Application Description

Figure 3 is the smart NiCd/NiMH battery charger circuit diagram. The system combines the voltage flat and the absolute temperature detection techniques followed by the trickle charge method to charge a NiCd/NiMH battery pack with three cells.

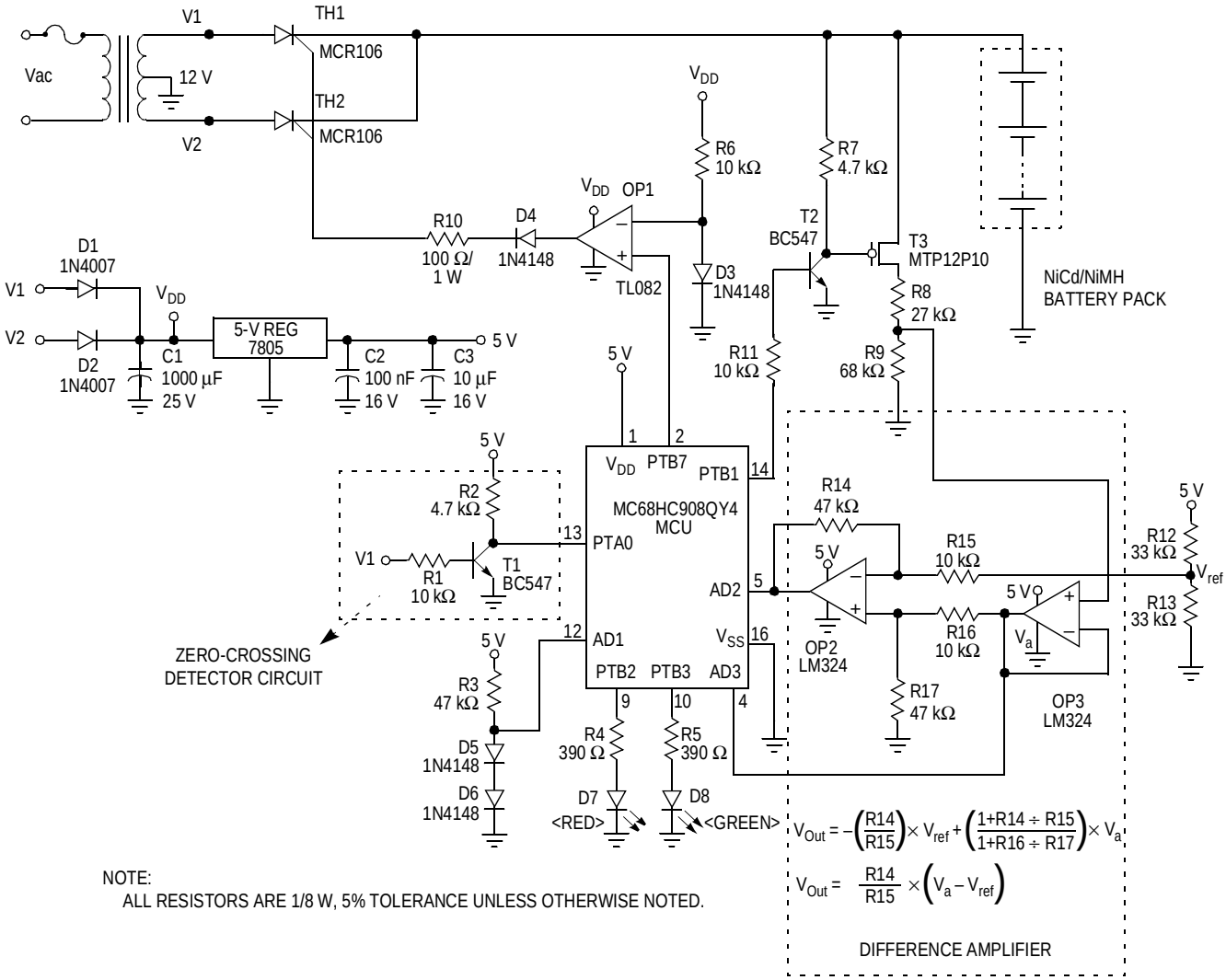

Figure 3. Battery Charger Circuit Diagram

Initialization

The battery charge procedure begins by determining whether a battery pack is available for charging by measuring for voltage.

1. MC68HC908QY4 MCU port PTB1 turns on the bipolar transistor T2.
2. The OP3 buffer allows the MCU ADC to read the battery pack voltage with the AD3 channel. An ADC reading greater than 500 mV indicates that a battery pack is available for charging. Discharging the battery set is not required before starting the charging process.

Controlled Rectifier

The MC68HC908QY4 MCU triggers the controlled rectifier, composed of two thyristors connected as a full-wave bridge rectifier, to provide the charge current for the battery pack.

3. The MCU activates the controlled rectifier after the zero-crossing of the ac sinusoidal signal within a user-defined time duration. The system uses the zero-crossing detector circuit to start the MCU timer counter by sensing any level transition in PTA0 and waits to generate the rectifier trigger signal via PTB7. For a 60 Hz ac line frequency, the time duration is set by default to approximately 4 ms to trigger the thyristors at the sinusoidal peak and provide 600 mA full charge current to the battery pack, as illustrated in Figure 4.

Charge Period

4. At the end of the user-defined charge period, the MCU pauses to turn on the controlled rectifier and measure the voltage variation of the battery packs. If the MC68HC908QY4 MCU is operating with the internal oscillator and it has been trimmed to obtain a 3.2-MHz bus clock frequency, the charge period is set by default to approximately 10 minutes.
5. To measure the battery voltage, the MCU asserts again the PTB1 pin to turn on the bipolar transistor T2 and apply the battery voltage to the resistor R9.

Overcharge Protection

The battery voltage variation measurement is intended to detect the decline of the battery pack voltage and the onset of the overcharging process.

6. The system detects when the slope of the battery voltage curve (during the charge procedure) becomes lower than zero ($\Delta V/\Delta t < 0$, see Figure 1 and Figure 2).
7. The system subtracts the reference voltage (V_{ref}) from the voltage on the battery packs using the difference amplifier (as shown in Figure 3). The MC68HC908QY4 MCU converts that data to a digital word using its internal ADC and the AD2 channel.
8. After the analog-to-digital conversion is complete, the MCU stores the subtraction result in an internal MCU variable. The initial value of this internal MCU variable is set by default to $00 and, assuming a difference amplifier gain of 4.7, the reference voltage for a battery pack of three cells is V_{ref} = 2.50 V.
   a. If the battery set voltage is lower than (or equal to) the reference voltage, the ADC reading is always larger than (or equal to) the value stored in the MCU variable at the end of the previous charge period. Therefore, the system continues the battery charge process during a subsequent charge period.
   b. If the battery set voltage begins to decline, the ADC reading will be lower than the previous value and the system stops the normal charge process and starts the trickle charge procedure. During trickle charge, current flowing into the batteries is reduced to 28 mA by adjusting the MCU timer counter to obtain a delay of 7 ms with regard to the zero-crossing of the sinusoidal ac signal, as shown in Figure 5. While in trickle charge, the system does not monitor the battery pack voltage.

Temperature Protection

9. To provide absolute temperature protection, the voltage drop across diodes D5 and D6 (located very near the battery pack) is measured to check for a user-defined battery temperature variation using the AD1 channel. Like the battery voltage variation measurement, the D5 and D6 voltage check is performed after a charge period. The initial D5 and D6 voltage drop is measured at the beginning of the normal charge process. For example, if the temperature increases by approximately 30^{o}C, the D5 and D6 voltage drop change would be equivalent to DV @ –125 mV (or roughly seven LSBs of the MC68HC908QY4 MCU 8-bit ADC). If the system detects a voltage-related temperature variation larger than the maximum allowed by user, it starts the trickle charge procedure.

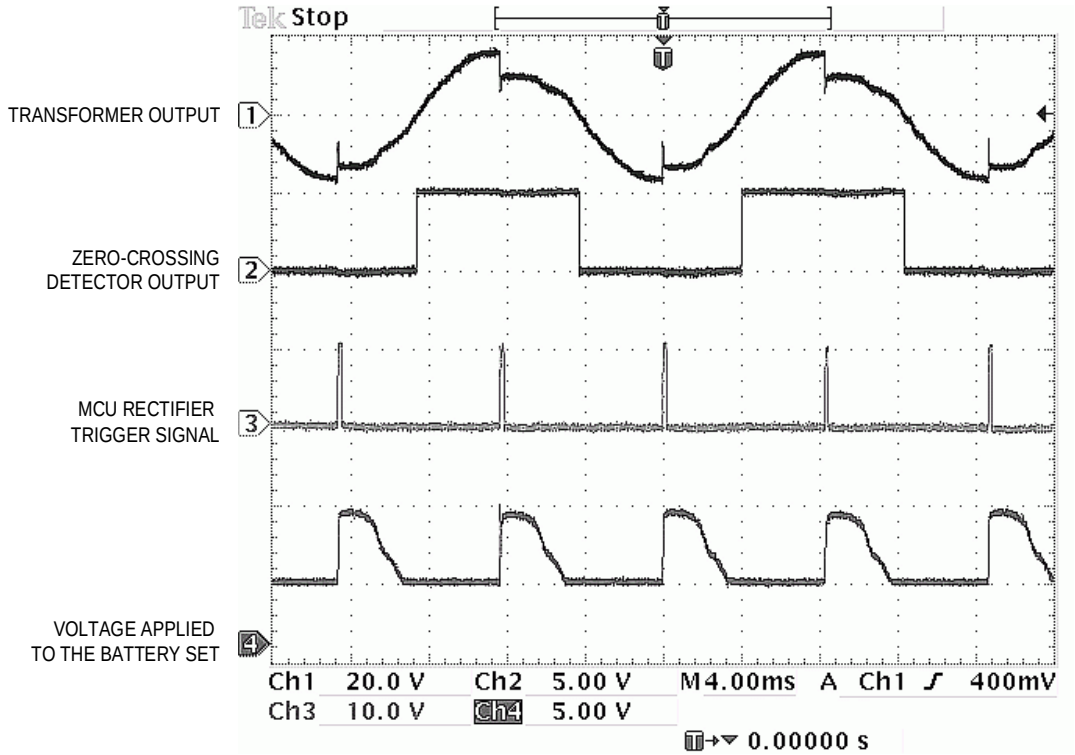

Figure 4. Normal NiCd/NiMH Battery Charge Waveforms

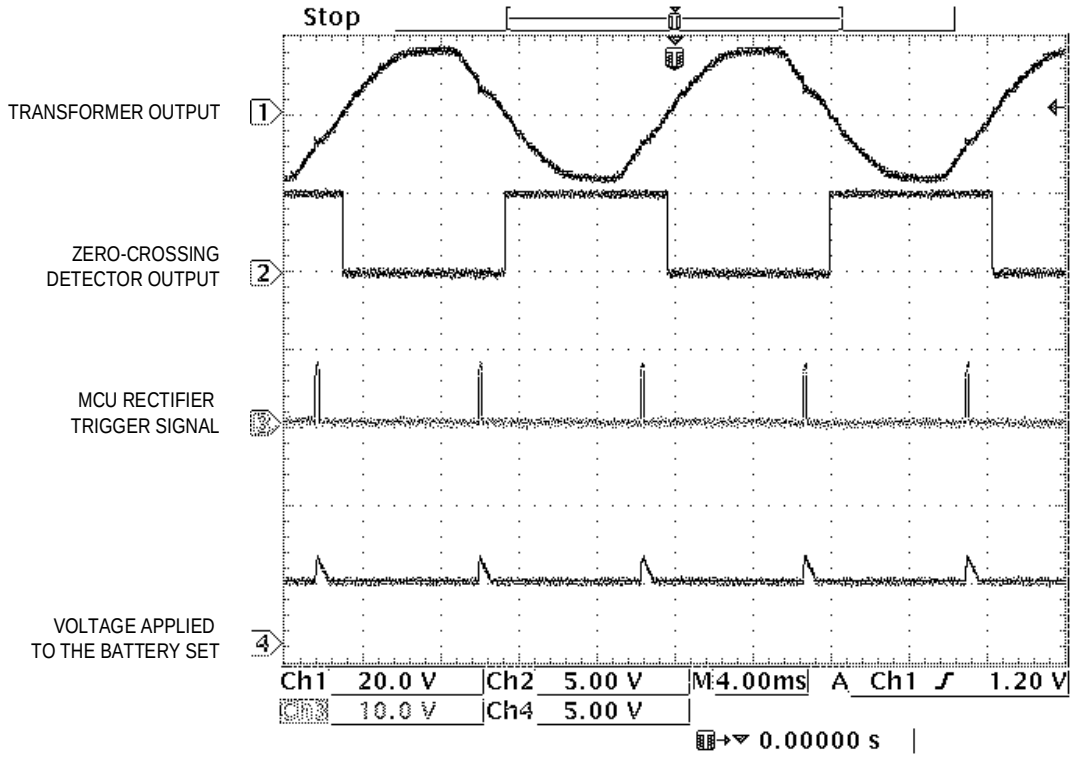

Figure 5. Trickle Charge Waveforms

Software Description

The following process is illustrated in Figure 6.

1. Initialization

The software starts configuring the I/O ports and registers properly, clearing variables and setting the timer to generate the appropriate thyristor conduction angle. In this application, timer registers are defined to overflow at about 4 ms. Constants *InitTMODH* and *InitTMODL* modify the values of the TMODH and TMODL registers on the timer module.

2. Monitoring Loop

After performing the initialization operations, the code enters a loop where the battery voltage is continually monitored through ADC channel 3 to determine whether a battery pack is connected. Constant *BatInit* sets the minimum value to check whether the battery pack is engaged.

As soon as a battery pack is connected to the charger, the ADC detects a voltage value larger than the one previously stored in *BatInit* and the charging process begins.

3. Begin Charging

Initially, the system measures the temperature of the battery packs. ADC channel 1 reads the voltage across the diodes D5 and D6. The value is converted and stored into the *FrstTmpRd* variable. This is the first value for temperature comparison. The red LED is turned on by setting PTB3, which indicates the beginning of the charging cycle.

4. Counter Loop

A loop is implemented and PTA0 waits for a level transition coming from the zero-crossing detector circuitry. After the transition is detected, the timer module is started and stays in a loop until it overflows. Then timer module is stopped and cleared and PTB7 triggers the thyristor.

5. Pause Charging

Counter variables are incremented. After the ten minute charging period (which can be changed by modifying the *StpChL* and *StpChH* constants), the charging process is paused. Counters are cleared and battery voltage is monitored again by setting PTB1 and reading ADC channel 2. A delay is needed to stabilize the voltage on the battery. The new ADR register value is subtracted from the last stored value.

- If the new value is lower than the previous value, the charging process is stopped and the trickle charging subroutine is defined.
- If the new voltage value is greater-than or equal-to the previous value, the charging cycle continues. At this time, the new voltage acquired by the ADC is stored in a variable to be compared with the next value that will be captured at the end of the next charge period.

Current temperature is captured (by reading the voltage over the diodes D5 and D6 by ADC channel 1) and compared with the first value. If difference in temperatures is greater than a predefined value on the *TmpSafe* constant, the routine goes to trickle charging. If the temperature is less than this value, the charging process continues. This is done to protect against battery overheat.

6. Trickle Charging

When trickle charging, PTB2 turns on the green LED and PTB3 turns off the red LED, which indicates the battery is fully charged. The timer module registers are changed to increment the overflow period. In this case, the timer overflows at about 7 ms after PTA0 detects the level transition. PTB7 pulses at the end of the power cycle line, reducing the current that charges the battery pack. It stays in a loop until the user disconnects the battery pack from the charger.

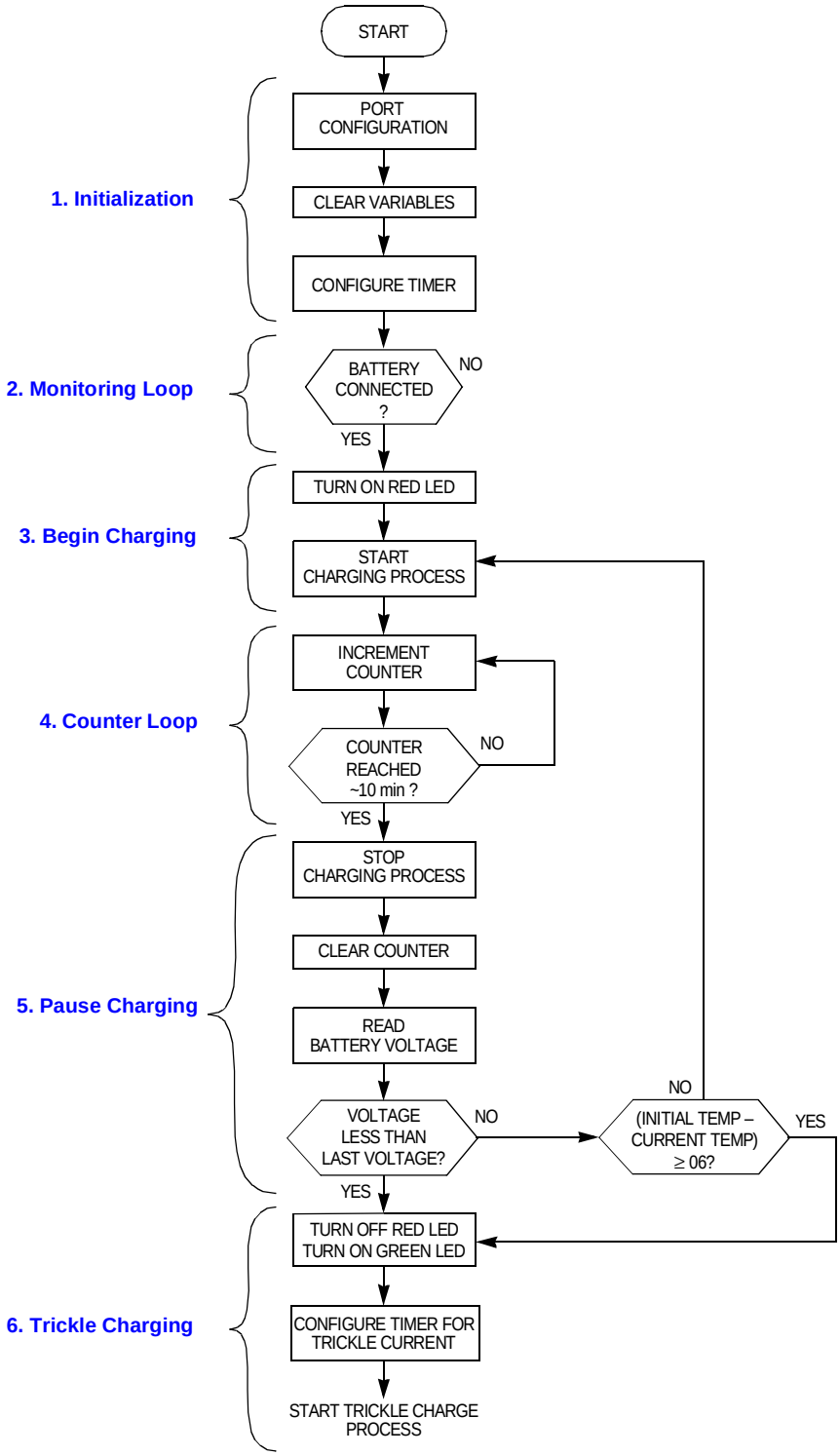

Figure 6. The NiCd/NiMH Battery Charger Software

```
;*******************************************************************************************
;* Equates and Data Table Includes
;*******************************************************************************************

                        include 'MC68HC908QT4.equu'    ; For the QT1, QT2, QT4, QY1, QY2, QY4

        org $FFC0

;trimval:    DC.B $FF           ;here we set the FLASH trim to a default value.
                               ;DO NOT change this value, as the trim will not be
                               ;automatically calibrated by the programming interface if
                               ;this value is anything other than $FF

;DEFAULT_RAM         SECTION SHORT
        org   RamStart


;*******************************************************************************************
;* Constants and Variables for this file
;*******************************************************************************************

               include 'battery.equ'

;DEFAULT_ROM         SECTION
            org   FlashStart


;*******************************************************************************************
;* SUBROUTINES
;* This part includes subroutines
;*******************************************************************************************

;Subroutine for Timer

TimerHalfL: mov   #initTim,TSC ;Timer - Cleared + Stopped.

            mov   #InitTMODHL,TMODH ;Set Timer to ~ 4ms or 1/4 Power Cycle Line (PCL)
            mov   #InitTMODLL,TMODL ;after we start the timer

            jmp  Skip

TimerHalfH: mov   #initTim,TSC ;Timer - Cleared + Stopped.

            mov   #InitTMODHH,TMODH ;Set Timer to ~ 4ms or 1/4 Power Cycle Line (PCL)
            mov   #InitTMODLH,TMODL ;after we start the timer.

            jmp  Skip


Trickle:    mov   #initTim,TSC  ;Timer 1 - Cleared + Stopped.

            mov   #initTricH,TMODH ;Set Timer to low current
            mov   #initTricL,TMODL ;after we start the timer

            jmp  Skip
```

```
;Subroutine for Thyristor gate control

Gate:       lda   #GateVal    ;Gate pulse duration
loop:       bset  PTB7,PTB
            dbnza loop

            bclr  PTB7,PTB    ;PTB7 generates a pulse on Thyristor gate

            jmp  Skip

;Subroutine for Battery reading

BatRead:    bset  PTB1,PTB    ;Turn transistor on

            mov   #initADCH3,ADSCR ;Start Conversion, CH3 selected

            lda   ADR
            cmp   #BatInit    ;Keep in this loop while battery not connected
            blo   BatRead

            bclr  PTB1,PTB    ;Turn transistor off

            bra   Skip

;Subroutine for Timer Overflow

TOverflow:  nop
            nop
            brclr TOF,TSC,TOverflow ;Wait for Timer Overflow

            lda   TSC
            and   #TSCClr
            sta   TSC         ;Clear TOF bit

            mov   #initTim,TSC ;STOP and RESET Counter

            bra   Skip

;Subroutine to delay about 2s before reading battery voltage

Delay:      ldx   #Del

Delay1:     lda   #Dela1
            sta   del1

Delay2:     lda   #Dela2
            sta   del2

Delay3:     nop
            dbnz  del2,Delay3

            dbnz  del1,Delay2

            dbnzx Delay1

            bra   Skip
```

```
;Subroutine for battery voltage reading

VbattH:     mov   #initADCH2,ADSCR  ;Start Conversion, CH2 selected

Waitcoco:   nop
            nop
            brclr COCO,ADSCR,Waitcoco ;Wait for Conversion complete

            lda   ADR         ;Load AD value
            and   #MaskLSB    ;Mask LSB
            sta   VoltReadH   ;store value into variable

            bra   Skip

;Subroutine for first battery temperature reading

VFrsttemp:  mov   #initADCH1,ADSCR  ;Start Conversion, CH1 selected

Waitcoco1:  nop
            nop
            brclr COCO,ADSCR,Waitcoco1 ;Wait for Conversion complete

            lda   ADR         ;Load AD value
            sta   FrstTmpRd   ;store value into variable

            bra   Skip

;Subroutine for battery temperature reading

VActemp:    mov   #initADCH1,ADSCR  ;Start Conversion, CH1 selected

Waitcoco2:  nop
            nop
            brclr COCO,ADSCR,Waitcoco2 ;Wait for Conversion complete

            lda   ADR         ;Load AD value
            sta   AcTmpRd     ;store value into variable

Skip:       rts

;*******************************************************************************************
;* main    - This is the point where code starts executing
;*             after a RESET.
;*******************************************************************************************
Entry:
main:
            mov   #initCfg1,CONFIG1 ;Set config1 register
                        ;(LVI and COP disabled)

            mov   #initCfg2,CONFIG2 ;set MCU to internal oscillator

            clr   PTB
            mov   #InitDDRB,DDRB  ;PTB7 -> Pulses on Thyristor gate
                        ;PTB3 -> Red LED (Bat. Charging)
                        ;PTB2 -> Green LED (Bat. Charged)
                        ;PTB1 -> Transistor Control
```

```
            bclr  DDRA0,DDRA  ;Zero Crossing detection

            mov   #ADclkval,ADICLK ;ADC clock, bus clock/ 16

;Enable ADCH3

            mov   #initADCH3,ADSCR ;Start Conversion, CH3 selected


            lda   TRIMLOC     ;load the TRIM value stored in FLASH
            sta   OSCTRIM     ;use this stored value.

            rsp
            clra
            clrx

;Clear Variables

            clr   Counter0
            clr   Counter1

            clr   VoltReadL
            clr   VoltReadH

            clr   AcTmpRd
            clr   FrstTmpRd

            jsr   TimerHalfL   ;Go config Timer
            cli               ;Allow interrupts to happen

            jsr   BatRead     ;Go read battery

            jsr   VFrsttemp   ;Go read First temp value

            bset  PTB3,PTB    ;Turn on Red LED (Battery is charging)

Waitpta0:   nop
            brclr PTA0,PTA,Waitpta0  ;Wait for a positive edge on PTA0 (Zero crossing)

            jsr   TimerHalfH  ;Go config Timer

            mov   #StartTim,TSC  ;Start the timer

            jsr   TOverflow   ;Go to Timer Overflow subroutine

            jsr   Gate        ;Go to Gate subroutine

Waitpta:    nop
            brset  PTA0,PTA,Waitpta  ;Wait for a negative edge on PTA0 (Zero crossing)

            jsr   TimerHalfL   ;Go config Timer

            mov   #StartTim,TSC ; Start the timer

            jsr   TOverflow   ;Go to Timer Overflow subroutine
```

```
            jsr   Gate        ;Go to Gate subroutine

            inc   Counter0    ;Increment 1st byte Counter for charge time OVF period
            lda   #StpChL
            cbeq  Counter0,Count1  ;Go to Count1 if Counter0 > $FF

            bra   Waitpta0

Count1:     inc   Counter1    ;Increment 2nd byte Counter for charge time OVF period
            lda   #StpChH
            cbeq  Counter1,Vbat  ;Go to Vbat if Counter1 > $90

            bra   Waitpta0

Vbat:       mov   #initTim,TSC  ;Stop and reset counter

            clr   Counter0
            clr   Counter1

            bset  PTB1,PTB    ;Turn transistor on

            jsr   Delay       ;Go to Delay subroutine

            jsr   VbattH      ;Jump to subroutine that reads battery voltage

            jsr   Delay       ;Go to Delay subroutine

            bclr  PTB1,PTB    ;Turn transistor off

            lda   VoltReadH
            sub   VoltReadL   ;Compare last battery voltage with current one

            blo   Charged     ;Jump to Charged if last value < current value

            lda   VoltReadH
            sta   VoltReadL   ;load variable with last value

            jsr   VActemp     ;Jump to subroutine that reads temperature

            lda   FrstTmpRd
            sub   AcTmpRd     ;Compare last temperature with current one
            cmp   #TmpSafe

            bhs   Charged     ;Jump to Charged if temperature increases more than
                              ;25oC.

            bra   Waitpta0

; Battery fully charged

Charged:    bclr  PTB3,PTB    ;Turn off Red LED (Battery charging)
            bset  PTB2,PTB    ;Turn on Green LED (Battery Charged)

            jsr   Trickle     ;Go to trickle current subroutine

            bra   Waitpta0
```

```
Dummytc:
      RTI

****************************************
* Vectors
****************************************

     ORG $FFDE
     DW Dummytc                  ; ADC conversion complete vector
     ORG $FFE0
     DW Dummytc                  ; Keyboard vector
     ORG $FFF2
     DW Dummytc                  ; TIM overflow vector
     ORG $FFF4
     DW Dummytc                  ; TIM Channel 1 vector
     ORG $FFF6
     DW Dummytc                  ; TIM Channel 0 vector
     ORG $FFFA
     DW Dummytc                  ; IRQ vector
     ORG $FFFC
     DW Dummytc                  ; SWI vector
     ORG $FFFE
     DW  main                    ; Reset vector

END
```

```
initCfg2:   equ   %00000000   ;Config2 Register value
;                  ||||||||   CONFIG2 is a write once register
;                  |||||||+-RSTEN   - 0 Reset function inactive in pin
;                  ||||||+--R       - 0 Reserved bit
;                  |||||+---R       - 0 Reserved bit
;                  ||||+----OSCOPT0 - 0 Set oscillator option as internal
;                  |||+-----OSCOPT1 - 0 Set oscillator option as internal
;                  ||+------R       - 0 Reserved bit
;                  |+-------IRQEN   - 0 disable IRQ function
;                  +--------IRQPUD  - 0 Internal pullup to connect IRQ and VDD

initADCH3:  equ   %00100011   ;AD configuration value
;                  ||||||||   ADC Status and Control Register
;                  |||||||+-CH0     - 1 Mux to select ADC channel
;                  ||||||+--CH1     - 1 Mux to select ADC channel
;                  |||||+---CH2     - 0 Mux to select ADC channel
;                  ||||+----CH3     - 0 Mux to select ADC channel
;                  |||+-----CH4     - 0 Channel 3 selected
;                  ||+------ADCO    - 1 Set ADC as continuous conversion
;                  |+-------AIEN    - 0 disable ADC interrupt
;                  +--------COCO    - 0 Conversions Complete Bit

initADCH2:  equ   %00000010   ;AD configuration value
;                  ||||||||   ADC Status and Control Register
;                  |||||||+-CH0     - 0 Mux to select ADC channel
;                  ||||||+--CH1     - 1 Mux to select ADC channel
;                  |||||+---CH2     - 0 Mux to select ADC channel
;                  ||||+----CH3     - 0 Mux to select ADC channel
;                  |||+-----CH4     - 0 Channel 2 selected
;                  ||+------ADCO    - 0 Set ADC as single conversion
;                  |+-------AIEN    - 0 disable ADC interrupt
;                  +--------COCO    - 0 Conversions Complete Bit

initADCH1:  equ   %00000001   ;AD configuration value
;                  ||||||||   ADC Status and Control Register
;                  |||||||+-CH0     - 1 Mux to select ADC channel
;                  ||||||+--CH1     - 0 Mux to select ADC channel
;                  |||||+---CH2     - 0 Mux to select ADC channel
;                  ||||+----CH3     - 0 Mux to select ADC channel
;                  |||+-----CH4     - 0 Channel 2 selected
;                  ||+------ADCO    - 0 Set ADC as single conversion
;                  |+-------AIEN    - 0 disable ADC interrupt
;                  +--------COCO    - 0 Conversions Complete Bit

initTim:    equ   %00110001   ;Timer Status and control Reg. value
;                  |||||||| TIM Status and Control Register
;                  |||||||+-PS0     - 1 Prescaler select bit
;                  ||||||+--PS1     - 0 Prescaler select bit
;                  |||||+---PS2     - 0 Tim clock source int. bus
;                  ||||+----0       - 0
;                  |||+-----TRST    - 1 TIM reset bit
;                  ||+------TSTOP   - 1 TIM counter stopped
;                  |+-------TOIE    - 0 disable TIM overflow interrupts
;                  +--------TOF     - 0 TIM overflow flag bit
```

```
StartTim:   equ   %00000001   ;Timer Status and control Reg. value
;                  ||||||||   TIM Status and Control Register
;                  |||||||+-PS0     - 1 Prescaler select bit
;                  ||||||+--PS1     - 0 Prescaler select bit
;                  |||||+---PS2     - 0 Tim clock source int. bus
;                  ||||+----0       - 0
;                  |||+-----TRST    - 0 TIM reset bit
;                  ||+------TSTOP   - 0 TIM counter started
;                  |+-------TOIE    - 0 disable TIM overflow interrupts
;                  +--------TOF     - 0 TIM overflow flag bit

InitDDRB:   equ   %10001110   ;PTB7 -> Pulses on Thyristor gate
                              ;PTB3 -> Red LED (Bat. Charging)
                              ;PTB2 -> Green LED (Bat. Charged)
                              ;PTB1 -> Transistor Control

InitTMODHL: equ   $1A         ;Set Timer to ~ 4ms or 1/4 Power Cycle Line (PCL)
InitTMODLL: equ   $1D         ;after we start the timer for negative edge.

InitTMODHH: equ   $0D         ;Set Timer to ~ 4ms or 1/4 Power Cycle Line (PCL)
InitTMODLH: equ   $0E         ;after we start the timer for positive edge.

initTricH:  equ   $2A         ;Set Timer to low current
initTricL:  equ   $6F         ;after we start the timer

ADclkval:   equ   %10000000   ;AD clock configuration
                              ;ADC Clock prescaler bit

GateVal:    equ   $50         ;Gate pulse duration

;Variables for counter for charge time overflow period
Counter0    rmb   1
Counter1    rmb   1           ;Time Counters

;Variables for voltage reading
VoltReadL   rmb   1
VoltReadH   rmb   1

;Variables for delay before reading battery voltage
del1        rmb   1
del2        rmb   1

;Variables for Temperature reading
AcTmpRd     rmb   1
FrstTmpRd   rmb   1

;Other Constants

BatInit     equ   $19         ;Value to identify if battery pack is connected

Del         equ   $10         ;First value for delay
Dela1       equ   $FF         ;Second value for delay
Dela2       equ   $FF         ;Third value for delay
```

```
MaskLSB     equ   $FE         ;Value to mask ADR LSB

StpChL      equ   $00         ;Low byte for stop charger period
StpChH      equ   $90         ;High byte for stop charger period

TmpSafe     equ   $06         ;Temperature value for backup

TSCClr      equ   $7F         ;Value to clear TOF bit on TSC register
```

This page is intentionally blank

This page is intentionally blank

[redacted]

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