Multi-Button IR Remote Control using the MC9RS08KA2

Designer Reference Manual

RS08 Microcontrollers

DRM081
Rev. 0
6/2006

freescale.com

freescale semiconductor

Multi-Button IR Remote Control using the MC9RS08KA2

Designer Reference Manual

by: T.C. Lun
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)
06/2006	0	Initial release	N/A

Table of Contents

Chapter 1
Introduction

1.1 Introduction

This document describes a reference design for an infrared (IR) remote control (RC) solution using the MC9RS08KA2 microcontroller.

For many air conditioner and small home appliance applications, there is a need for a wireless user interface such as a remote control unit to send data from a transmitter to a receiver using infrared communication. The basic requirements of an IR remote control unit are: lower power consumption in standby mode; low operating voltage; low system cost; and easy code modification for customizing to different models.

This reference design includes both the transmitter and the receiver unit. In this document, the focus is to show the use of the 6-pin DFN packaged MC9RS08KA2 microcontroller unit (MCU) for the IR remote control transmitter unit. For details on the receiver, please refer to the designer reference manual, Freescale document, *DRM082 – Infrared remote control using the MC68HC908LT8*.

A feature of this reference design is a 6-pin BDM interface header for in-circuit FLASH programming and debugging in the remote control transmitter.

1.2 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 remote control applications. This device is composed of standard on-chip modules including a very small and highly efficient RS08 CPU core, 62 bytes RAM, 2 Kbytes 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

1.3 Reference Demo Board

The remote control transmitter reference board has the following features:

- 9-button remote controller with ultra low cost MC9RS08KA2 MCU
- 38kHz carrier frequency generated by software delay
- Easy re-programming and debugging by 6-pin BDM interface
- Low operating voltage down to 1.8V
- Low power consumption in standby mode, typically 1μA[1]

Figure 1-1 shows the transmitter and receiver unit of the IR remote control reference design.

1. The power consumption is dependant on application and system requirements. The 1μA assumes that all modules are turned off except the internal clock source (ICS).

(a) MC9RS08KA2 IR RC Transmitter **(b) MC68HC908LT8 IR RC Receiver**

Figure 1-1. Infrared Remote Control Reference Design

Chapter 2
Fundamentals of IR Remote Control Communication

2.1 Configuration of the IR Remote Control Unit

An IR remote control transmitter generates infrared rays to a receiver by ways of a digital control frame pattern. The infrared transmitting diode and the infrared receiving module are important components for an efficient IR transmission through air. The carrier frequency for home appliance applications is typically around 38 kHz.

A typical configuration of IR remote control is shown in Figure 2-1.

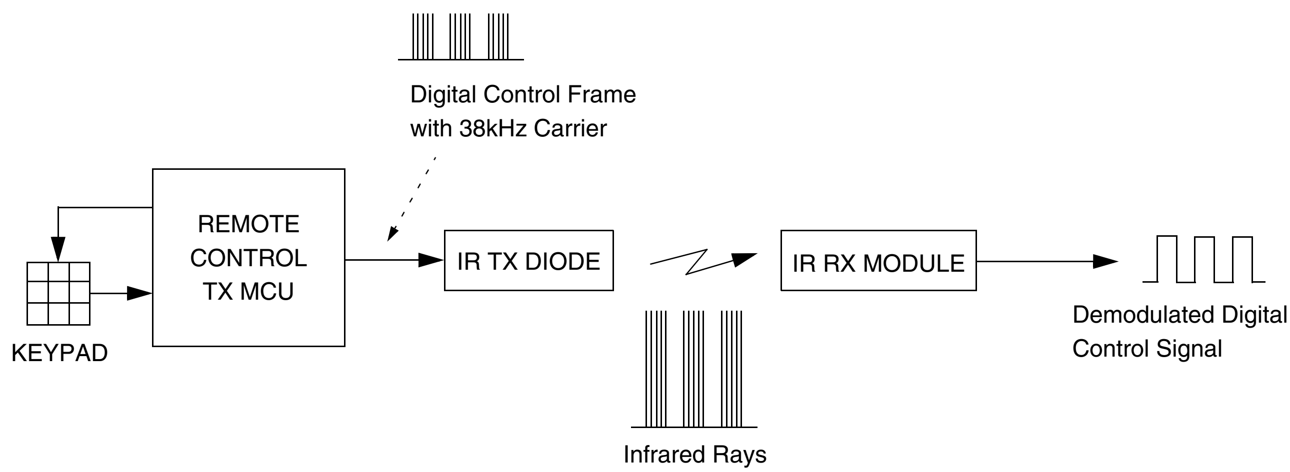


Figure 2-1. Configuration of IR Remote Control Unit

2.2 Control Frame Format

The IR control frame pattern is specific for different transmitter-receiver designs. It depends on application requirements such as controller purpose and features. Figure 2-2 shows the typical example of the control frame waveform that is used in this IR remote control reference design.

In Figure 2-2, the carrier is the 38 kHz with a 1/3 duty cycle. Having IR transmitting diode using 38 kHz carrier and 1/3 duty cycle allows a low power design for the IR transmission. If the carrier was 1/2 duty, the transmitting diode will be on for 13 μs and off for 13 μs. But for 1/3 duty, the diode is on for 8 μs and off for 18 μs. A reduction in turn on time means a reduction in power consumption.

The data bit for 0 or 1 is based on the duration of the carrier on/off. For data 0, both carrier on and off times are 0.5ms. For data 1, the carrier on time is 0.5ms and the carrier off time is 1.5ms.

Typically, the data frame consists of the header code, several bytes of data code, one byte of customer code, and one stop bit. The header code is used to indicate to the IR receiver that following transmissions will be the data code and customer code. The data code is used for control purposes, such as on/off, increase/decrease, modes, etc. The customer code is used for identifying different customers. And the stop bit is to indicate it is the last bit of the current transmission.

In this reference design, the above frame format is used for an air conditioner remote control unit.

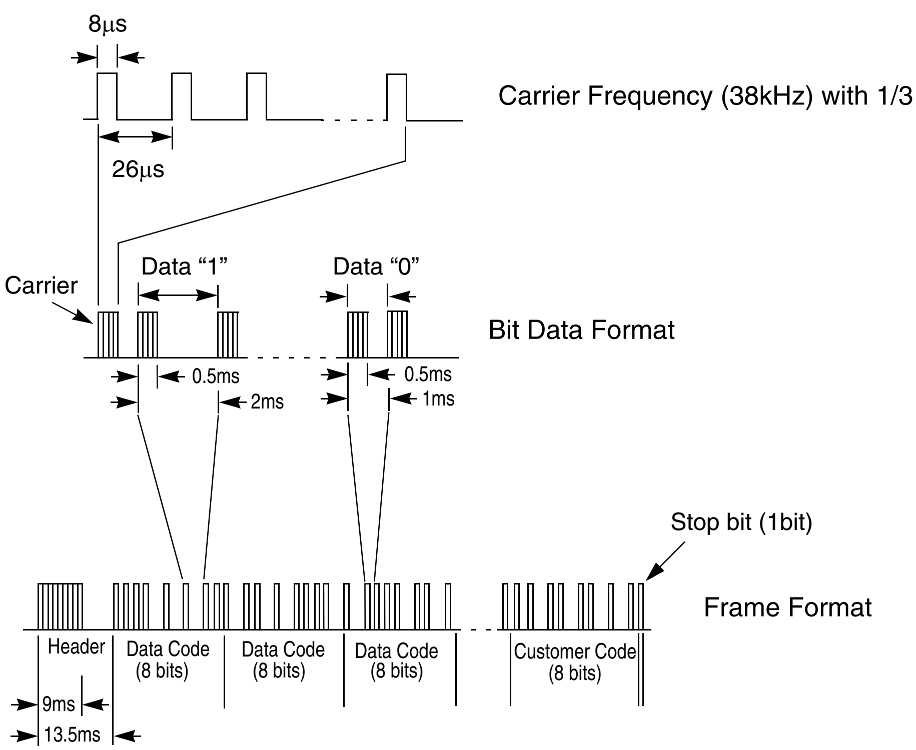


Figure 2-2. Control Frame Waveform

Chapter 3
System Concept

3.1 System Specification

This reference design demonstrates a remote controller for air conditioner/small appliance applications with re-programming and debugging features. The design meets the following performance specifications:

- Low power consumption in standby mode
- Low operating voltage
- 6-pin BDM interface for software development
- MC9RS08KA2 transmitter and MC68HC908LT8 receiver for system evaluation in real time
- Transmitter and receiver use standard type-AAA batteries as power source

Figure 3-1(a) shows the front of the transmitter unit with the 9-key keypad. Figure 3-1(b) shows the back of the transmitter unit with the BDM interface header and battery cover.

Figure 3-2(a) shows the front of the receiver unit, with the key switch, LCD and LED display, and the IR receiver module. Figure 3-2(b) shows the back of the receiver unit, with the MON08 interface header, battery holder, and ON/OFF switch.

3.2 Application Description

The design uses the MC9RS08KA2 in the transmitter unit and the MC68HC908LT8 in the receiver unit.

In the transmitter unit, the MC9RS08KA2 performs the following tasks:

- Keyboard scanning
- Frame encoding
- Carrier generating
- Transmitting the encoded frame to IR with carrier

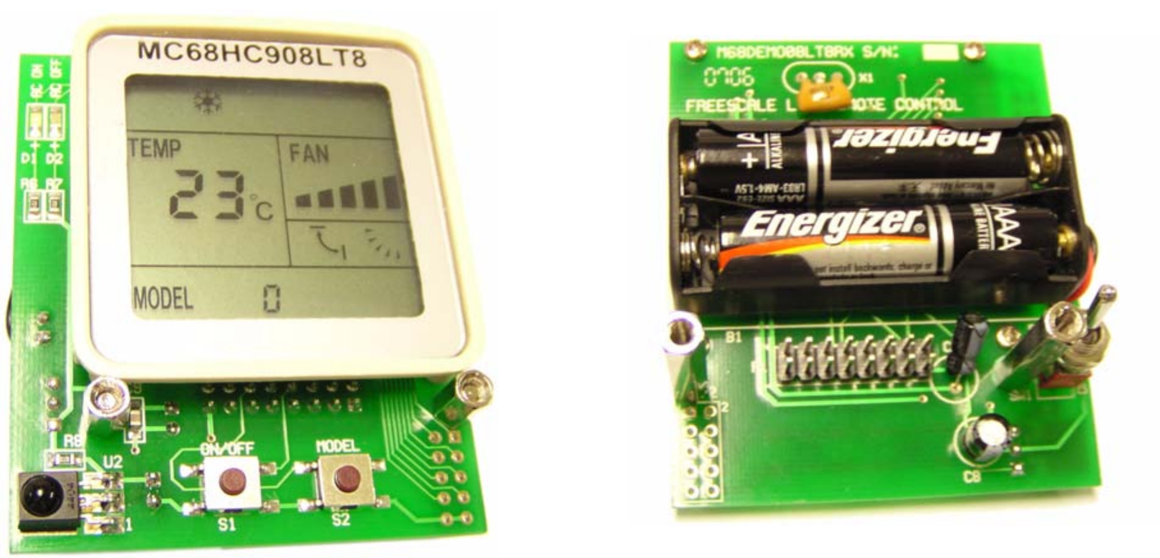
In the receiver unit, the MC68HC908LT8 performs the following tasks:

- Keyboard scanning
- Frame decoding
- LCD and LED displaying

This document covers the MC9RS08KA2 transmitter unit only.

(a) Front of IR RC Transmitter **(b) Back of IR RC Transmitter**

Figure 3-1. MC9RS08KA2 IR Remote Control Transmitter Unit



(a) Front of IR RC Receiver **(b) Back of IR RC Receiver**

Figure 3-2. MC68HC908LT8 IR Remote Control Receiver Unit

3.3 Control Process

Since the design is targeted for an air conditioner remote controller application. Some general control parameters must be included, such as power ON/OFF, temperature data, and mode selection.

Table 3-1 summarizes the control data frame definition for this reference design.

Table 3-1. Remote Control Frame Definition

Data Code Name	Bit Definition								Function	Remarks		
	7	6	5	4	3	2	1	0				
C1	0								**A/C OFF**			
	1								**A/C ON**			
		0	0	0					**AUTO mode(1)**	no temp.		no sleep
		0	0	1					**COOL mode(2)**	custom temp.	custom wind	
		0	1	0					**HUMIDITY mode(2)**	custom temp.	custom wind	
		0	1	1					**WIND mode(2)**	custom temp.	custom wind	no sleep
		1	0	0					**HEAT mode(3)**	custom temp.	custom wind	
					0	0			**°C**			
					0	1			**Reserved**			
					1	0			**°F (Lower range)**			
					1	1			**°F (Higher range)**			
							0		**Light ON**			
							1		**Light OFF**			
								X	**Reserved**			
C2					0				**Sleep OFF**			
					1				**Sleep ON**			
						0			**Swing OFF**			
						1			**Swing ON**			
							0	0	**AUTO Wind Speed**			
							0	1	**LOW Wind Speed**			
							1	0	**MIDDLE Wind Speed**			
							1	1	**HIGH Wind Speed**			
									Temperature	°C	°F	°F
										C1[3:2] = 0:0	C1[3:2] = 1:0	C1[3:2] = 1:1
	0	0	0	0						15°C	59°F	75°F
	0	0	0	1						16°C	60°F	76°F
	0	0	1	0						17°C	61°F	77°F
	0	0	1	1						18°C	62°F	78°F

Table 3-1. Remote Control Frame Definition (Continued)

	0	1	0	0						19°C	63°F	79°F
	0	1	0	1						20°C	64°F	80°F
	0	1	1	0						21°C	65°F	81°F
	0	1	1	1						22°C	66°F	82°F
	1	0	0	0						23°C	67°F	83°F
	1	0	0	1						24°C	68°F	84°F
	1	0	1	0						25°C	69°F	85°F
	1	0	1	1						26°C	70°F	86°F
	1	1	0	0						27°C	71°F	
	1	1	0	1						28°C	72°F	
	1	1	1	0						29°C	73°F	
	1	1	1	1						30°C	74°F	
									Model[4]			
C3	0	0	0	0					**0**			
	0	0	0	1					**1**			
	0	0	1	0					**2**			
	0	0	1	1					**3**			
	0	1	0	0					**4**			
	0	1	0	1					**5**			
	0	1	1	0					**6**			
	0	1	1	1					**7**			
	1	0	0	0					**8**			
	1	0	0	1					**9**			
					0				**Model Set ON**			
					1				**Model Set OFF**			
						x	x	x	**Reserve**			
C4	1	0	1	0	1	0	0	1	**Customer Code[3]**	Same Model number between transmitter and receiver		

NOTES:
1. Default mode for the reference design after a power-on-reset.
2. Default value of temperature is 25°C and needs to store temperature and wind speed individually.
3. Default value of temperature is 28°C.
4. Same model number for transmitter and receiver.

Since each customer has their own requirements and definitions, Table 3-1 only includes the general and common control parameters. Additional parameters can be added, thus increasing the frame length by the additional control bytes.

Chapter 4
Hardware

4.1 Hardware Implementation

This chapter will focus on the hardware implementation of MC9RS08KA2 transmitter unit. The MC68HC908LT8 receiver unit is covered in the Freescale document, *DRM082 – Infrared remote control using the MC68HC908LT8*; which also outlines an implementation of the transmitter unit using the MC68HC908LT8.

The IR remote control transmitter unit can be divided into the following parts:

- Internal oscillator circuit
- Keypad scan and decode
- IR transmitter diode drive
- BDM interface

4.2 MC9RS09KA2 IR Remote Control Transmitter

The MC9RS08KA2 IR remote controller transmitter unit is mounted on an optimized PCB and fits in an actual remote controller casing, with keypad, battery holder, and a BDM interface header for firmware development and system evaluation.

This reference design uses the 6-pin packaged MC9RS08KA2 to implement the basic functions of the IR remote controller transmitter unit. An 8-pin packaged version can be used if more features and functions are required.

4.2.1 Oscillator Circuit

As the MC9RS08KA2 has an internal clock source (ICS) module, an external crystal is not required to generate the clock for the device. The 6-pin packaged device has enough pins to implement a 9-key IR remote controller transmitter unit. The ICS in the KA2 is a RC oscillator with a maximum frequency of 20 MHz (10 MHz bus) and an accuracy of ±2% after trimming. This ±2% accuracy is sufficient for IR remote control applications.

4.2.2 Keypad Scanning

Although MC9RS08KA2 does not have a built-in analog-to-digital converter (ADC), an ADC function can be emulated using its built-in comparator. Together with a resistor network, for different voltages, the comparator can be used to detect the different buttons being pressed. The technique for emulated ADC on the MC9RS08KA2 is discussed in Freescale document, *AN3266 — Getting Started with RS08*.

From Figure 4-1, the idea is to decode the 9-button keypad using the keyboard interrupt (KBI2) and the comparator module (ACMP+ and ACMP–) emulated as ADC. The 9-button keypad is implemented using contacts on the printed circuit board (PCB) and a 9-button membrane with tactile domes for closing the contacts on the PCB. The switch contacts on the PCB are designed in a way to provide the necessary separation between the KBI2 and ACMP– pins.

Pressing a button connects KBI2 to ground, and hence causes a keyboard interrupt on the MCU. At the same time, the resistor divider for the button also connects to ground, and hence causes a defined voltage potential on the ACMP– pin. When a KBI occurs the RC network (R11 and C3) on ACMP+ starts charging. Which button is pressed is determined by counting the time for the charge voltage on ACMP+ to equal the resistor divider voltage on ACMP–.

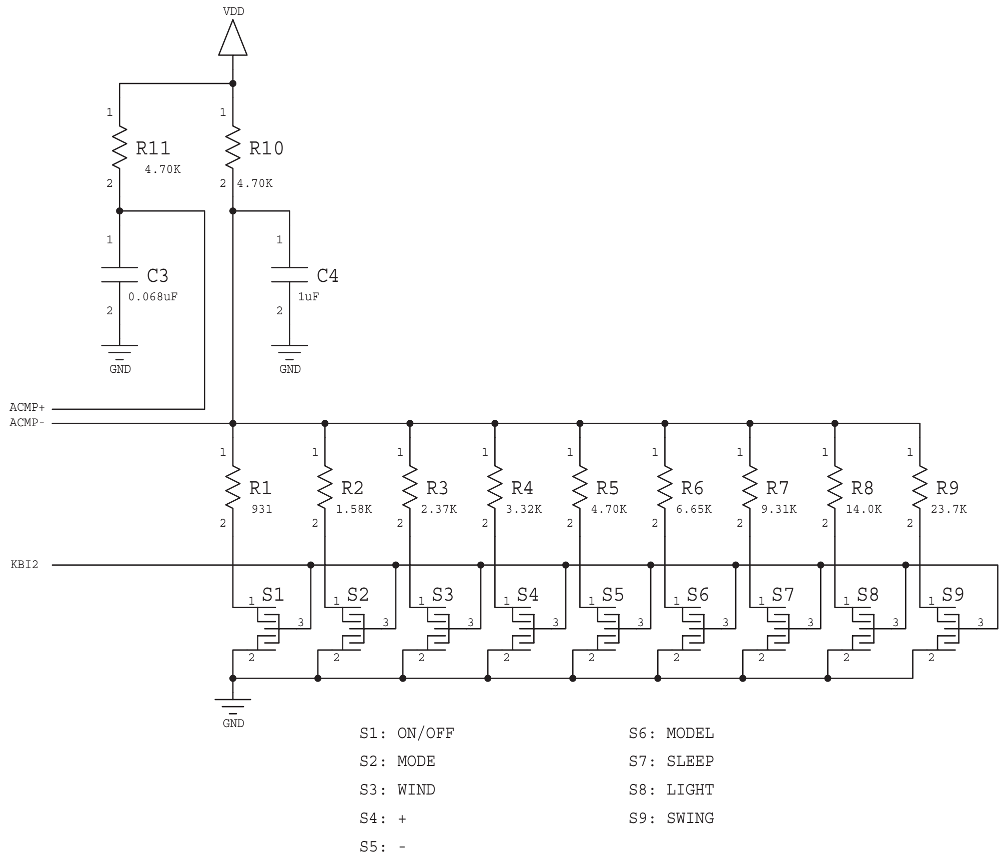


Figure 4-1. Transmitter 9-Key Circuit

There are nine buttons on the transmitter unit. The function of each button is summarized Table 4-1.

Table 4-1. Buttons on the IR Remote Control Transmitter Unit

Button	Function
S1	This is the ON/OFF button. Pressing S1 toggles the air conditioner power on and off. When the receiver is in the OFF mode, the OFF LED will be on, ON LED is off, and the LCD will be off. When the receiver is in the ON mode, the ON LED will be on, OFF LED is off, and the LCD will be on.
S2	This is the mode selection button. Pressing S2 toggles through the operating modes of the air conditioner: AUTO → COOL → HUMIDITY → WIND → HEAT and back again to AUTO (see Table 3-1). The corresponding icon on the receiver LCD is activated accordingly (see 4.3 LCD and LED Display in Receiver).
S3	This is the WIND speed selection button. Pressing S3 toggles through the wind speeds of the air conditioner: AUTO → LOW → MIDDLE → HIGH and back again to AUTO (see Table 3-1). The default setting is AUTO when the air conditioner is switched from off to on. The corresponding icon on the receiver LCD is activated accordingly (see 4.3 LCD and LED Display in Receiver).
S4	This is the “+” or increase button for temperature setting. Each S4 press increases the temperature by one Degree or Fahrenheit.
S5	This is the “–” or decrease button for temperature setting. Each S5 press decreases the temperature by one Degree or Fahrenheit.

Table 4-1. Buttons on the IR Remote Control Transmitter Unit

Button	Function
S6	This is the model selection button. Since there is no display on the transmitter, this key performs no function on this reference design (the model number is fixed to model #0).
S7	This is the sleep mode button. Pressing S7 activates the sleep timer and turn off the receiver LCD. The air conditioner switches off when the sleep timer expires. The actual sleep timer is not implemented on this reference design.
S8	This is the LCD backlight on/off button. Pressing S8 toggles the backlight on the receiver LCD on and off. In this reference design, this button actually toggles one of the receiver LCD icons on and off.
S9	This is the air swing selection button. Pressing S9 toggles the air conditioner louver air swing on the off. The corresponding icon on the receiver LCD is activated accordingly (see 4.3 LCD and LED Display in Receiver).

4.3 LCD and LED Display in Receiver

Figure 4-2 shows the LED and LCD display on the MC68HC908LT8 IR remote control receiver unit.

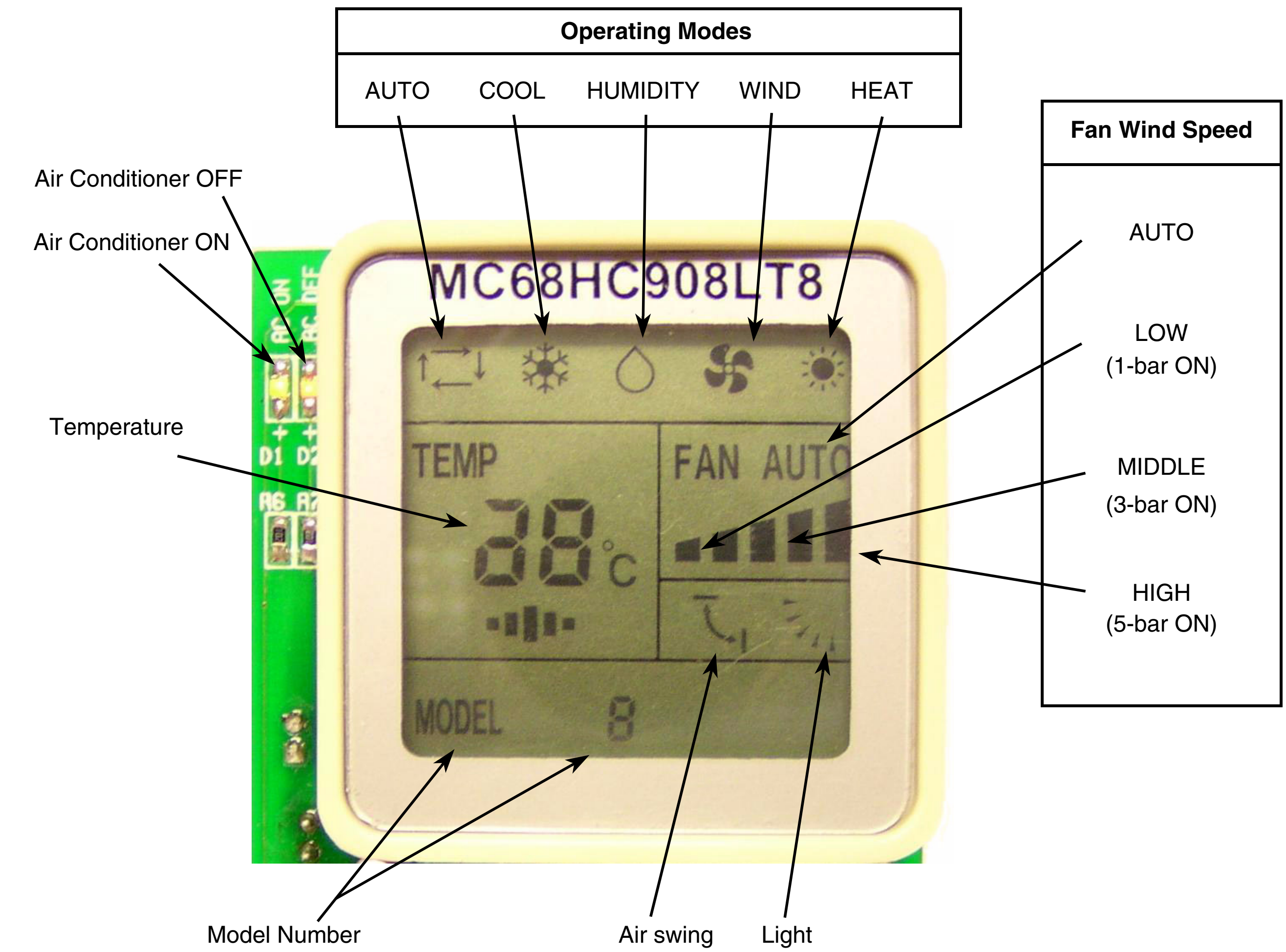


Figure 4-2. Typical Air Conditioner Receiver LCD Display

4.4 IR Transmitter Diode Drive

To keep system cost low, the MC9RS08KA2 MCU drives the IR transmitting diode directly. Figure 4-3 shows the typical drive circuit for the IR transmitting diode.

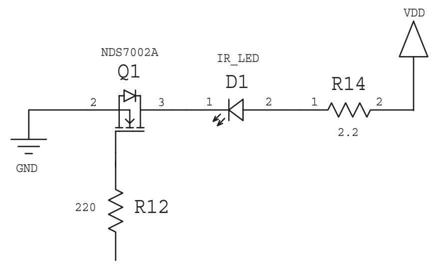


Figure 4-3. Circuit for IR Transmitting Diode

The circuit uses PTA3 to drive the IR transmitter diode. The timer overflow and software delay are used to generate the 1/3 duty cycle, 38kHz waveform and all data bits of the transmitter control frame. When PTA3 is at logic high, the IR transmitter diode is on. R14 is a current limiting resistor for the IR transmitter diode. The value of R14 depends on the requirement of output power in the IR transmitting diode. Lowering the value of R7 will increase the output power of the IR transmitter diode. Also, the output power of the IR transmitter diode can be changed by altering the duty cycle of the output PWM signal.

4.5 BDM Interface Header

For easier reprogramming of FLASH and evaluating purposes, a 6-pin BDM header is included in this reference design. The BDM interface provides in-circuit programming and debugging features.

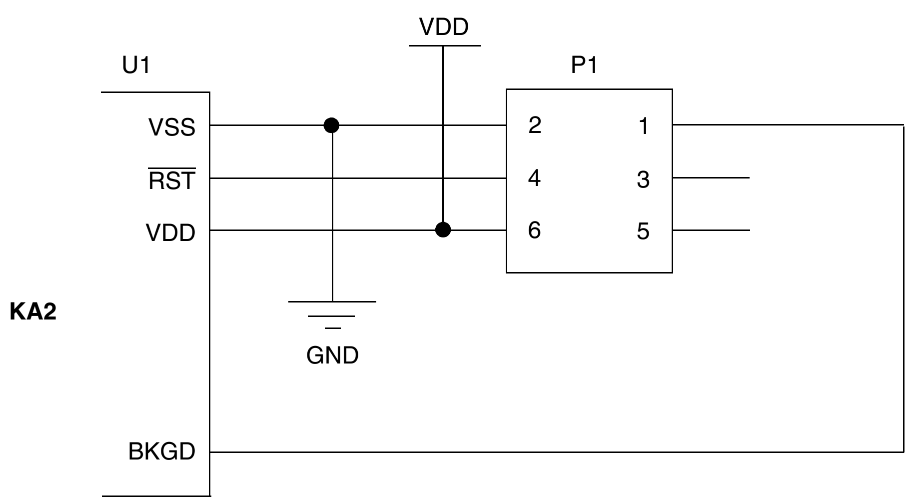


Figure 4-4. BDM Interface Circuit

Figure 4-4 shows the connection between the BDM header (P1) with MC9RS08KA2 in the transmitter unit. To reprogram or debug code, just connect the hardware interface board between the PC and the BDM header. For normal transmitter unit operation, pins 3 and 4 are shorted, and the $\overline{\text{RST}}$ pin becomes the KBI2 pin.

Chapter 5
Software Design

5.1 Introduction

This chapter describes the software design for the IR remote controller reference design. This includes outlines for the following:

- General flow chart
- Transmitter software implementation

5.2 Transmitter Flow Chart

The control algorithm of the remote control transmitter is shown in Figure 5-1. Detail processes in the code are explained in the following sections. After the remote control transmitter is powered on, the MC9RS08KA2 registers will be initialized, such as the I/O ports, timer, and keyboard interrupt modules. After the register initialization, the keyboard interrupt is enabled, ready to detect any button press. If no button is pressed on the transmitter unit, the MC9RS08KA2 will enter stop mode for power saving. In stop mode, all MCU modules are turned off. If a button is pressed, the MCU will wakeup from stop mode and then determine which button has been pressed using the emulated ADC method. Once the button pressed is determined, the control data frame is updated accordingly. After that, the control frame will be transmitted by the IR transmitter diode. Once the button is released, the code will jump back to the beginning and wait for a button press again.

5.3 Transmitter Software Implementation

This section discusses the transmitter software implementation in details.

5.3.1 Initialization

After transmitter power on, the following are initialized on the MC9RS08KA2 MCU:

- ICS trimmed to 16-MHz, $\pm$ 2%
- Clear COP counter
- Set configuration registers
  - Disable COP and RTI
  - Enable LVI
  - Disable IRQ
- Initialize system variables
- Initialize control frame data
- Initialize GPIO A modules
  - All GPIO as outputs low except PTA2
  - sets A0 and A1 to logic high
- Initialize timer module
  - Set clock source to bus frequency divide-by-64
- Enable KBI2 pin for falling-edge trigger interrupts

After initialization, the main routine will enter and remain in stop mode for power saving (stop I_{DD} = 1μA) until a button is pressed. On wake up, the button is software debounced and decoded, the control frame updated, and the frame transmitted out. After the IR signal is transmitted, the system will return to stop mode again, ready to detect a button press.

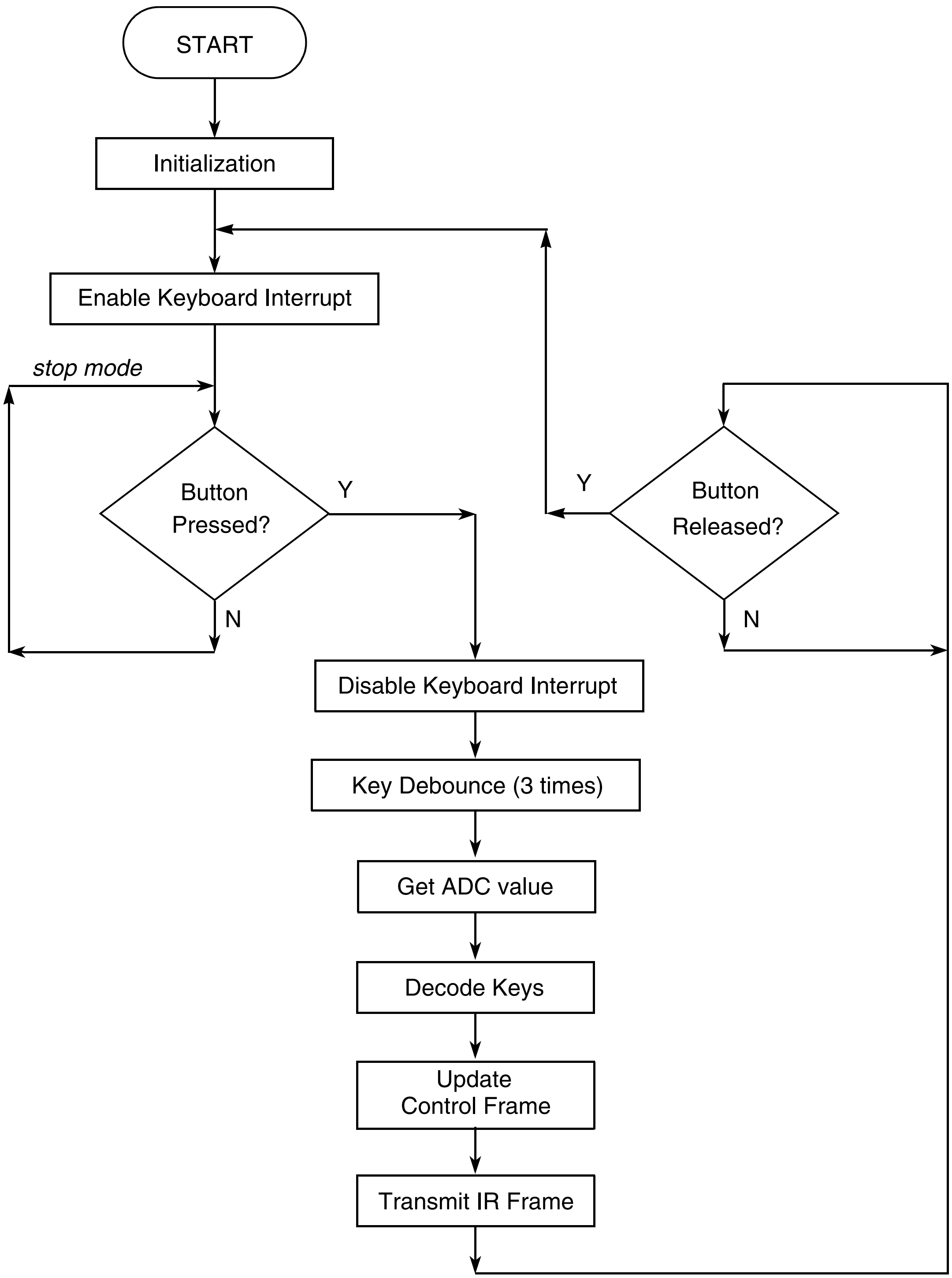


Figure 5-1. General Flow Chart of Transmitter

5.3.2 Key Decoding

The flow chart in Figure 5-1 shows that when a button is pressed, the system will wake up from stop mode. After that, key decoding is performed. The detail operation of the key decoding is shown in the flow chart of Figure 5-2. When the system wakes up, the keyboard interrupt will be disabled and key debounce performed to eliminate the noise that may trigger a wrong key pressed. After key debounce, key decoding is performed to determine which button is pressed. This reference design uses an emulated ADC to find out which key is pressed. When a key is pressed, both comparator pins (ACMP+ and ACMP–) and the timer will be enabled and timer count starts. The voltage on the ACMP– pin is dependent on the resistor divider that is connected to the keys on the keypad. The timer counts the period for the voltage on the ACMP+ pin to charge up until it is equal to the voltage on the ACMP– pin. When the two voltages are equal, an interrupt will occur and the timer count is recorded. As each button has a different voltage and hence, different timer counts, the button pressed can be determined.

When the key is identified, the control frame will be updated according the function of the key. The frame is then transmitted out through the IR transmitter diode with 38 kHz, 1/3 duty, cycle carrier.

5.3.3 Transmission Control Frame Update

When the key is identified, the transmission control frame is updated based on the assigned function of the key. The definition of the transmission control frame is shown in Table 3-1 and the definition of the key function are described in 4.2.2 Keypad Scanning.

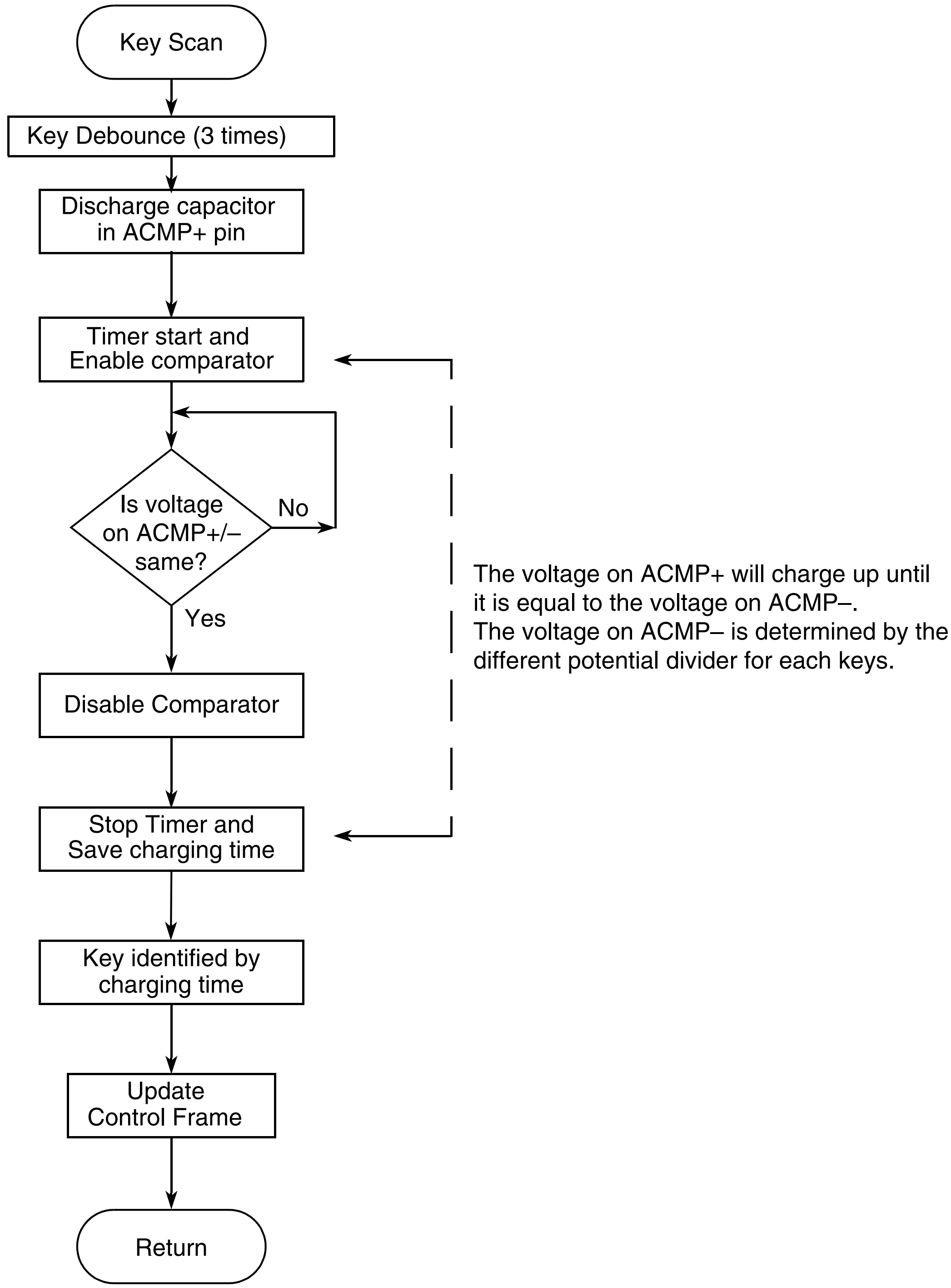
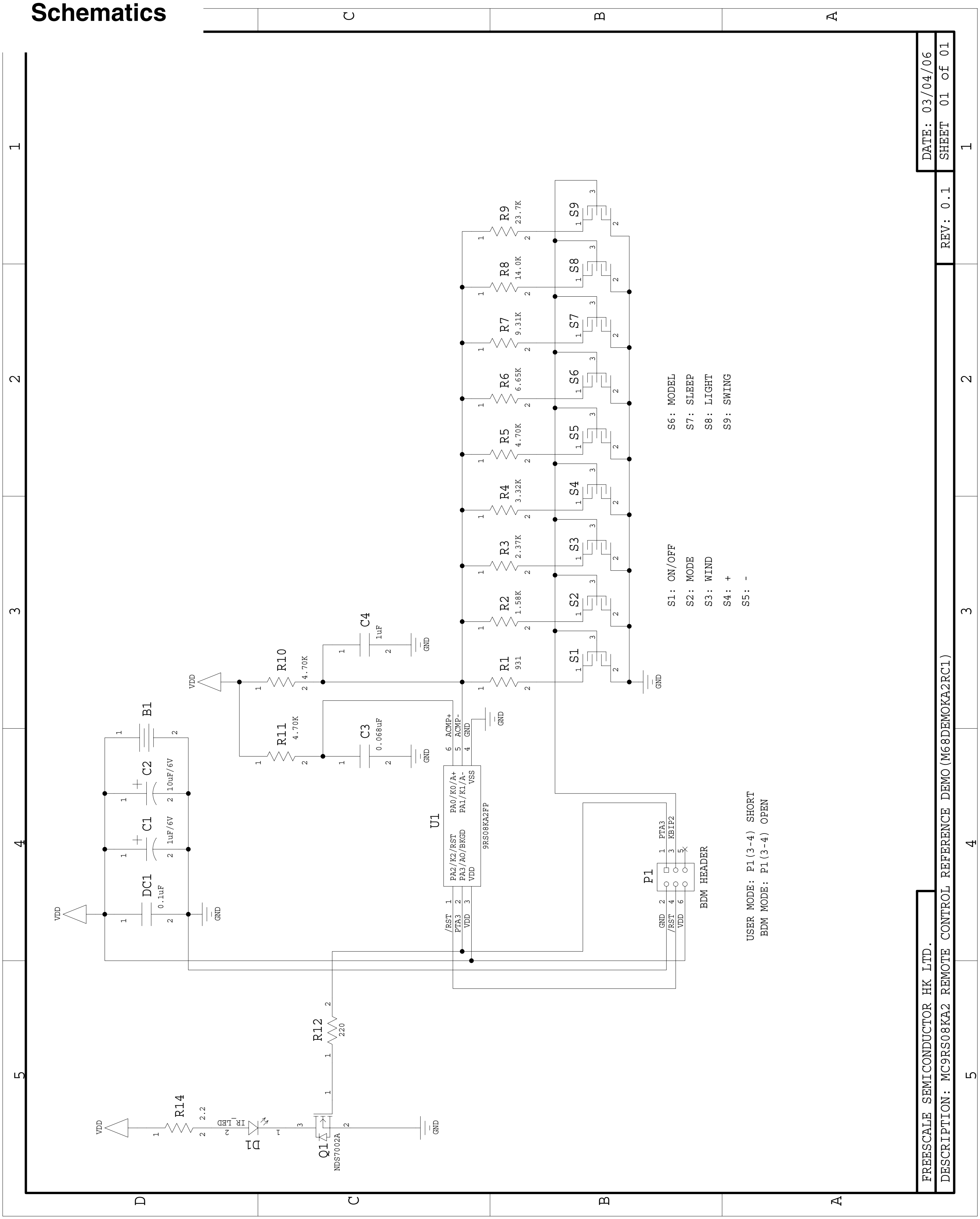


Figure 5-2. Keyboard Decoding in Transmitter

Appendix A. Schematics



Appendix B.
Program Listing

```
;*************************************************************
;
; (c) copyright Freescale Semiconductor. 2006
; ALL RIGHTS RESERVED
;
;*************************************************************

;*************************************************************
;* Remote Control Coding for 9RS08KA2
;*
;* Author:     T.C. Lun
;* Date:       Feb 2006
;*
;* PTA0/KBI0/ACMP+            Keypads input
;* PTA1/KBI1/ACMP-            RC input
;* PTA2/KBI2/TCLK/RESETb/VPP  KBI for S6-S9
;* PTA3/ACMPO/BKGD/MS         Unused
;* PTA4/KBI4                  KBI for S1-S5
;* PTA5/KBI5                  IR output
;*************************************************************
; 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
KEY            equ      PTAD_PTAD2
mKEY           equ      mPTAD_PTAD2
IR             equ      PTAD_PTAD3
mIR            equ      mPTAD_PTAD3
Test           equ      PTAD_PTAD5
mTest          equ      mPTAD_PTAD5
mTest1         equ      mPTAD_PTAD4

Step           equ      10    ;Step Size

; Tested value for 4K7 + 0.68uF RC
Lowest_Boundary  equ    03    ; Lowest Boundary Limit Value [224uS]
S1_Boundary    equ      21    ; S1 Boundary Value (0.50V = 15) [608uS]
S2_Boundary    equ      30    ; S2 Boundary Value (0.75V = 23) [896uS]
S3_Boundary    equ      40    ; S3 Boundary Value (1.00V = 32) [1216uS]
S4_Boundary    equ      51    ; S4 Boundary Value (1.25V = 43) [1568uS]
S5_Boundary    equ      65    ; S5 Boundary Value (1.50V = 55) [2016uS]
S6_Boundary    equ      81    ; S6 Boundary Value (1.75V = 70) [2528uS]
S7_Boundary    equ      102   ; S7 Boundary Value (2.00V = 88) [3200uS]
S8_Boundary    equ      129   ; S8 Boundary Value (2.25V = 111) [4064uS]
S9_Boundary    equ      173   ; S9 Boundary Value (2.50V = 143) [5472uS]

TableStart:    equ      $00003E00
```

```
Auto_Mode_Init   equ   %10100000 ;25oC, Sleep_off, swing_off, auto_wind (Tx_Data32)
Heat_Mode_Init   equ   %11010000 ;28oC, Sleep_off, swing_off, auto_wind (Tx_Data32)

Tx_Flag_Init     equ   %00100011 ;TX_READY=0, TX_CNT=35 (Count down)

Data10_Init      equ   %00000010 ;AC_OFF, Auto_mode, oC, Light ON
Data32_Init      equ   %10100000 ;25oC, Sleep_off, Swing_off, auto_wind
Data54_Init      equ   %00001000 ;Model Set to Model 0 (b0 always equal to 0)
CtmCode_Init     equ   %10101001 ;0.63ms low + 0100101 customer code (Tx LSB first)

;(Value for Tx frame delay call for 0.5mS delay) for FSL
Head_Time_ON   equ      $10    ; Carrier on time for heading (8mS) 16*0.5ms
Head_Time_OFF  equ      $08    ; Carrier off time for heading (4mS) 8*0.5ms

; 0us for compensation of time delay by the instruction delay
Data0_Time_ON    equ   $32    ; Carrier on time for data 0 (500uS) 50*10us
Data0_Time_OFF   equ   $32    ; Carrier off time for data 0 (500uS) 50*10us
Data1_Time_ON    equ   $32    ; Carrier on time for data 1 (500uS) 50*10us
Data1_Time_OFF   equ   $96    ; Carrier off time for data 1 (1500uS) 150*10us

; Key_Flag bit definition
KEY_ON         equ      7      ;=1 if KBI occur, =0 if key released
KEY_WRONG      equ      6      ;=1 if Key Wrong, =0 if Key O.K.
KEY_FIRST_ON   equ      5      ;=1 if first timer setting ON key pressed
KEY_CONFIRM    equ      4      ;=1 if second timer setting ON key pressed
LCD_READY      equ      3      ;=1 go to LCD routine
KEY_REP        equ      2      ;=1 if Key Repeat within 250mS
TIM_FLASH      equ      1      ;=1 if toggle in 250mS T1OF
S34_KEY_ON     equ      0      ;=1 if S3 or S4 pressed

TX_READY       equ      7      ;=1 if Tx ready, =0 if Tx not ready

; [8.0ms delay, Timer clock = bus / 64 = 32us, 32us*250=8.0ms]
D_1mS   equ     31
D_2mS   equ     63
D_3mS   equ     94
D_4mS   equ     125
D_5mS   equ     156
D_6mS   equ     188
D_7mS   equ     219
D_8mS   equ     250

;==========================================================================
; Application Macro
;==========================================================================
StartTimer: macro
       mov     DelayPeriod, MTIMMOD          ; OF period
       mov     #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC ; Reset and Start Timer
       endm

StopTimer: macro
       mov     #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; Reset and Stop Timer
```

```
        endm

        org     TINY_RAMStart

; variable/data section
KeyRead         ds.b    1 ; ADC value from Keypad [00]

Auto_Mode       ds.b    1 ; Store wind speed + Temperature (Tx_Data32)=%10010000  [01]
Cool_Mode       ds.b    1 ; Store wind speed + Temperature (Tx_Data32)=%10010000  [02]
Humd_Mode       ds.b    1 ; Store wind speed + Temperature (Tx_Data32)=%10010000  [03]
Wind_Mode       ds.b    1 ; Store wind speed + Temperature (Tx_Data32)=%10010000  [04]
Heat_Mode       ds.b    1 ; Store wind speed + Temperature (Tx_Data32)=%11000000  [05]

Tx_Data10       ds.b    1 ; Nibble 1-0 (first 4-bit will be shift out)=%0000xxxx [06]
Tx_Data32       ds.b    1 ; Nibble 3-2 =%10010000 [07]
Tx_Data54       ds.b    1 ; Nibble 5-4 =%00000001 [08]
Tx_CtmCode      ds.b    1 ; Nibble 9-8 (last bit will be shift out)=%x0100101 [09]

Tx_Flag         ds.b    1       ; Tx Flag [0A]
Tx_Data_Temp    ds.b    1       ; Tmp Tx Data Store [0B]
Key_Flag        ds.b    1       ; Key_Flag [0C]

        org     RAMStart

; variable/data section
Tmp3            ds.b   1        ; Tmp [20]
;....
Tmpx            ds.b   1        ; Tmp [4F]

        org     ROMStart
; code section [3800]
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; Select $3FC0 - $3FFF
        mov     MAP_ADDR_6(NV_FTRIM), ICSSC   ; $3FFB -> ICSSC
        mov     MAP_ADDR_6(NV_ICSTRM), ICSTRM ; $3FFA -> ICSTRIM
        mov     #ICS_DIV_4, ICSC2             ; Use 2MHz

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

;--------------------------------------------------------
; Init RAM
```

```
;----------------------------------------------------------
        clr     KeyRead
; ------------------------------------------------------------------------------- *
; System Variable Init
; ------------------------------------------------------------------------------- *
        clr     Tx_Flag                         ; Initial Tx flag

        lda     #Auto_Mode_Init                 ; Initial difference mode value
                                                ; wind speed + Temperature (Tx_Data32)
        sta     Auto_Mode
        sta     Cool_Mode
        sta     Humd_Mode
        sta     Wind_Mode
        mov     #Heat_Mode_Init,Heat_Mode

        mov     #Data10_Init,Tx_Data10          ; Initial Tx Data + Customer code
        mov     #Data32_Init,Tx_Data32
        mov     #Data54_Init,Tx_Data54
        mov     #CtmCode_Init,Tx_CtmCode
;----------------------------------------------------------
; Config GPIO
; RC - init L
; IR - init L
;----------------------------------------------------------
        mov     #%00000011, PTAD                ; set all to 0 except PTA0 & 1
        mov     #(mRC|mIR|mTest|mTest1), PTADD; Set all to output including unused pins
;----------------------------------------------------------
;Config MTIM
;
;Timer prescalar=256 -> Timer clk~8kHz
;Bus = 2MHz (0.5uS)
;Max OF period = 32.768ms (128us *256)
;Timer resolution = 128us

;Timer prescalar=64 -> Timer clk 31.25KHz
;Bus = 2MHz (0.5uS)
;Max OF period = 8.192ms (32us *256)
;Timer resolution = 32us
;----------------------------------------------------------
        mov     #(MTIM_BUS_CLK|MTIM_DIV_64), MTIMCLK
        mov     #255, MTIMMOD
;----------------------------------------------------------
;Config KBI (KBI1ES default falling edge trigger)
;----------------------------------------------------------
        bclr    KEY, MAP_ADDR_6(PTAPUD)         ; Pullup selected [PTA2]
        mov     #(mKEY), MAP_ADDR_6(PTAPE)      ; Pullup/down Enable [PTA2]

        bclr    KEY, KBIES                      ; Keypads falling edge trigger
        mov     #(mKEY), KBIPE                  ; KBI Enable [KBI2]

;----------------------------------------------------------
;Key Scan Start
;----------------------------------------------------------
```

```
KeyScanStart:

        bset    RC, PTAD                        ; set PTA0 = 1

        mov     #%00000110, KBISC               ; Ack + KBI enable + Edge only

        STOP                                    ; Wait for KBI

        bset    KBISC_KBACK, KBISC              ; Clear Flag
        bset    KBISC_KBIE, KBISC               ; Disable KBI interrupt

        jsr     Delay_5mS

        brset   KEY, PTAD, KeyScanStart         ; Debounce

        jsr     Delay_5mS

        brset   KEY, PTAD, KeyScanStart         ; Debounce

        jsr     Delay_5mS

        brset   KEY, PTAD, KeyScanStart         ; Debounce

        bclr    RC, PTAD                        ; Discharge before start ADC test
        jsr     Delay_5mS
        jsr     Delay_5mS

        jsr    ReadSensor                       ; Read ADC value (1ms)

        lda     KeyRead

        cmp     #Lowest_Boundary                ; [vs 19]
        blo     Key_Error                       ; Key Error
        cmp     #S1_Boundary                    ; [vs 38]
        blo     S1                              ; S1 (ON/OFF) key pressed confirm
        cmp     #S2_Boundary                    ; [vs 63]
        blo     S2                              ; S2 (MODE) key pressed confirm
        cmp     #S3_Boundary                    ; [vs 86]
        blo     S3                              ; S3 (WIND) key pressed confirm
        cmp     #S4_Boundary                    ; [vs 113]
        blo     S4                              ; S4 (+) key pressed confirm
        cmp     #S5_Boundary                    ; [vs 138]
        blo     S5                              ; S5 (-) key pressed confirm
        cmp     #S6_Boundary                    ; [vs 163]
        blo     S6                              ; S6 (MODEL) key pressed confirm
        cmp     #S7_Boundary                    ; [vs 188]
        blo     S7                              ; S7 (SLEEP) key pressed confirm
        cmp     #S8_Boundary                    ; [vs 213]
        blo     S8                              ; S8 (LIGHT) key pressed confirm
        cmp     #S9_Boundary                    ; [vs 238]
        blo     S9                              ; S9 (SWING) key pressed confirm
        bra     Key_Error                       ; Key Error
```

```
Key_Error:

        bra     KeyScanStart

S1:     jmp     S1_Key
S2:     jmp     S2_Key
S3:     jmp     S3_Key
S4:     jmp     S4_Key
S5:     jmp     S5_Key
S6:     jmp     S6_Key
S7:     jmp     S7_Key
S8:     jmp     S8_Key
S9:     jmp     S9_Key

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Delay 8ms
; [8.0ms delay, Timer clock = bus / 64 = 32us, 32us*313=10.0ms]
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Delay_1mS:
        mov     #31, MTIMMOD                  ; OF period (1ms)
        bra     DelayX
Delay_3mS:
        mov     #94, MTIMMOD                  ; OF period (3ms)
        bra     DelayX
Delay_4mS:
        mov     #125, MTIMMOD                 ; OF period (3ms)
        bra     DelayX
Delay_5mS:
        mov     #156, MTIMMOD                 ; OF period (5ms)
        bra     DelayX
Delay_6mS:
        mov     #188, MTIMMOD                 ; OF period (6ms)
        bra     DelayX
Delay_7mS:
        mov     #219, MTIMMOD                 ; OF period (7ms)
        bra     DelayX
Delay_8mS:
        mov     #250, MTIMMOD                 ; OF period (8ms)

DelayX:
        mov     #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC ; Reset and Start Timer
        wait
        mov     #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; mask interrupt and clear flag
        rts

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Delay Xms (1-8mS)
; [8.0ms delay, Timer clock = bus / 64 = 32us, 32us*156=1-8ms]
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Delay_XmS:
        sta     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
        rts

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Read Keypad voltage Value
; Timer prescalar=8 -> Timer clk~250kHz
; Bus = 2MHz
; Max OF period = 1.02ms
; Timer resolution = 4us
; [i/p: ACMP interrupt]
; [o/p: KeyRead]
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ReadSensor:
        mov     #(MTIM_BUS_CLK|MTIM_DIV_8), MTIMCLK ; Change Timer resolution
        mov     #255, 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
        cmp     $0                              ;3 dummy~1.2us total, ACMP start delay
        bset    ACMPSC_ACF, ACMPSC              ; Clear ACMP Flag
        wait            ; delay 0.8ms and make the read process deterministic ??
        brclr   ACMPSC_ACF, ACMPSC, NoReading
        mov     MTIMCNT, KeyRead
        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_64), MTIMCLK ; Reset Timer resolution
        rts

NoReading:
        mov     #$FF, KeyRead                   ; Biggest Number
        clr     ACMPSC                          ; disable ACMP
        mov     #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; mask interrupt and clear flag
        mov     #(MTIM_BUS_CLK|MTIM_DIV_64), MTIMCLK ; Reset Timer resolution
        rts

; -------------------------------------------------------------------------------- *
; Tx Data Update
; -------------------------------------------------------------------------------- *
; -------------------------------------------------------------------------------- *
S1_Key:                                         ; ON/OFF Key pressed(ON/OFF)
        brclr   7,Tx_Data10,S1_ON               ; Check ON/OFF status

        bclr    7,Tx_Data10                     ; Change to OFF state
        bra     Clear_T_S

S1_ON:
        bset    7,Tx_Data10                     ; Change to ON state

Clear_T_S:
        bclr    3,Tx_Data32     ; Clear Sleep
```

```
        bclr    3,Auto_Mode
        bclr    3,Cool_Mode
        bclr    3,Humd_Mode
        bclr    3,Wind_Mode
        bclr    3,Heat_Mode
        bset    TX_READY,Tx_Flag                ; Tx ready

        jmp     Tx_Frame
; ----------------------------------------------------------------------------- *
S2_Key:                                         ; Modes Key pressed (MODE)
                                                ; (Auto>Cool>Humd>Wind>Heat)
        brclr   7,Tx_Data10,Slip_S2             ; No action if S1=OFF

        bclr    3,Tx_Data32                     ; Clear Sleep
        bclr    3,Auto_Mode
        bclr    3,Cool_Mode
        bclr    3,Humd_Mode
        bclr    3,Wind_Mode
        bclr    3,Heat_Mode
        lda     Tx_Data10                       ; Here (S1=ON)
        lsra                                    ; shift higher nibble to lower nibble
        lsra                                    ; All higher nibble is %0000
        lsra
        lsra
        cmp     #%00001100                      ; Reach Max. value (Heat mode)?
        blo     Inc_Modes
        lda     Tx_Data10
        and     #%10001111                      ; Change mode to 000
        sta     Tx_Data10
        bset    TX_READY,Tx_Flag                ; Tx ready
        bra     Mode_Update

Inc_Modes:
        inca
        lsla
        lsla
        lsla
        lsla
        sta     Tx_Data_Temp                    ; Store higher nibble
        lda     Tx_Data10
        and     #%00001111                      ; mask higher nibble
        ora     Tx_Data_Temp
        sta     Tx_Data10
        bset    TX_READY,Tx_Flag                ; Tx ready

Mode_Update:                                    ; Mode parameter to Tx_Data32

        lda     Tx_Data10
        and     #%01110000
        cmp     #%00000000                      ; Check Auto mode?
        beq     Auto_2_D32
        cmp     #%00010000                      ; Check Cool mode?
        beq     Cool_2_D32
```

```
        cmp     #%00100000                      ; Check Humd mode?
        beq     Humd_2_D32
        cmp     #%00110000                      ; Check Wind mode?
        beq     Wind_2_D32
        mov     Heat_Mode,Tx_Data32             ; It is Heat mode
        bra     End_S2

Auto_2_D32:
        mov     Auto_Mode,Tx_Data32
        bra     End_S2
Cool_2_D32:
        mov     Cool_Mode,Tx_Data32
        bra     End_S2
Humd_2_D32:
        mov     Humd_Mode,Tx_Data32
        bra     End_S2
Wind_2_D32:
        mov     Wind_Mode,Tx_Data32

Slip_S2:
End_S2:

        jmp     Tx_Frame
; --------------------------------------------------------------------------------- *
S3_Key:                                         ; "Fan Speed" Key pressed (WIND)
                                                ; (Auto>low>mid>high)
        brclr   7,Tx_Data10,Slip_S3             ; No action if AC OFF

        lda     Tx_Data32                       ; Here (AC ON)
        and     #%00000011                      ; mask other bit except b1-0
        cmp     #%00000011                      ; Reach Max. value (high)?
        beq     Rst_Wind
Inc_Wind:
        inc     Tx_Data32
        bra     End_S3

Rst_Wind:
        bclr    0,Tx_Data32
        bclr    1,Tx_Data32                     ; Change to Min. value (Auto)

End_S3:
        jsr     Data32_To_Modes                 ; Check & update Data32 & Modes
        bset    TX_READY,Tx_Flag                ; Tx ready

Slip_S3:

        jmp     Tx_Frame
; --------------------------------------------------------------------------------- *
S4_Key:                                         ; + Key pressed for oC / Model Set (^)
S4_Normal:
        brclr   7,Tx_Data10,Slip_S4             ; No action if S1=OFF
        lda     Tx_Data10
        and     #%01110000
```

```
        beq     Slip_S4                         ; No action if in Auto Mode

        lda     Tx_Data32
        lsra                                    ; shift higher nibble to lower nibble
        lsra                                    ; All higher nibble is %0000
        lsra
        lsra

        cmp     #%00001111                      ; Is reach Max $111, No need Inc
        bhs     No_Inc_Data32

        inca
        lsla
        lsla
        lsla
        lsla
        sta     Tx_Data_Temp                    ; Store higher nibble
        lda     Tx_Data32
        and     #%00001111                      ; mask higher nibble
        ora     Tx_Data_Temp

        sta     Tx_Data32

No_Inc_Data32:
        jsr     Data32_To_Modes                 ; Check & update Data32 & Modes
        bset    TX_READY,Tx_Flag                ; Tx ready
        bset    S34_KEY_ON,Key_Flag

Slip_S4:
End_S4:

        jmp     Tx_Frame
; ----------------------------------------------------------------------------- *
S5_Key:                                         ; - Key pressed for oC / Model Set (v)
S5_Normal:
        brclr   7,Tx_Data10,Slip_S5             ; No action if S1=OFF
        lda     Tx_Data10
        and     #%01110000
        beq     Slip_S5                         ; No action if in Auto Mode

        lda     Tx_Data32
        lsra                                    ; shift higher nibble to lower nibble
        lsra                                    ; All higher nibble is %0000
        lsra
        lsra
        cmp     #%00000000                      ; Is reach Min $0000, No need Dec
        beq     No_Dec_Data32

        deca
        lsla
        lsla
        lsla
        lsla
```

```
        sta     Tx_Data_Temp                    ; Store higher nibble
        lda     Tx_Data32
        and     #%00001111                      ; mask higher nibble
        ora     Tx_Data_Temp

        sta     Tx_Data32

No_Dec_Data32:
        jsr     Data32_To_Modes                 ; Check & update Data32 & Modes
        bset    TX_READY,Tx_Flag                ; Tx ready
        bset    S34_KEY_ON,Key_Flag

Slip_S5:
End_S5:

        jmp     Tx_Frame
; ----------------------------------------------------------------------------- *
S6_Key:                                         ; "Model Set" Key pressed (SET)
                                                ; Model Number cannot be change
                                                ; due to no LCD in KA2 demo

        brclr   3,Tx_Data54,Model_Set

Model_Confirm:
        bclr    3,Tx_Data54                     ; Model Confrim (MODEL ON)
        bra     End_S6

Model_Set:
        bset    3,Tx_Data54                     ; Model Set (MODEL flash)

End_S6:

        jmp     Tx_Frame
; ----------------------------------------------------------------------------- *
S7_Key:                                         ; Sleep Key pressed (OK)
        brclr   7,Tx_Data10,Slip_S7             ; No action if S1=OFF

        brset   3,Tx_Data32,Sleep_OFF           ; Check ON/OFF? (1=ON)

                                                ; Here AC ON
        lda     Tx_Data10
        and     #%01110000                      ; mask all bit except b6-4
        cmp     #%00010000                      ; Check Cool mode (001)
        beq     Sleep_ON_Cool
        cmp     #%00100000                      ; Check Humd mode (010)
        beq     Sleep_ON_Humd
        cmp     #%01000000                      ; Check Heat mode (100)
        beq     Sleep_ON_Heat
        bra     Slip_S7                         ; Slip if in others modes

Sleep_ON_Cool:
        bset    3,Cool_Mode
        bra     Set_Data32
```

```
Sleep_ON_Humd:
        bset    3,Humd_Mode
        bra     Set_Data32

Sleep_ON_Heat:
        bset    3,Heat_Mode

Set_Data32:
        bset    3,Tx_Data32                   ; OFF -> ON
        bset    TX_READY,Tx_Flag              ; Tx ready
        bra     End_S7

Sleep_OFF:                                    ; Here AC OFF

        lda     Tx_Data10
        and     #%01110000                    ; mask all bit except b6-4
        cmp     #%00010000                    ; Check Cool mode (001)
        beq     Sleep_OFF_Cool
        cmp     #%00100000                    ; Check Humd mode (010)
        beq     Sleep_OFF_Humd
        cmp     #%01000000                    ; Check Heat mode (100)
        beq     Sleep_OFF_Heat
        bra     Slip_S7

Sleep_OFF_Cool:
        bclr    3,Cool_Mode
        bra     Clr_Data32
Sleep_OFF_Humd:
        bclr    3,Humd_Mode
        bra     Clr_Data32

Sleep_OFF_Heat:
        bclr    3,Heat_Mode

Clr_Data32:
        bclr    3,Tx_Data32                   ; OFF -> ON
        bset    TX_READY,Tx_Flag              ; Tx ready
        bra     End_S7

Slip_S7:
End_S7:

        jmp     Tx_Frame
; ------------------------------------------------------------------------------ *
S8_Key:                                       ; Light Key pressed (M.WIND)
        brset   1,Tx_Data10,Light_OFF         ; Check ON/OFF? (1=ON)
        bset    1,Tx_Data10                   ; OFF -> ON
        bset    TX_READY,Tx_Flag              ; Tx ready
        bra     End_S8

Light_OFF:
        bclr    1,Tx_Data10                   ; ON -> OFF
        bset    TX_READY,Tx_Flag              ; Tx ready
```

```
End_S8:

        jmp     Tx_Frame
; ------------------------------------------------------------------------------ *
S9_Key:                                         ; Swing Key pressed (A.M.WIND)
        brclr   7,Tx_Data10,Slip_S9             ; No action if S1=OFF

                                                ; Here (S1=ON)
        brset   2,Tx_Data32,Swing_OFF           ; Check ON/OFF? (1=ON)
        bset    2,Tx_Data32                     ; OFF -> ON
        bset    2,Auto_Mode
        bset    2,Cool_Mode
        bset    2,Humd_Mode
        bset    2,Wind_Mode
        bset    2,Heat_Mode
        bset    TX_READY,Tx_Flag                ; Tx ready
        bra     End_S9

Swing_OFF:
        bclr    2,Tx_Data32                     ; ON -> OFF
        bclr    2,Auto_Mode
        bclr    2,Cool_Mode
        bclr    2,Humd_Mode
        bclr    2,Wind_Mode
        bclr    2,Heat_Mode
        bset    TX_READY,Tx_Flag                ; Tx ready

Slip_S9:
End_S9:

        jmp     Tx_Frame
; ------------------------------------------------------------------------------ *
; Update Data32 to Difference Modes (Auto mode check can be remove)
; ------------------------------------------------------------------------------ *
Data32_To_Modes:

        lda     Tx_Data10
        and     #%01110000
        cmp     #%00000000                      ; Check Auto mode?
        beq     D32_2_Auto
        cmp     #%00010000                      ; Check Cool mode?
        beq     D32_2_Cool
        cmp     #%00100000                      ; Check Humd mode?
        beq     D32_2_Humd
        cmp     #%00110000                      ; Check Wind mode?
        beq     D32_2_Wind
        mov     Tx_Data32,Heat_Mode             ; It is Heat mode
        rts
D32_2_Auto:
        mov     Tx_Data32,Auto_Mode
        rts
D32_2_Cool:
```

```
        mov     Tx_Data32,Cool_Mode
        rts
D32_2_Humd:
        mov     Tx_Data32,Humd_Mode
        rts
D32_2_Wind:
        mov     Tx_Data32,Wind_Mode
        rts

; ------------------------------------------------------------------------------ *
; Frame Tx (need to check Tx_Ready flag)
;
; <Need to fine turn the timming of bit transmission>!!!!!!
; ------------------------------------------------------------------------------ *
Tx_Frame:

; ---------------------------------------------------------- *
Tx_Header:                                      ; Header Code Tx

; Here Bus = 2MHz, Timer clock = 128us, Instruction Cycle = 0.5us
; 26us 8us(ON)+14us(OFF) carrier freq.
        mov     #(MTIM_BUS_CLK|MTIM_DIV_256), MTIMCLK ; Change Timer resolution
        mov     #62, MTIMMOD ; OF period = 128*62=8.0mS
        mov     #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC ; Reset and Start Timer
HEADER_ON:
        bset    IR,PTAD         ; IR ON         [5]
        bset    IR,PTAD         ; IR ON         [5]
        bset    IR,PTAD         ; IR ON         [5]
        nop                     ;               [1]
                                ; 16*.5 = 8us
        bclr    IR,PTAD         ; IR OFF        [5]
        bclr    IR,PTAD         ; IR OFF        [5]
        bclr    IR,PTAD         ; IR OFF        [5]
        bclr    IR,PTAD         ; IR OFF        [5]
        bclr    IR,PTAD         ; IR OFF        [5]
        bclr    IR,PTAD         ; IR OFF        [5]
        nop                     ;               [1]
        brclr   MTIMSC_TOF, MTIMSC, HEADER_ON;[5]
                                ; 36*.5 = 18us

        mov     #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; mask interrupt and clear flag

HEADER_OFF:
        mov     #31, MTIMMOD ; OF period = 128*31=4mS
        mov     #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC ; Reset and Start Timer
        wait

        mov     #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; mask interrupt and clear flag
        mov     #(MTIM_BUS_CLK|MTIM_DIV_64), MTIMCLK ; Reset Timer resolution

; ------------------------------------------------------------------------------ *
; Tx Data from 2.0 - 9.3
; ------------------------------------------------------------------------------ *
```

```
        lda     Tx_Data10

        ldx     #$08                            ; [1us]
Tx_Loop_10:
        lsra                                    ; Right shift LSB to Carry flag [0.5us]
        blo     Data_N_10                       ; [1.5us] jump if C=1
        bsr     Data_0                          ; Send Data 0 [2us] if C=0
        bra     Tx_N_10_Next
Data_N_10:
        bsr     Data_1                          ; Send Data 1 [2us] if C=1
Tx_N_10_Next:
        dbnzx   Tx_Loop_10                      ; Transmit again if X>0 [2us]

; ----------------------------- *
        lda     Tx_Data32

        ldx     #$08                            ; [1us]
Tx_Loop_32:
        lsra                                    ; Right shift LSB to Carry flag [0.5us]
        blo     Data_N_32                       ; [1.5us] jump if C=1
        bsr     Data_0                          ; Send Data 0 [2us] if C=0
        bra     Tx_N_32_Next
Data_N_32:
        bsr     Data_1                          ; Send Data 1 [2us] if C=1
Tx_N_32_Next:
        dbnzx   Tx_Loop_32                      ; Transmit again if X>0 [2us]

; ----------------------------- *
        lda     Tx_Data54

        ldx     #$08                            ; [1us]
Tx_Loop_54:
        lsra                                    ; Right shift LSB to Carry flag [0.5us]
        blo     Data_N_54                       ; [1.5us] jump if C=1
        bsr     Data_0                          ; Send Data 0 [2us] if C=0
        bra     Tx_N_54_Next
Data_N_54:
       bsr Data_1                               ; Send Data 1 [2us] if C=1
Tx_N_54_Next:
        dbnzx   Tx_Loop_54                      ; Transmit again if X>0 [2us]

; ----------------------------- *
        lda     Tx_CtmCode
        ldx     #$08                            ; [1us]
Tx_Loop_76:
        lsra                                    ; Right shift LSB to Carry flag [0.5us]
        blo     Data_N_76                       ; [1.5us] jump if C=1
        bsr     Data_0                          ; Send Data 0 [2us] if C=0
        bra     Tx_N_76_Next
Data_N_76:
        bsr Data_1                              ; Send Data 1 [2us] if C=1
Tx_N_76_Next:
        dbnzx   Tx_Loop_76                      ; Transmit again if X>0 [2us]
```

```
; ---------------------------- *
Tx_Stop:                                ; Stop bit Tx

       bsr     Data_1                   ; Send Data_1 as stop bit !!!

       bclr    TX_READY,Tx_Flag ; clear TX_READY to avoid next Tx until other key
                                ; pressed
       clr     Tx_Flag

; --------------------------------------------------------- *
Key_Released:

       jsr     Delay_1mS
       brset   KEY, PTAD, Key_Released

       jmp     KeyScanStart             ; Repeat Key Scan

; --------------------------------------------------------- *

; --------------------------------------------------------------------------- *
; Tx Data "0" OR Data "1"
; --------------------------------------------------------------------------- *
Data_0:                       ; 630us carrier + 560us No carrier

; Here Bus =2MHz, Timer clock = bus/64 = 32us, Instruction Cycle = 0.5us
; 26us 8us(ON)+14us(OFF) carrier freq.

       mov     #16, MTIMMOD                   ; OF period = 32*16=500us
       mov     #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC ; Reset and Start Timer
Data0_ON:
       bset    IR,PTAD        ; IR ON         [5]
       bset    IR,PTAD        ; IR ON         [5]
       bset    IR,PTAD        ; IR ON         [5]
       nop                    ;               [1]
                              ; 16*.5 = 8us
       bclr    IR,PTAD        ; IR OFF        [5]
       bclr    IR,PTAD        ; IR OFF        [5]
       bclr    IR,PTAD        ; IR OFF        [5]
       bclr    IR,PTAD        ; IR OFF        [5]
       bclr    IR,PTAD        ; IR OFF        [5]
       bclr    IR,PTAD        ; IR OFF        [5]
       nop                    ;               [1]
       brclr   MTIMSC_TOF, MTIMSC, Data0_ON; [5]
                              ; 36*.5 = 18us
       mov     #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; mask interrupt and clear flag

Data0_OFF:
       mov     #16, MTIMMOD                   ; OF period = 32*16=500us
       mov     #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC ; Reset and Start Timer
       wait

       mov     #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; mask interrupt and clear flag
```

```
        rts                                 ; [4us]
; ------------------------------------------------------------------------------- *
Data_1:                                     ; 630us carrier + 1660us No carrier

; Here Bus =2MHz, Timer clock = bus/64 = 32us, Instruction Cycle = 0.5us
; 26us 8us(ON)+14us(OFF) carrier freq.
        mov     #16, MTIMMOD                ; OF period = 32*16=500us
        mov     #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC ; Reset and Start Timer
Data1_ON:
        bset    IR,PTAD         ; IR ON         [5]
        bset    IR,PTAD         ; IR ON         [5]
        bset    IR,PTAD         ; IR ON         [5]
        nop                     ;               [1]
                                ; 16*.5 = 8us
        bclr    IR,PTAD         ; IR OFF        [5]
        bclr    IR,PTAD         ; IR OFF        [5]
        bclr    IR,PTAD         ; IR OFF        [5]
        bclr    IR,PTAD         ; IR OFF        [5]
        bclr    IR,PTAD         ; IR OFF        [5]
        bclr    IR,PTAD         ; IR OFF        [5]
        nop                     ;               [1]
        brclr   MTIMSC_TOF, MTIMSC, Data1_ON ;[5]
                                ; 36*.5 = 18us
        mov     #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; mask interrupt and clear flag

Data1_OFF:
        mov     #47, MTIMMOD    ; OF period = 32*47=1664us
        mov     #(mMTIMSC_TRST|mMTIMSC_TOIE), MTIMSC ; Reset and Start Timer
        wait

        mov     #(mMTIMSC_TSTP|mMTIMSC_TRST), MTIMSC ; mask interrupt and clear flag

        rts                     ; [4us]

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; 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

DRM081
Rev. 0, 6/2006