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

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

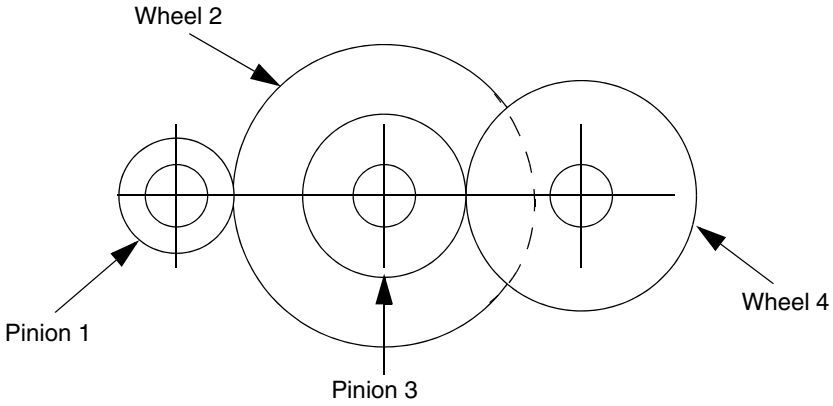

Figure 2. Gear System of a Mechanical Timer

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.

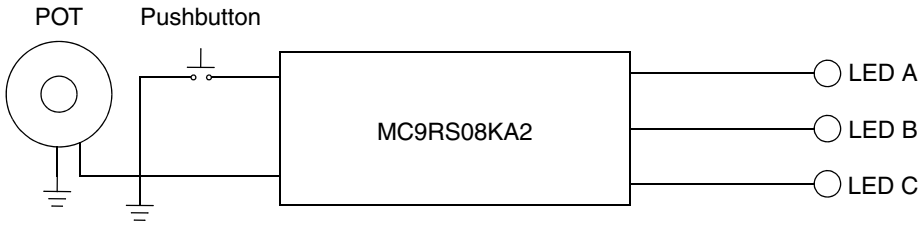

Figure 3. Digital Timer Hardware Configuration

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.

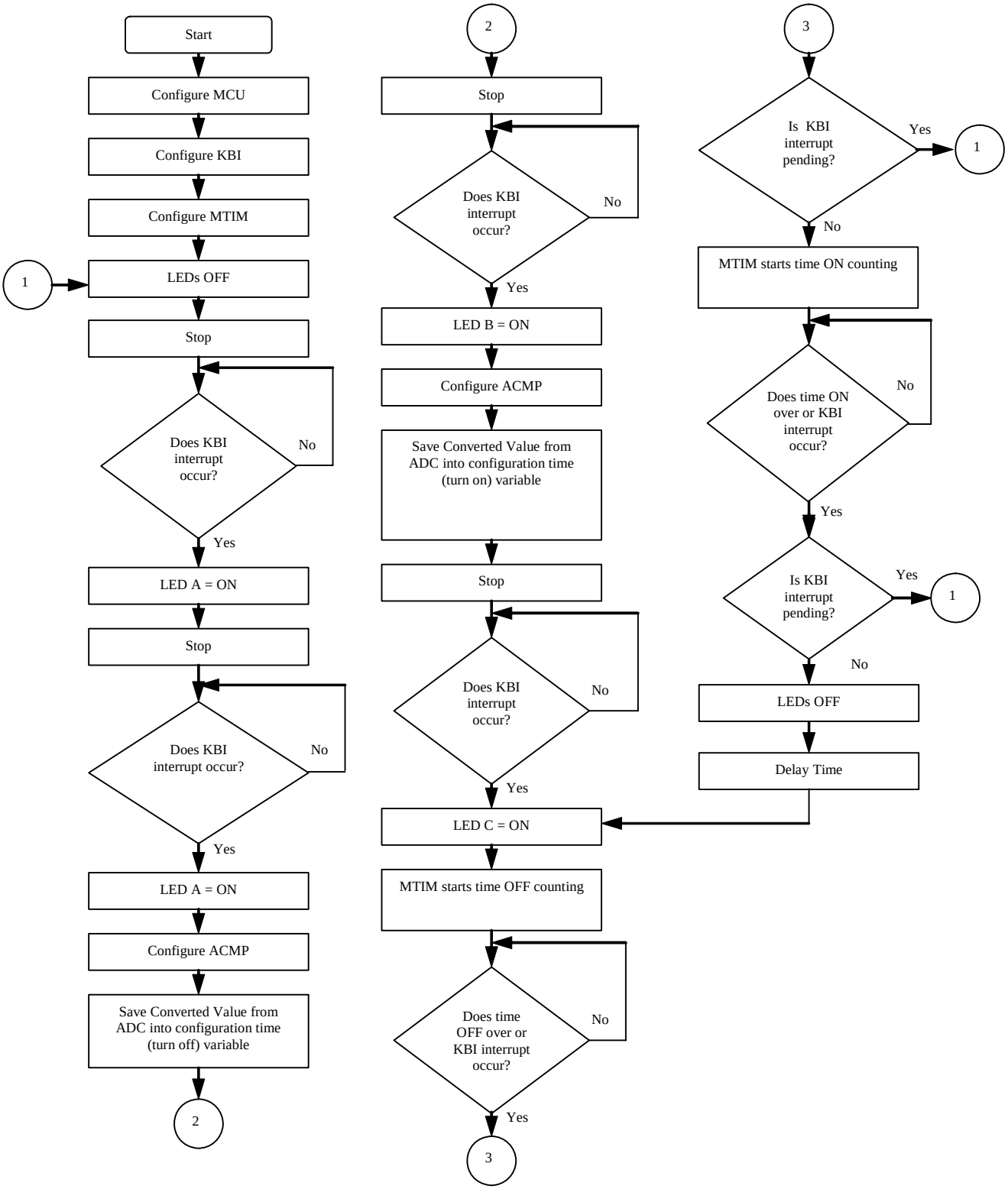

Figure 4. Basic Code Functions Flow Chart

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

freescale™
semiconductor