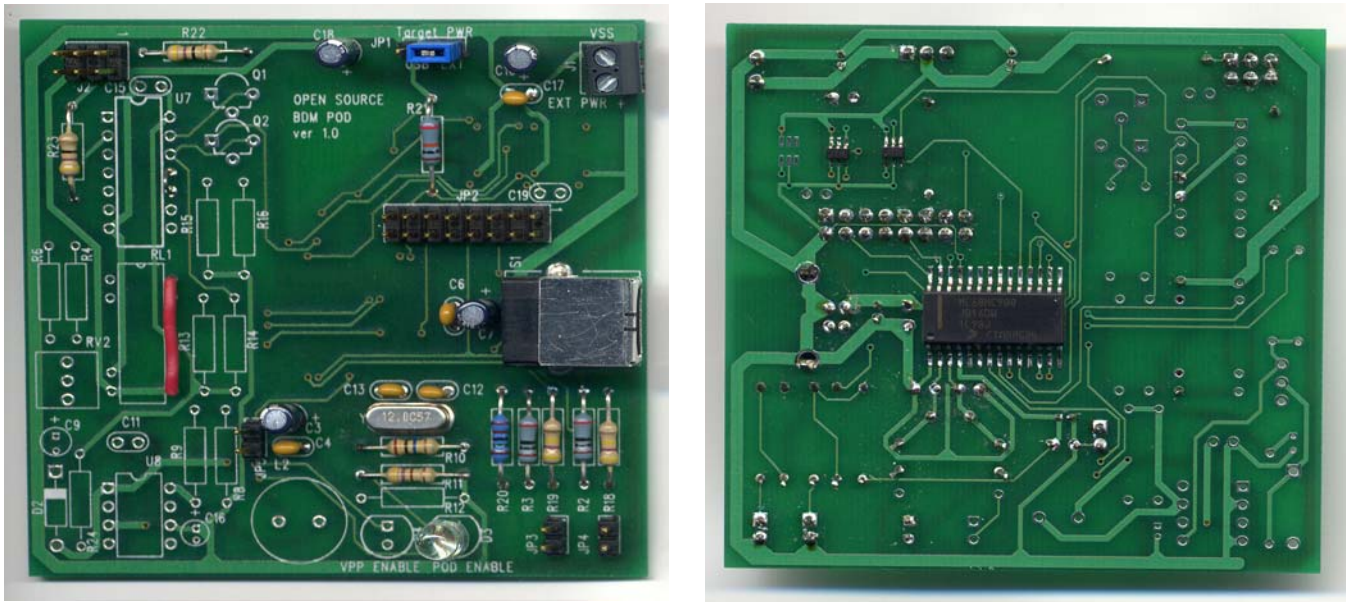
Open Source BDM is not supported by Freescale; it is open source. Any bugs, enhancements, or support questions should be addressed through the Open Source BDM forum. Open Source BDM has been thoroughly tested, but there are no guarantees about error-free operation. All of the work, with exception of the GDI DLL, is available to anyone under the GNU general public license. The Open Source BDM is a deriviative project of the TBMDL project.

2.0 Open Source BDM PCB Hardware and MC68HC908JB16

Photos of the Open Source BDM PCB top and bottom sides are provided in Figure 5. The bottom side shows the MC68HC908JB16 used to decode incoming USB messages into BDC commands and then bit-bang the BDC commands to a target. The Open Source BDM PCB is designed with many desirable features, making it a robust development tool. The following list is a summary of those features:

- Based on MC68HC908JB16 MCU from Freescale with a 12MHz crystal
- Provides circuitry to facilitate Firmware upgrades of the MC68HC908JB16
- USB interface with a type B USB connector

- PCB provides optional circuitry used to update the Open Source BDM PCB to support other targets besides HCS08 devices
- Double-side PCB uses mostly through-hole design for ease of assembly
- PCB can optionally supply 5.0V power to the MCU target board
- PCB has an optional 16-pin MONO8 header to program the JB16



Top Side **Bottom Side**

Figure 5. Open Source BDM PCB

2.1 MC68HC908JB16 Overview

The Open Source BDM is based on MC68HC908JB16 MCU from Freescale. The MC68HC908JB16 provides:

- Relatively low-cost HC08
- A low speed USB 2.0 compatible interface
- Surface mount 28-pin SOIC package

Figure 6 provides quick reference of the MC68HC908JB16 pinout.

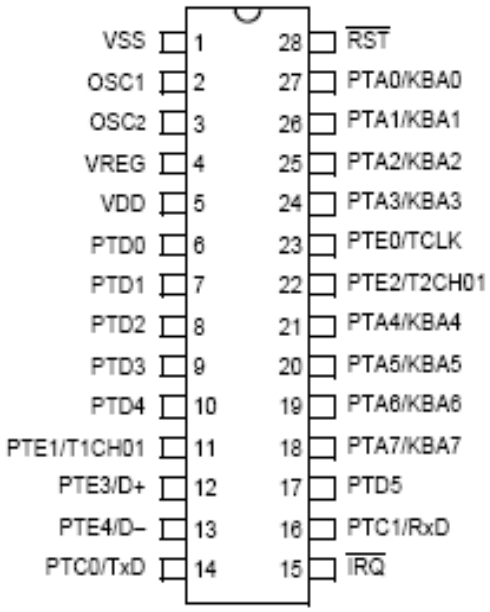


Figure 6. MC68HC908JB16 Pin-Out

2.1.1 Other MC68HC908JB16 Features

- High-performance M68HC08 architecture
- Low-power design; fully static with Stop and Wait modes
- 6MHz internal bus frequency
- 16,384 bytes of on-chip FLASH memory with Security1 feature
- 384 bytes of on-chip random access memory (RAM)
- Up to 21 general-purpose input/output (I/O) pins
- Two 16-bit, 2-channel timer interface modules (TIM1 and TIM2)
- Universal Serial Bus specification 2.0 low-speed functions
  - — In-circuit programming capability using USB communication
- Serial communications interface (SCI) module
- System protection features:
  - — Optional computer operating properly (COP) reset
  - — Optional Low-voltage detection with reset
- IRQ interrupt pin with internal pull-up and Schmidt-trigger input

2.1.2 MC68HC908JB16 USB

Open Source BDM is designed to use USB as the means of talking to the computer because:

- USB provide plug and play functionality
- USB can provide power to the Open Source BDM PCB and the target, avoiding the requirement of an additional power supplies
  - By routing the USB power to the BDM connector, the Open Source BDM PCB can also provide 5.0V power to the target board

The LS USB peripheral of the MC68HC908JB16 provides sufficient USB functionality to facilitate the Open Source BDM operations. The MC68HC908JB16 USB features include:

- 1.5Mbps data rate
- On-chip 3.3V regulator
- Three Endpoints
  - Endpoint 0 with 8-byte transmit buffer and 8-byte receive buffer
  - Endpoint 1 with 8-byte transmit buffer
  - Endpoint 2 with 8-byte transmit buffer and 8-byte receive buffer

2.1.3 MC68HC908JB16 Development

- The development environment and the hiwave debugger looks the same for both HC08 and HC(S)08
- The development environment is available free of charge for the size of project

2.2 Open Source BDM PCB

The schematics are provided for the Open Source BDM PCB in Figure 7. The schematics provide details of the power and clocking connections for the Open Source BDM PCB. These schematics also show various jumpers settings. The optional circuity is not discussed in this document.

Besides the HC908JB16, the schematic of the Open Source BDM interface shows several main parts:

- A USB interface with jumper options
  - Options to program the HC908JB16
  - Options for standard Open Source BDM operation
- BDM interface driver based on 74LVC1T45 buffer with tri-state outputs

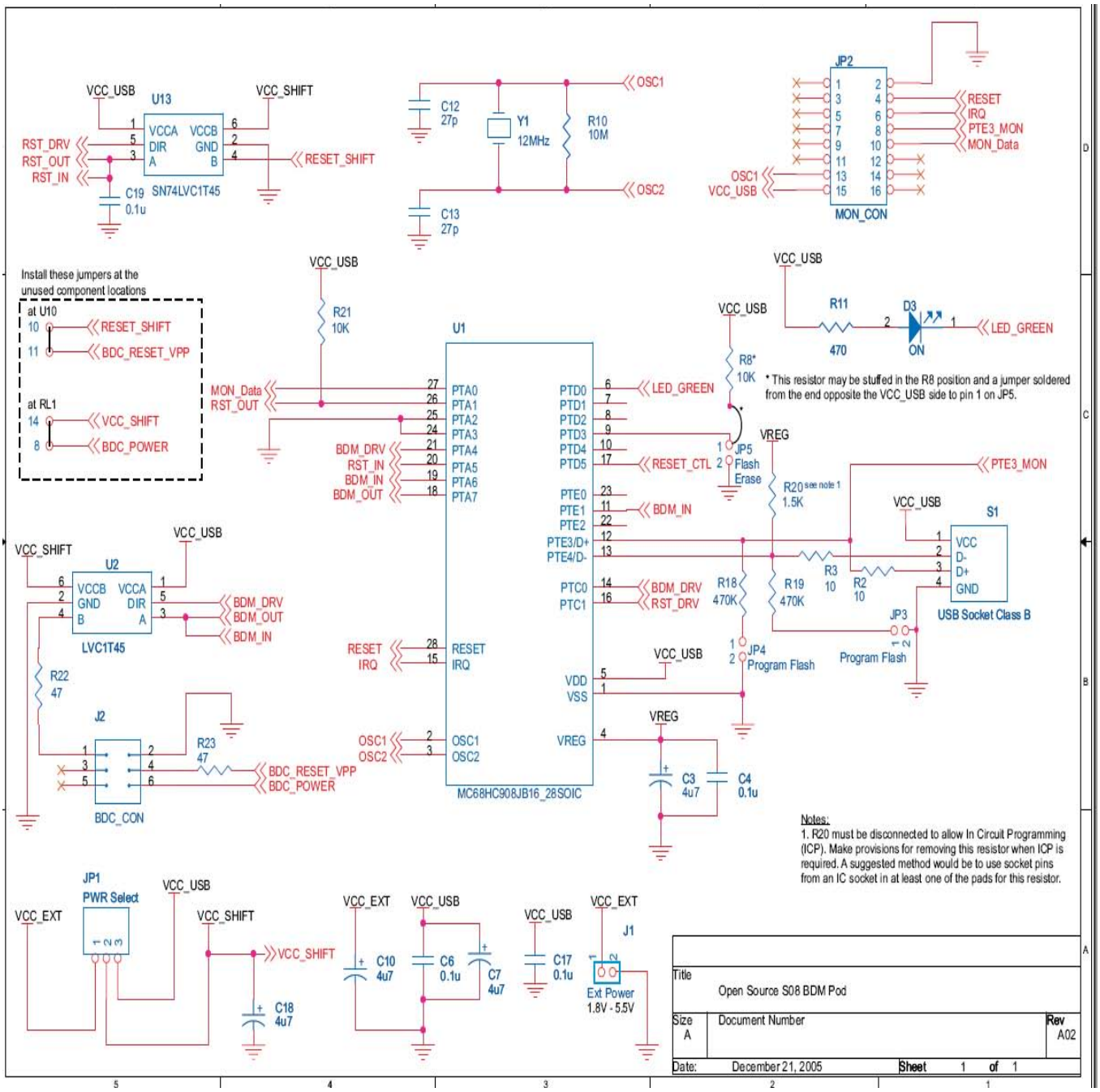


Figure 7 . Open Source BDM Schematic

2.2.1 USB interface Details

A 1.5 KΩ (±1%) pull-up resistor is provided on D-USB signal. A jumper is added to the pull-up resistor in order to assist in the ICP programming of the JB16, using techniques from AN2399.

Note: PCB is provided with footprint for the USB B connector. This is actually in violation of the USB specification as low-speed devices should have the cable hard-wired to them, but a detachable cable is very useful. If violation of the specification is not desired, solder the cable straight into the PCB.

2.2.2 Response Time and Transfer Rate

By the nature of the USB protocol the response time for low and full speed devices cannot be below 1ms. Optimization of the communication protocol on the USB to achieve maximum throughput was tested. However, practical limitations, caused by the Windows operating system, cause additional delays. Average execution times for different kinds of commands are detailed in Table 2.

Table 2 Average Execution Command Times

Command Type	Description	Average Execution Speed
Short	Commands which transfer up to 5 bytes of data into Open Source BDM and require no return values.	3ms
Normal	Commands which transfer up to 5 bytes of data into Open Source BDM and request up to 8 bytes of return values.	4ms
Data transfer	Commands which transfer large blocks of data.	6.7 kB/s

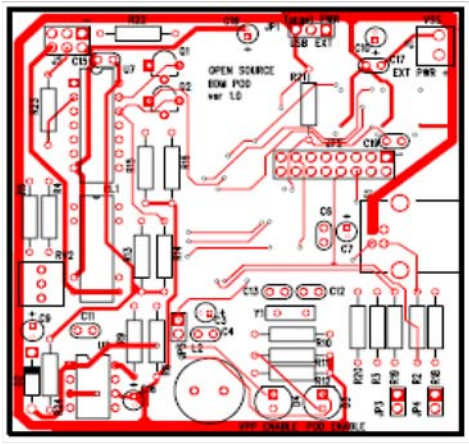
When programming the flash of the target MCU, there is additional overhead created by the flash programming routines. The speed is also dependant on crystal frequency; the higher, the better. The Metrowerks hiwave debugger with Open Source BDM interface connected to HCS08 target with 4.0MHz crystal typically programs the flash at 2.7kB/s rate.

2.2.3 AN2399 ICP Programming Jumpers

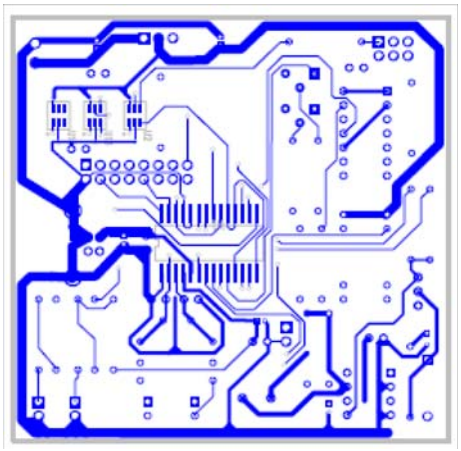
The ICP programming setup, procedures, and operation details are provided in the *JB16 USB/ BDM firmware* section.

2.3 Open Source BDM Layout and Guidelines

The Open Source BDM is a two-sided PCB, roughly 3 in x 3 in larger, illustrated in Figure 8. The gerber files are provided with the Open Source BDM in a later release. The gerber files can be used to develop the design into another form factor if desired.



Top View



Bottom View

Figure 8. Open Source PCB Layout

2.4 Getting Open Source BDM Components

A complete BOM is provided in Table 3, including all parts required for the Open Source BDM. Each part is provided with order numbers from Digi-Key. Some of these components are available as free samples. Free samples of Freescale's HC908JB16 are available from www.freescale.com.

Table 3 BOM

Generic Part	Description	Package	Component No.	Digi-Key Part No.	Qty
MC68HC908JB16DW	MCU	28-Pin SOIC	U1	MC68HC908JB16DW-ND	1
SN74LVC1T45DBVR	Level Shifter	6-Pin SOT-23	U2, U13	296-16843-1-ND	2
Crystal, 12 MHz	Citizen HC49US12.000MABJB-UB	HC-49US	Y1	300-8492-ND	1
Capcitator, Electronic, 4.7µF	Panasonic ECE-A1VKG4R7	Radial Lead	C3, C7, C10, C18	P920-ND	4
Capcitator, Ceramic, 27pF	EPCOS B37979N1270J054	Radial Lead	C12, C13	495-1005-1-ND	2
Capcitator, 0.1µF	Kemet C320C104K5R5CA	Radial Lead	C4, C6, C17, C19	399-2054-ND	4
Resistor, 10 1/4W 5%		Axial Lead	R2, R3	10QBK-ND	2
Resistor, 470K 1/4W, 5%		Axial Lead	R18, R19	470KQBK-ND	2
Resistor, 10M 1/4W 5%		Axial Lead	R10	10MQBK-ND	1
Resistor, 470 1/4W 5%		Axial Lead	R11	470QBK-ND	1
Resistor, 47 1/4W 5%		Axial Lead	R22, R23	47QBK-ND	2
Resistor, 1.5K 1/4W 1%		Axial Lead	R20	1.5KXBK-ND	1
Resistor, 10K 1/4W 5%		Axial Lead	R21, R8*	10KQBK-ND	2
LED, Green, Round, 5mm, T13/4	Fairchild HLMP3950A	Radial Lead	D3	HLMP3950AFS-ND	1
USB Connector, Type B, PCB	Tyco 292304-1	Through Hole PCB	S1	A31725-ND	1
Power Connector, 2-Pin Terminal Blk	On-Shore Tech ED555/2	Through Hole PCB	J1	ED1514-ND	1
Dual Row Header, 22-Pin	Molex/Waldom Electronics Corp.	Through Hole PCB	JP2, J2	WM6822-ND	1
Single Row Header, 9-Pin	Molex/Waldom Electronics Corp.	Through Hole PCB	JP1, JP3, JP4, JP5	WM6509-ND	1

3.0 JB16 USB/BDM Firmware

The HC908JB16 firmware was developed with the 3.1 special edition version of the CodeWarrior Development Studio for HCS08. The project also works with the 5.0 special edition version. The special edition provides a free solution for developing and debugging the Open Source BDM project.

3.1 Firmware Description

The CodeWarrior Project Manger window for the HC908JB16 firmware project is illustrated in Figure 9. The source files are organized so the firmware is delineated into logical blocks including:

- USB Block
- Command Processing Block
- BDM Block

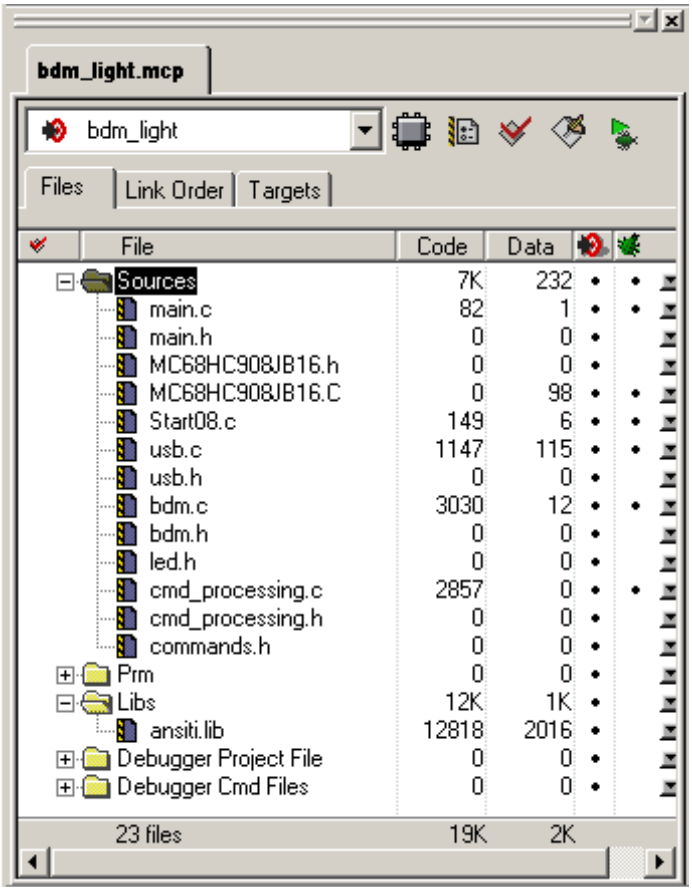


Figure 9. Open Source BDM CodeWarrior Project Manager

main.c/.h	Main program
Command.h	Provides command code use within the USB messages
cmd_processing.c/.h	Decodes USB messages and commands into BDM commands conform to the BDC. This source code calls functions from bdm.c/.h.
bdm.c/.h	Provides tx/rx function to driver JB16 ports controlling the targets BKGD pin. Time critical code uses assembly code.
usb.c/.h	Provides function for USB communication using endpoint 0 and endpoint 2

3.2 Programming Firmware into HC908JB16

Before a computer can recognize the Open Source BDM interface as a valid USB peripheral, the Open Source BDM firmware requires downloading into the HC908JB16 microprocessor. The following text describes programming of a blank HC908JB16 using the USB interface. Programming the HC908JB16 using the USB interface uses the techniques described in AN2399. A step-by-step guide is provided below.

1. Unzip the USB_ICP_demo.zip onto your computer. Included in these files are the USB ICP drivers.

3.2.1 Configuring the Open Source BDM PCB

2. Close Jumpers JP3 and JP4. Also make sure R20 is not placed. This configures the JB16 for USB In Circuit Programming.
3. Connect the Open Source BDM board to your computer through the USB cable.

3.2.2 Installing USB ICP Drivers

4. Once the Open Source BDM board to the computer, you should be prompted that a new device has been detected and the Hardware Wizard should appear. Select *Install from a list or specific location*, illustrated in Figure 10.

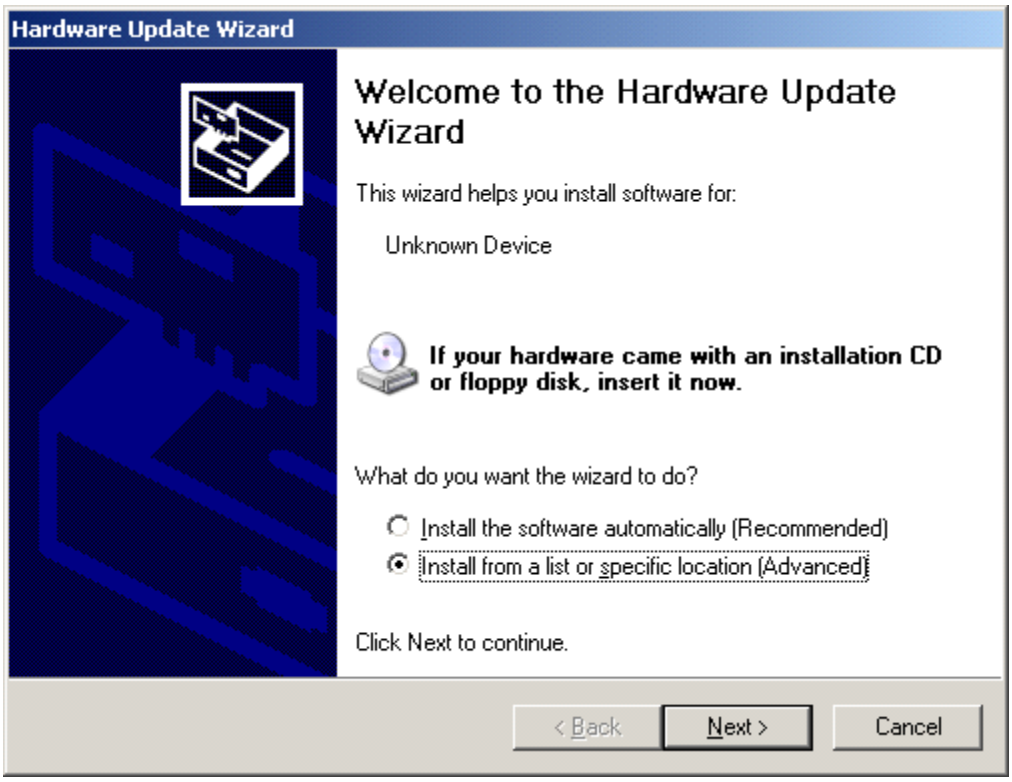


Figure 10. Installation Wizard Screen

5. Direct the install to the folder where the USBICP files reside. Your computer should now recognize a USBICP device is connected.

Note: It may be necessary to attempt this install many times, sometimes the computer will not recognize the board. If this occurs try connecting reset to GND before plugging in the USB cable, then releasie reset. Reset and GND are accessible at the MON08 header.

3.2.3 Operation of AN2399 Flash Programming Application

6. Open USBICP.exe, see Figure 11, when prompted for an IMP file select jb16_icp_me.imp.

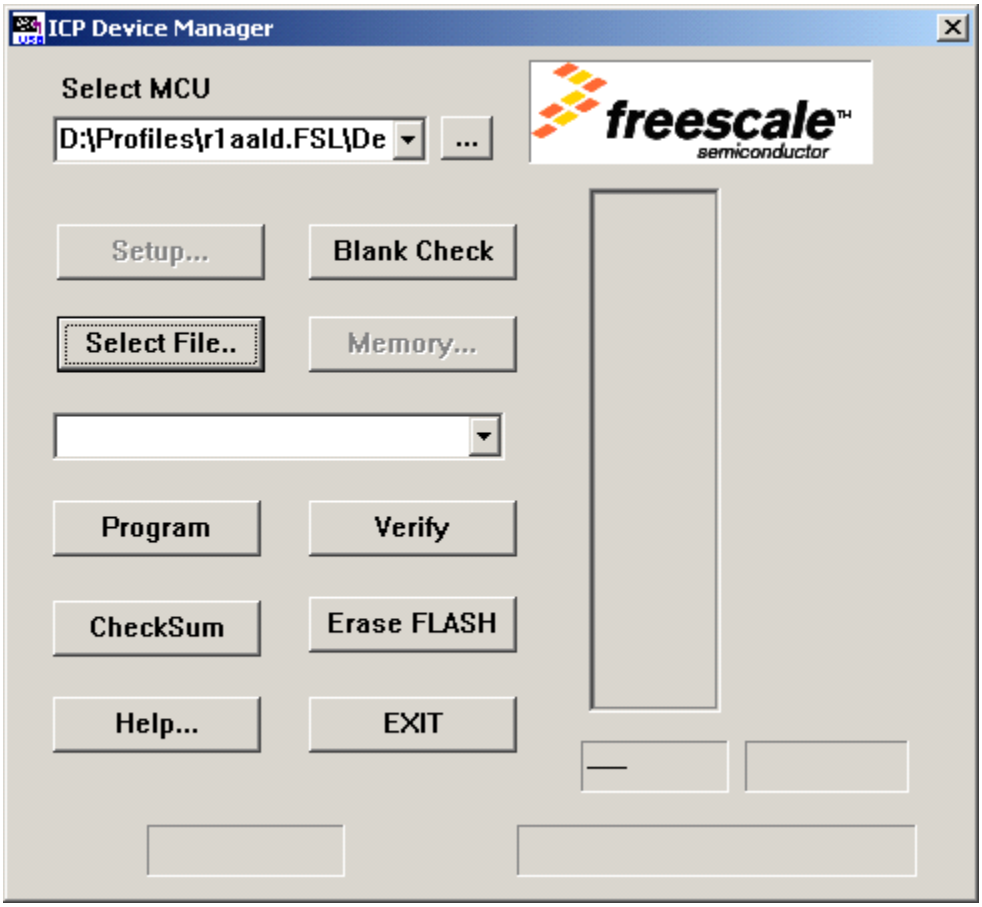


Figure 11. Flash Operation Screen

Selection of the Flash operation in the above screen results in the following screen in Figure 12.

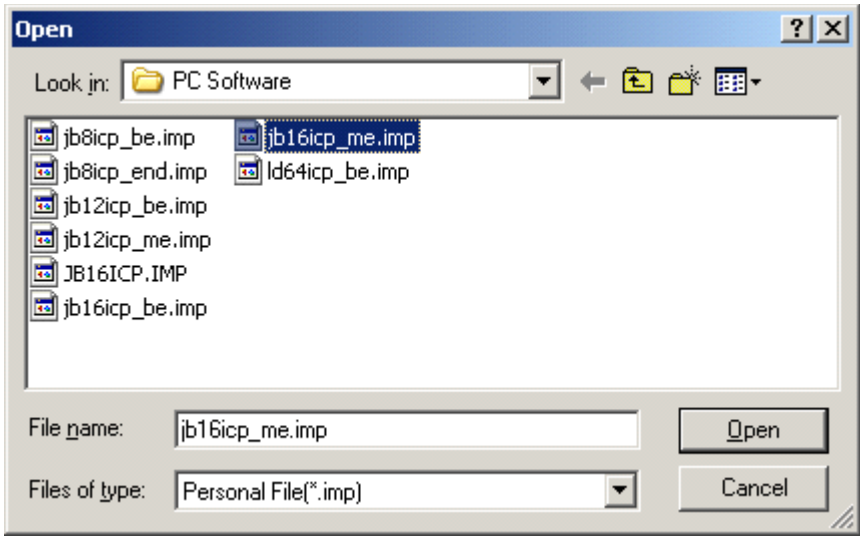


Figure 12. PC Software Section Screen

7. Select File. Navigate to the desired s19 file.

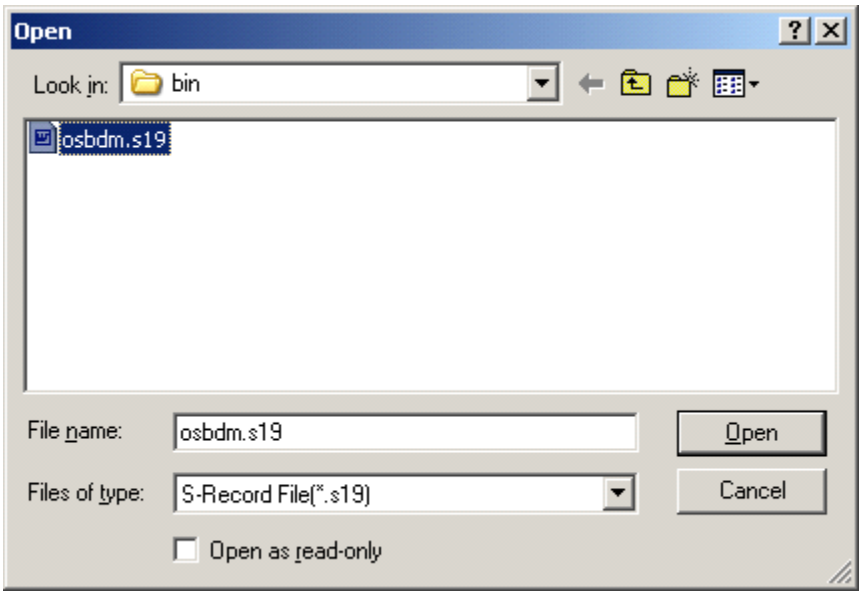


Figure 13. Open Screen

8. Select Program. The firmware is now installed.

3.2.4 Updating the USB Firmware

To update the firmware:

1. Power down the board by removing the USB connection.
2. Close jumper JP5. This tells the USB firmware to jump to the ICP routines upon power-up.
3. Connect the Open Source BDM board to your computer through the USB cable. The computer should recognize the board as a USB ICP device.
4. Open USBICP.exe. when prompted for an IMP file select jb16_icp_me.imp
5. Select *Erase Flash*.
6. If Problems persist with In Circuit Programming to update the firmware the best solution is to replace the JB16 unit with a blank part. This will ensure that an update to the firmware can be completed.

The Firmware is now removed. The steps outlined in *programming the firmware into the JB16* can be used to update the firmware.

Note: Remember to remove R20.

Note: Firmware installed on Alpha Units (Boards Pre-Programmed with the firmware) does not contain the ability to do this.

4.0 Open Source BDM PC Software

4.1 Overview

This section describes the Open Source BDM solution PC software components. These components include:

- Open Source BDM Generic Debug Instrument (GDI) DLL plug-in for the CodeWarrior hiwave debugger
- Open Source BDM interface DLL
- USB driver (libusb.lib)

Figure 14 illustrates these software components and a simplification of their interfaces.

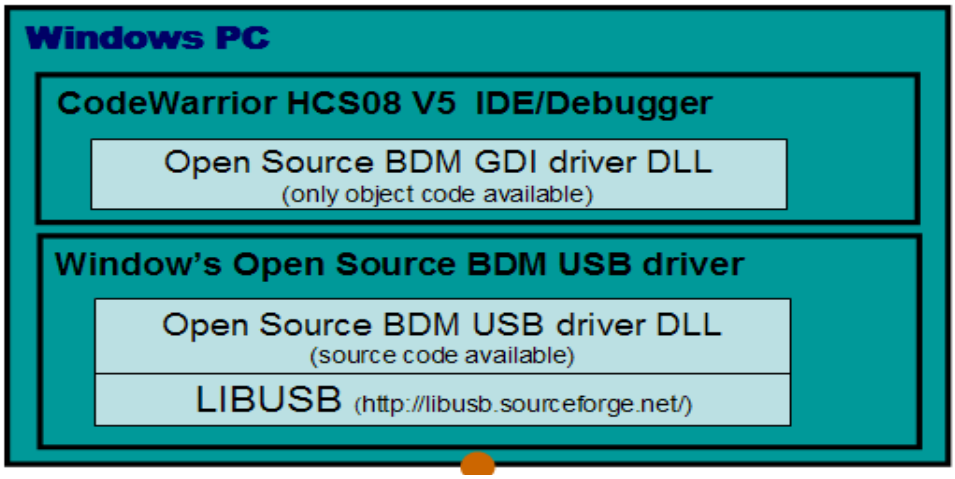


Figure 14. Open Source BDM Windows PC Software and Interfaces

4.2 Open Source BDM GDI DLL

The Open Source BDM GDI DLL plug-in for the CodeWarrior hiwave debugger is only available as a binary file. No source code is provided. The GDI DLL for the CodeWarrior hiwave debugger is created partially based on information not available in the public domain. The license attached to these files does not allow disclosure of the file's source code.

The Open Source BDM takes advantage of the CodeWarrior version 5.0 Degugger's Generic Debug Instrument (GDI) protocol interface. Tasking Incoporated provides more detailed information about the GDI Open Interface Specification (see http://www.tasking.com/resources/technologies/debuggers/gdikdi/); the Tasking specification is publicly available.

4.3 Open Source BDM DLL and LIBUSB

The Open Source BDM DLL provides an interface between the Open Source BDM GDI DLL and the Open Source BDM firmware. This section describes the API of the Open Source BDM DLL including a Windows Open Source USB drivers library, LIBUSB. The source code for the Open Source BDM DLL is available.

4.3.1 Command.h

Command.h provides command code use within the USB messages. This file matches the command.h file in the Open Source BDM firmware.

4.3.2 OpenSourceBDM.h/.c

OpenSourceBDM.c/.h provides the API functions for the Open Source BDM DLL. These functions are called by the Open Source BDM GDI DLL. The functions in turn call functions of LIBUSB. These functions are encapsulated into the OpenSourceBDM.dll. OpenSourceBDM.dll is a part of the Windows USB install package for the Open Source BDM. The Open Source BDM DLL functions are listed and briefly described below.

unsigned char OpenSourceBDM_OpenSourceBDM_dll_version(void)

Returns version of the DLL in BCD format (major in upper nibble and minor in lower nibble).

unsigned char OpenSourceBDM_init(void)

Initializes the USB interface, returning the number of Open Source BDM devices found attached to the computer. This function should be called before a device can be opened.

unsigned char OpenSourceBDM_open(unsigned char device_no)

Opens communication with device number *device_no*. First device has number 0. Returns 0 on success and non-zero on failure. A device must be open before any communication with the device can take place.

void OpenSourceBDM_close(void)

Closes communication with currently opened device.

unsigned int OpenSourceBDM_get_version(void)

Returns version of HW (MSB) and SW (LSB) of the Open Source BDM interface in BCD format.

unsigned char OpenSourceBDM_get_last_sts(void)

Returns status of the last executed command: 0 on success and non-zero on failure.

unsigned char OpenSourceBDM_set_target_type(target_type_e target_type)

This function sets target MCU type. *target_type* can be either *HC12* or *HCS08*. Returns 0 on success and non-zero on failure.

unsigned char OpenSourceBDM_target_sync(void)

Measures BDM frequency of the target using the SYNC BDM feature and connects to the target. Returns 0 on success and non-zero on failure (no device connected or the SYNC feature not supported). If this function succeeds, there is no need to set the BDM communication speed as it is measured automatically.

unsigned char OpenSourceBDM_target_reset(target_mode_e target_mode)

Resets the target MCU to normal or special mode. *target_mode* can be either *SPECIAL_MODE* or *NORMAL_MODE*. Returns 0 on success and non-zero on failure (reset pin stuck to ground, etc.).

unsigned char OpenSourceBDM_bdm_sts(bdm_status_t *bdm_status)

bdm_status is a pointer to allocated structure the function fills with current state of BDM communication. Returns 0 on success and non-zero on failure.

The structure has the following format:

typedef struct {

ackn_state_e ackn_state;

reset_state_e reset_state;

connection_state_e connection_state; } bdm_status_t;

ackn_state can be either *ACKN* (target supports ACKN BDM feature) or *WAIT* (target does not support ACKN BDM feature).

reset_state can be either *RESET_INACTIVE* (no reset activity detected) or *RESET_DETECTED* (target was reset since the last call). *reset_state* defaults to *RESET_INACTIVE* after each call.

connection_state can be *NO_CONNECTION* (no target MCU detected), *SYNC* (target supports the SYNC BDM feature) or *MANUAL_SETUP* (BDM speed was set-up by calling OpenSourceBDM_set_speed - see below).

unsigned char OpenSourceBDM_target_go(void)

Starts target code execution from current PC address. Returns 0 on success and non-zero on failure.

unsigned char OpenSourceBDM_target_step(void)

Steps over a single target instruction. Returns 0 on success and non-zero on failure.

unsigned char OpenSourceBDM_target_halt(void)

Brings the target into active background mode, i.e., debug mode with user code execution halted. Returns 0 on success and non-zero on failure.

unsigned char OpenSourceBDM_set_speed(float crystal_frequency)

Sets the BDM communication speed. *crystal_frequency* is crystal, or external source, frequency in MHz. Returns 0 on success and non-zero on failure. It is essential to provide frequency accurate at least to two decimal places in MHz.

float OpenSourceBDM_get_speed(void)

Returns crystal (or external source) frequency of the target in MHz.

unsigned char OpenSourceBDM_read_byte(unsigned int address)

Reads one byte from memory at the supplied address.

void OpenSourceBDM_write_byte(unsigned int address, unsigned char data)

Writes one byte to memory at the supplied address.

void OpenSourceBDM_read_block(unsigned int address, unsigned int count, unsigned char *data)

Reads *count* bytes from address *address*. The data is written to a user supplied buffer.

void OpenSourceBDM_write_block(unsigned int address, unsigned int count, unsigned char *data)

Writes *count* bytes to address *address*. The data is take from a user supplied buffer.

unsigned char OpenSourceBDM_read_regs(registers_t *registers)

Reads contents of target registers. Returns 0 on success and non-zero on failure. The register values are filed into user-allocated structure of the following format:

```
typedef union {
struct {
unsigned int pc;
unsigned int sp;
unsigned int ix;
unsigned int iy;
unsigned int d;
unsigned int ccr;
} hc12;
struct {

unsigned int pc;
unsigned int sp;
unsigned int hx;
unsigned int a;
unsigned int ccr;
} hcs08; } registers_t;
```

void OpenSourceBDM_write_reg_pc(unsigned int value)

Writes a new value into the PC target register.

void OpenSourceBDM_write_reg_sp(unsigned int value)

Writes a new value into the SP target register.

void OpenSourceBDM_write_reg_x(unsigned int value)

Writes a new value into the H:X (S08) target register.

void OpenSourceBDM_write_reg_d(unsigned int value)

Writes a new value into the A (S08) target register.

void OpenSourceBDM_write_reg_ccr(unsigned int value)

Writes a new value into the CCR target register.

LIBUSB

The LIBUSB is Open Source software available under combination of GNU general and lesser general public licenses. The Open Source BDM DLL functions utilize the following LIBUSB API functions:

- usb_find_buses
- usb_find_devices
- usb_close
- usb_open
- usb_get_version
- usb_init
- usb_control_msg
- usb_bulk_write
- usb_bulk_read

5.0 Erasing and Programming Algorithms

Other aspects of the development tool system are the erasing and programming algorithms. Before a new program can be uploaded into the target system, the target must be erased before it can be programmed. In order to both erase and program the target, a program must be loaded into RAM first. Both are separate erase and program algorithms. These programs provide a means to write to, and erase flash.

Every part family and every derivative has a distinct erasing and programming algorithm file. These files are not included in this version of the Open Source BDM, but some erasing and programming algorithm file from within the CodeWarrior installation can be used. Erasing and programming algorithm are not directly supported by CodeWarrior, consquently they are not guaranteed. Figure 15 illustrates the location of these files within the CodeWarrior installation in the fpp directory. It is not necessary to access the files directly, the Open Source BDM is configured to access the files during target erase and programming operations.

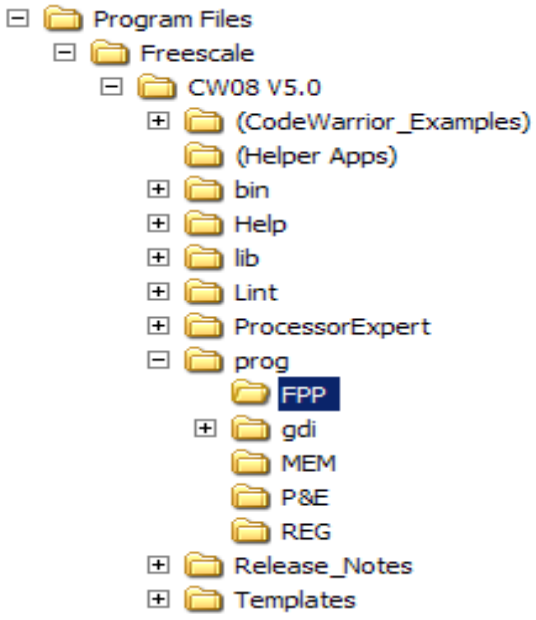


Figure 15. Path to FPP

6.0 Installation and Operation of the Open Source BDM

6.1 Configuration of the Open Source BDM PCB

This section assumes the Open Source BDM firmware is programmed on to the JB16.

6.1.1 Open Source BDM PCB HCS08 Configuration

Table 4 details the Open Source BDM PCB jumper configuration settings for Programming and debugging targets for the HCS08. Other types of target may required other settings.

Table 4 Open Source BDM PCB HCS08 Configuration

Jumper	Description	Setting
JP1	Target Power Selection	• Short 1-2: Power target externally via J1 or target is self-powered • Short 2-3: Power target using 5V USB power
JP3	Flash Program	Off
JP4	Flash Program	Off
JP5	Flash Erase	Off

6.2 Installing Windows Open Source BDM DLL and USB Drivers

The following procedure specifies the installing of the Open Source BDM USB hardware drivers under the Windows operating system. This procedure assumes Open Source BDM windows USB driver package is unpacked onto the development PC, running the CodeWarrior Studio.

With the Open Source BDM PCB configured, the Open Source BDM PCB can be connected to the development PC USB port. When the configured Open Source BDM PCB is connected to the PC for the first time, the Windows operating system recognizes a new USB device, the Open Source BDM PCB. This initial connection starts the Windows driver installation procedure. Figure 16 illustrates the Windows New Hardware Wizard dialog box that opens.

Figure 16. Found New Hardware Dialog Box

For this installation, select the option to *Install from a Specific Location,* then click *Next*. When the *Next* button is selected, the *Specify Location of the Drivers* dialog box opens, illustrated in Figure 17.

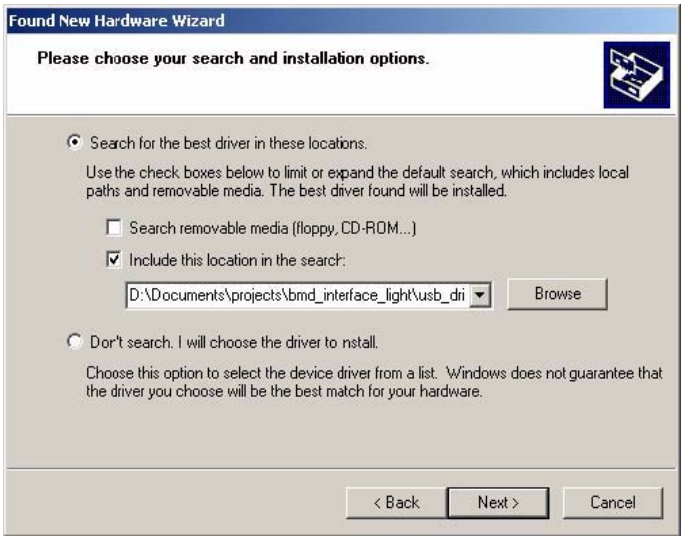


Figure 17. Specifying Location of Drivers Dialog Box

In the *Specify Location of the Drivers* dialog box, point the Hardware Wizard to the unzipped Open Source BDM windows driver package, using the *Browse* button. Once the Windows *Specify Location of the Drivers* dialog box is configured with the correct path to the Open Source

BDM windows driver package, select the *Next* button. This will initiate the installation of the Open Source BDM USB required driver and DLL file, ilustrated in Figure 18.

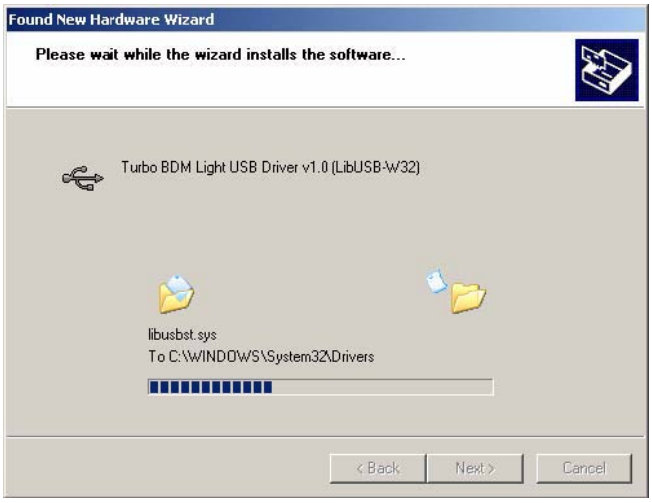


Figure 18. Driver Installation in Progress

Once the installation procedure is completed, the device will be ready to use, illustrated in Figure 19. Select the *Finish* button at this point. Because of the *plug and play* nature of USB, a reboot of Windows is not required.

Figure 19. Finishing Installation Open Source BDM Windows USB Driver

6.3 Configuring the Hiwave Debugger for the Open Source BDM GDI DLL Plug-in

Once the Open Source BDM device is recognized by Windows, the CodeWarrior version 5.0 hiwave debugger is one step closer to programming and debugging targets with the Open Source BDM development tool. The initial release of the CodeWarrior version 5.0 and the hiwave debugger does not have obvious support for the Open Source BDM. A patch is planned for the CodeWarrior version 5.0 Studio, adding more visible Open Source BDM support. This section describes configuration of the hiwave for the Open Source BDM with and without the patch to add Open Source BDM support.

The procedure detailed in this section explains how to configure the hiwave debugger to work with the Open Source BDM interface. Please be certain to download the latest version of the tools from Metrowerks. The debugger interface of older versions do not support the required features. The Open Source BDM was tested with the minimum required debugger version 6.1.

6.3.1 Operation Without the CW Service Patch

Even without the Open Source BDM patch for the CodeWarrior HCS08 Studio, the hiwave debugger can be configured to select the OpenSourceBDM GDI DLL using the *set gdi* command in the Debugger command window, illustrated in Figure 20.

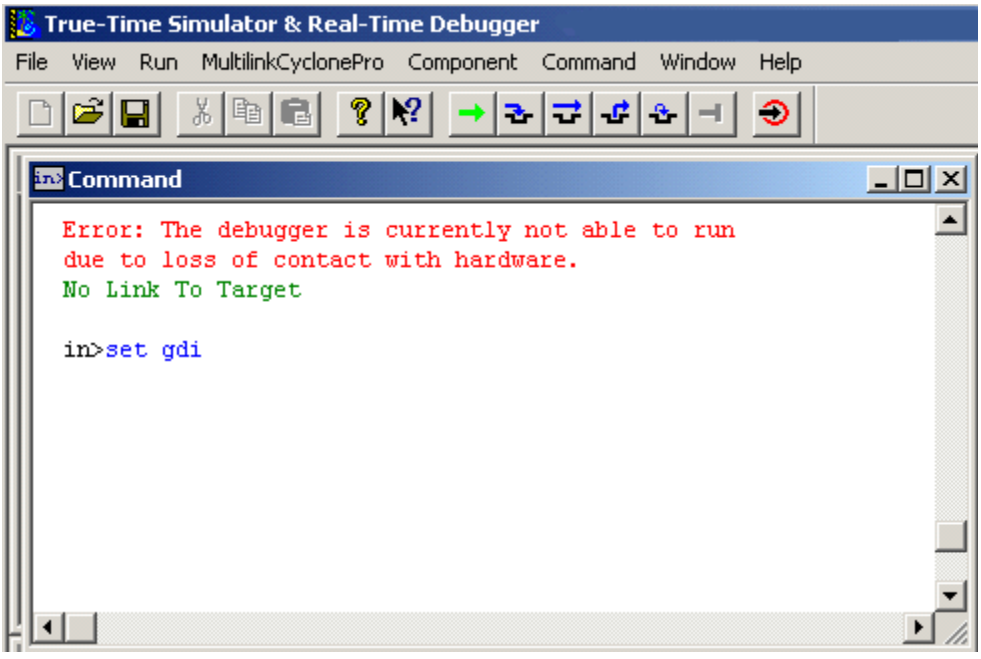


Figure 20. *set gdi* Command

When the *set gdi* command is executed, the *GDI Setup DLL* dialog box opens. To use the Open Source BDM, select the *Browse* button to choose the required OpenSourceBDM_gdi.dll file. The *GDI Setup DLL* dialog box is illustrated in Figure 21.

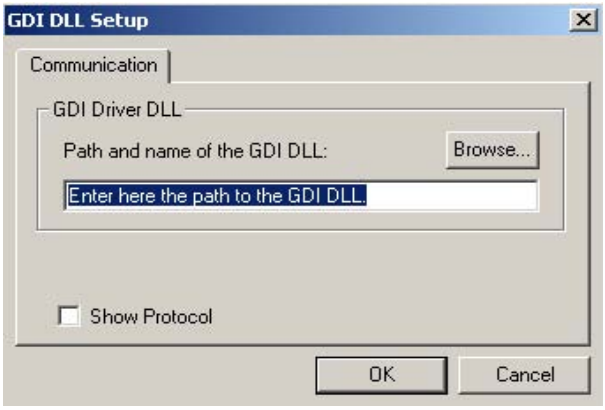


Figure 21. GDI Setup DLL Dialog Box

Sometimes the "set gdi" command does not force a change of the GDI DLL setup (note the active GDI setup is shown in the debugger menu, i.e. Open Source BDM menu option vs. MultilinkCyclonePro vs. MONITOR-HCS08 vs. Softec-HCS08). This issue occurs when the Softec-HCS08 debugger target is selected in the Code Warrior IDE. The solutions is to NOT to select the Softec-HCS08 debugger target (select P&E instead). When a service patch for Code Warrior to add support for the Open Source BDM is avaiable, the set gdi command is not necessary since the Open Source BDM target will be selectable from the Code Warrior IDE.

6.3.2 Operation With the CW Service Patch

In this case, when a project is created, it can be created based on the Open Source BDM connection. Figure 22 illustrates the *Connection Option* dialog box. CodeWarrior provides a Open Source BDM option in this dialog box.

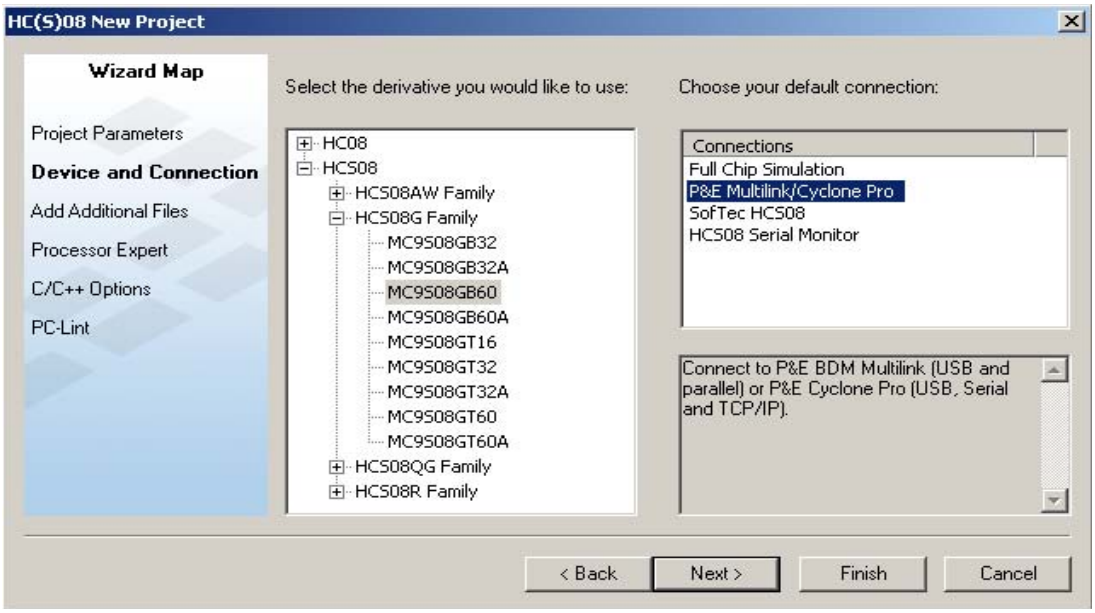


Figure 22. New Project Debugger Interface Connections Dialog Box

Alternatively, if the project was built with other connection using the *New Project Debugger Interface Connections* dialog box, it can be easily changed in the CodeWarrior IDE project manager, illustrated Figure 23. Also, the *set gdi* command can still be used in the hiwave debugger.

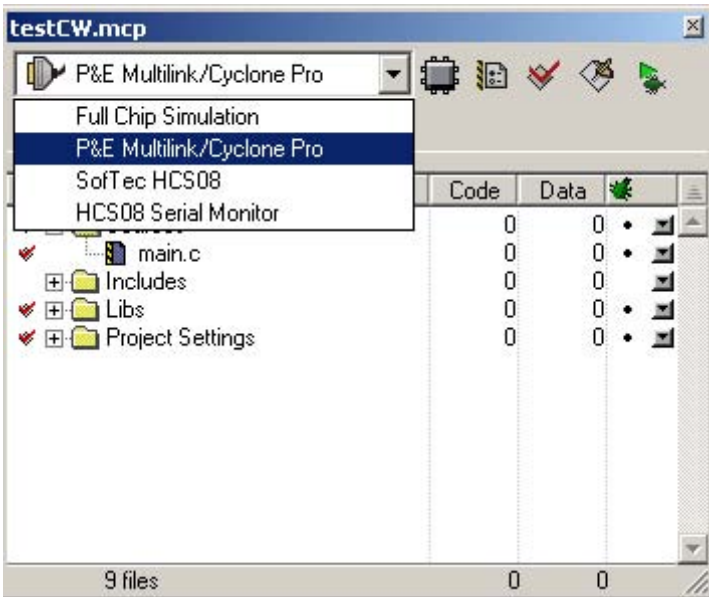


Figure 23. Project Manager Connection Options

6.3.3 Hiwave Debugger Options with Open Source BDM

Once the OpenSourceBDM GDI DLL is configured and the debugger is open, the hiwave debugger and OpenSourceBDM GDI DLL provide full featured functionality provided below. The folowing options are found in the OpenSourceBDM menu:

- Show status dialog box
- Reset to normal command option
- Select derivative option
- Detection and indication of target frequency changes
- Auto derivative selection

Otherwise, operation for the hiwave debugger remains unchanged from other debugger interfaces. Figure 24 illustrates the hiwave program opened and configured for the Open Source BDM debugger interface, shown by the hiwave menu bar with menu entry call Open Source BDM.

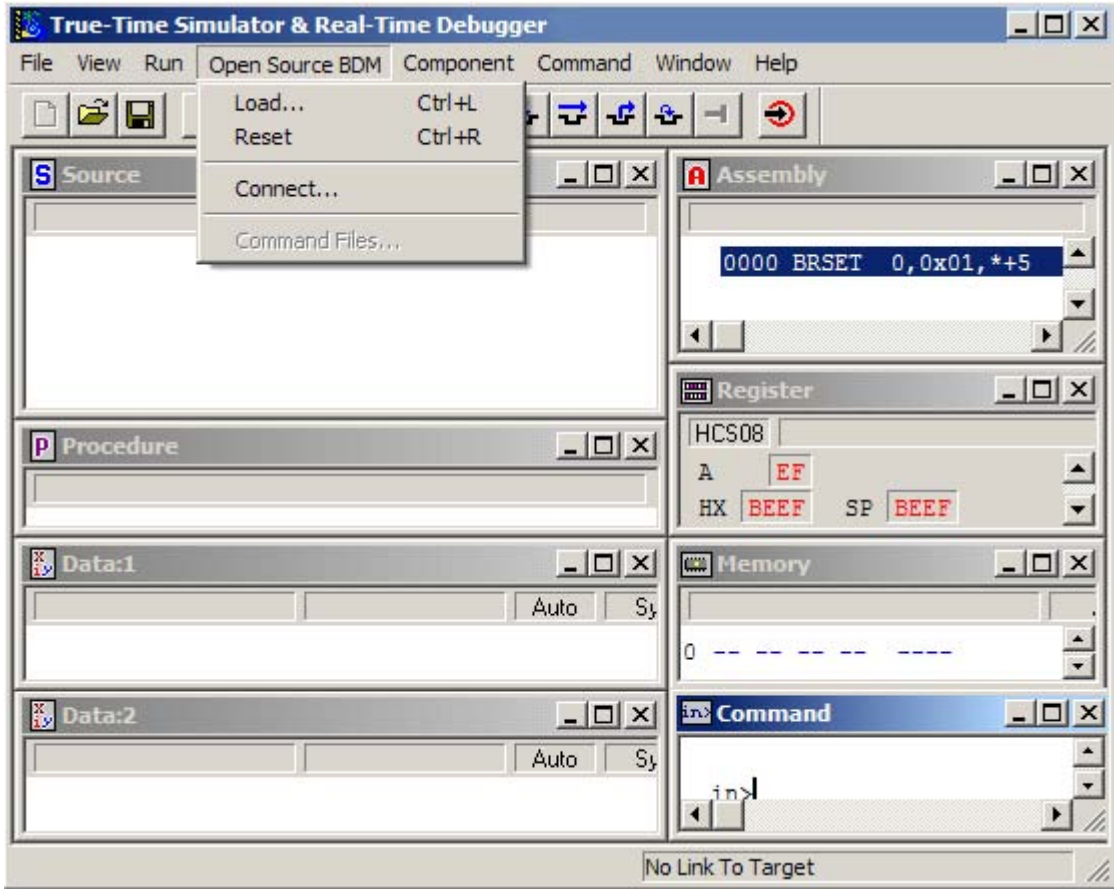


Figure 24. Hiwave Program Opened and Configured

6.4 Programming and Erasing Flash with Open Source BDM

Using the Open Source BDM is no different than using other debugger development tools. The basics steps for configuration and operation of the Open Source BDM for programming and debugging a target system are:

1. Build or otherwise obtain the Open Source BDM PCB
2. Configure the debugger hardware for the Open Source BDM PCB. See "Open Source BDM PCB HCS08 Configuration," Section 6.1.1
3. Connect the debugger hardware as shown in Figure1
4. If necessary, install the debugger hardware windows drivers. See "Installing Windows Open Source BDM DDL and USB Drivers," Section 6.1
5. Develop a user application and select the desired debugger interface in the CodeWarrior IDE. See "Configuring the Hiwave Debugger for the Open Source BDM GDI DLL Plug-in" Section 6.3
6. Execute the command to open the hiwave debugger, making certain the correct debugger interface is selected. Open Source BDM. See step five above.
7. The hiwave debugger opens and erases, then programs the target system. A manually selected file can also be used to upload into the target system. See "Step-by-step Instructions: Flash Programming with the Hiwave Debugger," Section 6.4.2
8. Debug the code using the hiwave debugger. Users have access to all debugger functions including breakpoints, trace, program halt/go, program step/step-over/step-into, and many other debugger command options.

Figure 25 illustrates the cable connections between the PC, the Open Source BDM, and the user target. Detailed information is provided below about using the CodeWarrior IDE.

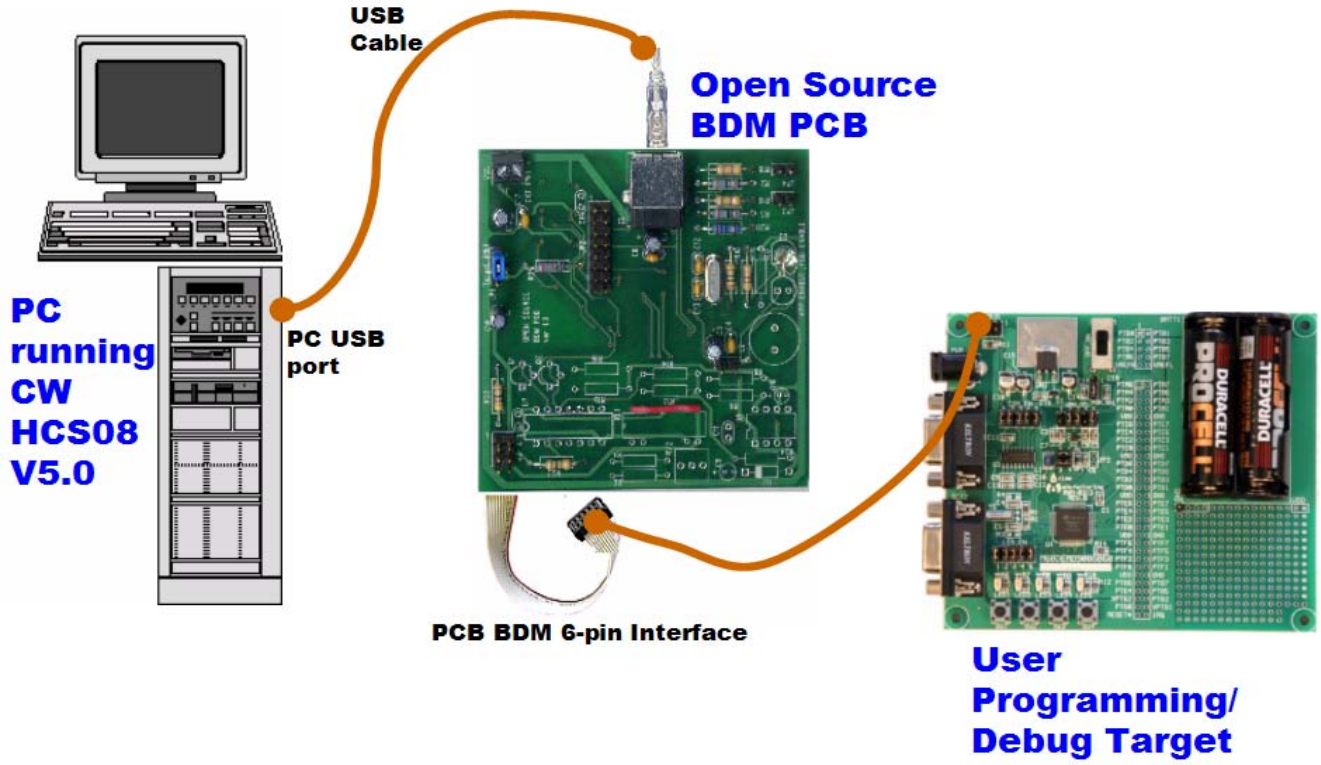


Figure 25. Cable Connections Between the PC and the Open Source BDM

6.4.1 Opening the Hiwave Debugger

Once the main code is added, the project code can be downloaded into the target MCU flash memory. Figure 26 shows pressing the debug icon in the CodeWarrior IDE's program manger window will initiate the programming of the target MCU flash memory. This icon executes a command to open the hiwave program.

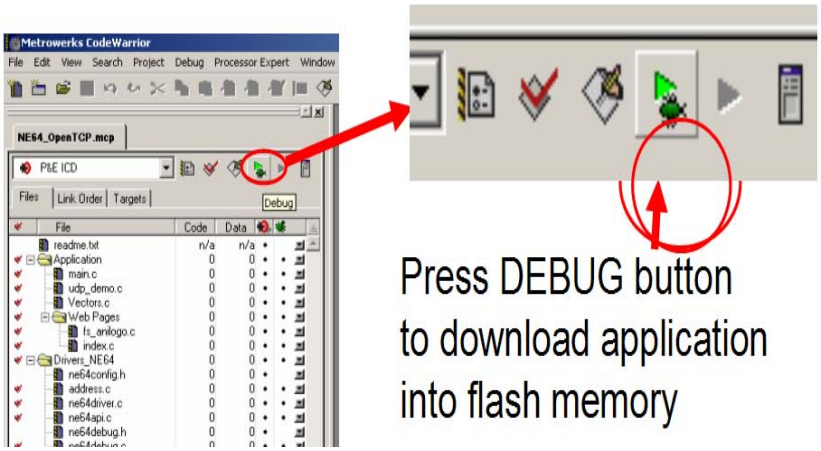


Figure 26. Code Warrior IDE Debug Icon

Figure 27 illustrates the hiwave program again. Selecting the *Load...* or *Flash* command in the Open Source BDM menu initiates flash programming algorithms to erase and reprogram the target MCU. The *Load...* menu command will first allow a selected file to upload during the flash programming operation.

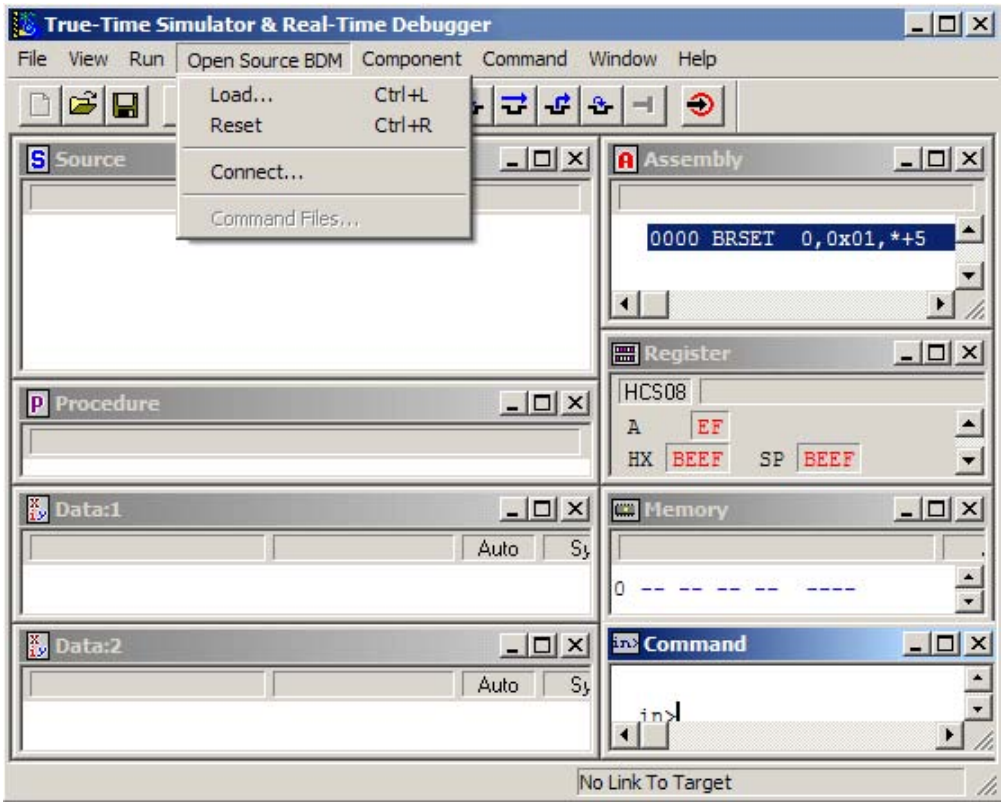


Figure 27. Code Warrior Programmer/Debugger Interface

6.4.2 Step-by-Step Instructions: Flash Programming with the Hiwave Debugger

This sections provides detailed step-by-step instructions for flash programming with the Hiware debugger. From the Open Source BDM menu, both the "Flash" and "Load" command can be used to flash program the target.

6.4.2.1 Using the Load Command:

1. Select Open Source BDM > Load... from the menu
2. When the load command is executed, the "Load Executable File" dialog box opens.

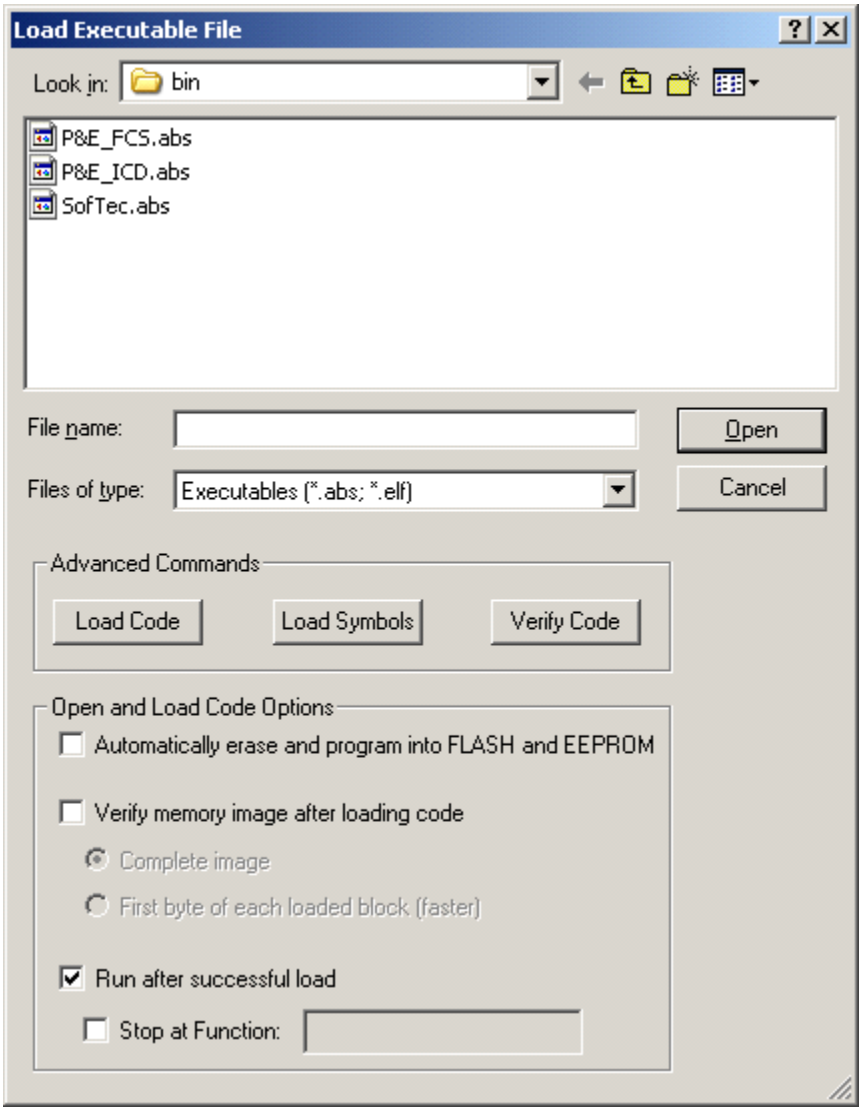


Figure 28. Load Executable File Dialog Box

3. The user must navigate to the file that will be used to program the part and select it
4. Before pressing the "Open Button," the "Automatically erase and program into FLASH and EEPROM" checkbox must be checked

To make the "Automatically erase and program into FLASH and EEPROM" option the default setting for a project, the user must configure the debugger accordingly. These steps are provide below:

1. Select File > Configuration from the menu
2. The "Preference" dialog box opens
3. Select the "Load" Tab

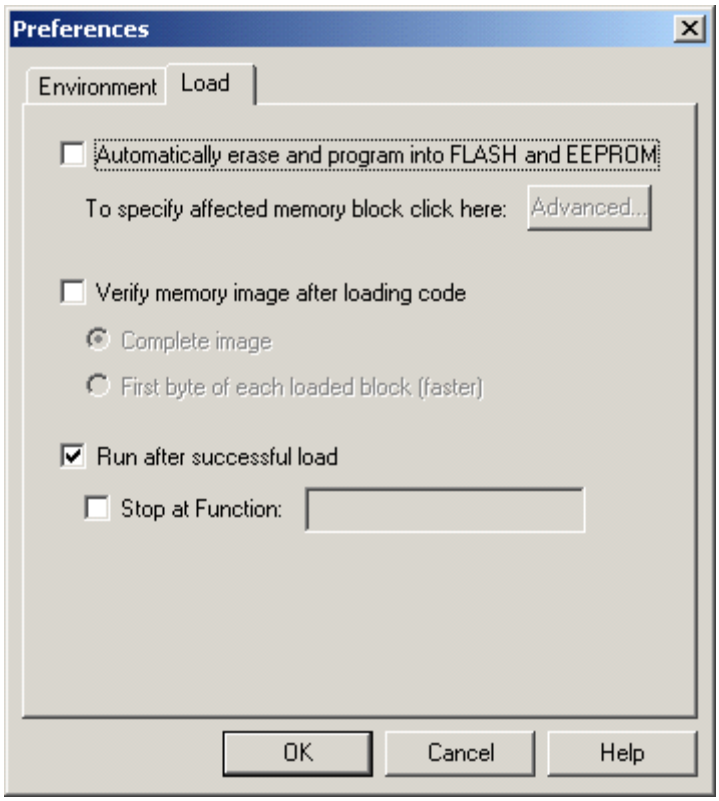


Figure 29. Preferences Dialog Box

4. Check the "Automatically erase and program into FLASH and EEPROM" checkbox
5. Close "Preference" dialog box by pressing the "OK" button
6. Select File > Save Configuration from the menu
7. Next time, auto erase and flash functions will be performed by default

6.4.2.2 Using the Flash Command:

1. Select Open Source BDM > Flash... from the menu
2. The "Non Volatile Memory Control" dialog box opens

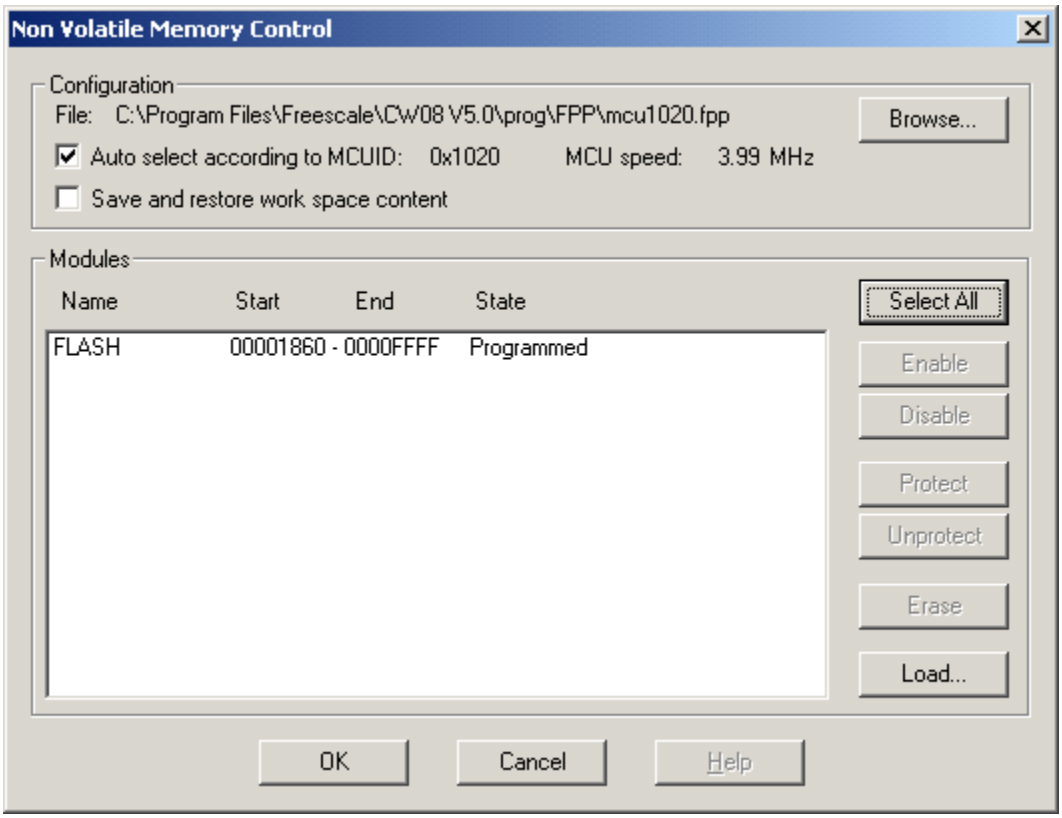


Figure 30. Non Volatile Memory Control Dialog Box

3. Press the "Select All" button
4. Press the "Erase" button
5. Press the "Load..." button
6. When the load button is pressed, the "Load Executable File" dialog box opens.
7. The user must navigate to the file that will be used to program the part and select it
8. Pressing the "Open" button programs the part

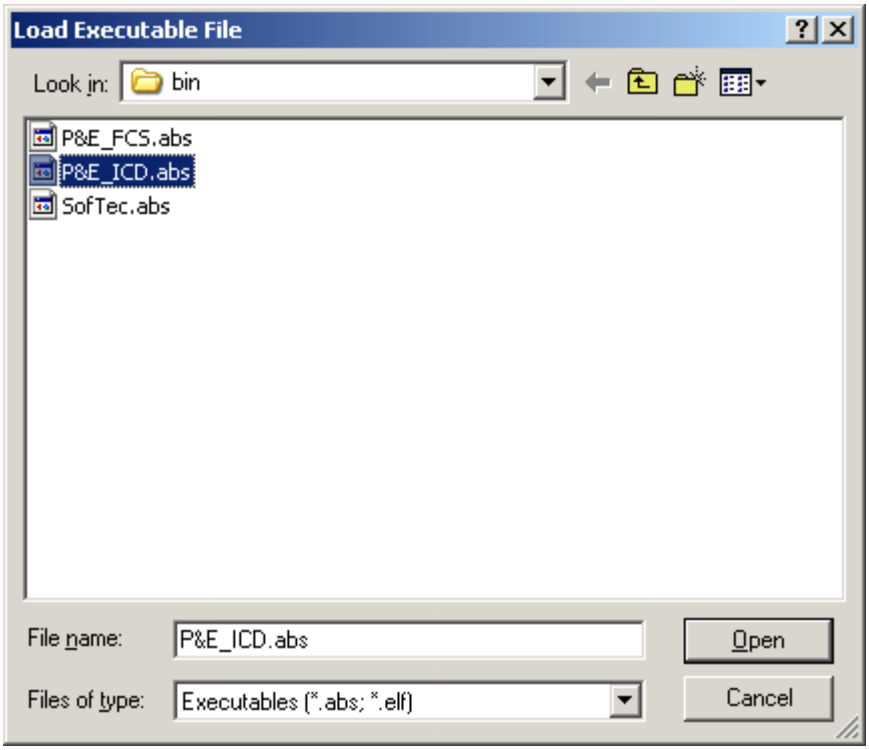


Figure 31. Load Executable Dialog Box

9. To close the "Non Volatile Memory Control" dialog box, press the "Unselect All"
10. Then press the "OK" button"

7.0 References

1. LIBUSB documentation, http://libusb.sourceforge.net/
2. Data sheet to HC908JB16, MC68HC908JB16.PDF available from Freescale
3. Documentation to Generic Debugging Interface, available from Tasking, www.tasking.com
4. TBDML HCS12 project, see either http://forums.freescale.com or www.freegeeks.net