

# Impact Measurement Using Accelerometers

by: C.S. Chua  
Sensor Application Engineering, Singapore, A/P

## INTRODUCTION

This application note describes the concept of measuring impact of an object using an accelerometer, microcontroller hardware/software and a liquid crystal display. Due to the wide frequency response of the accelerometer from d.c. to 400 Hz,

the device is able to measure both the static acceleration from the Earth's gravity and the shock or vibration from an impact. This design uses a 40g accelerometer and yields a minimum acceleration range of -40g to +40g.

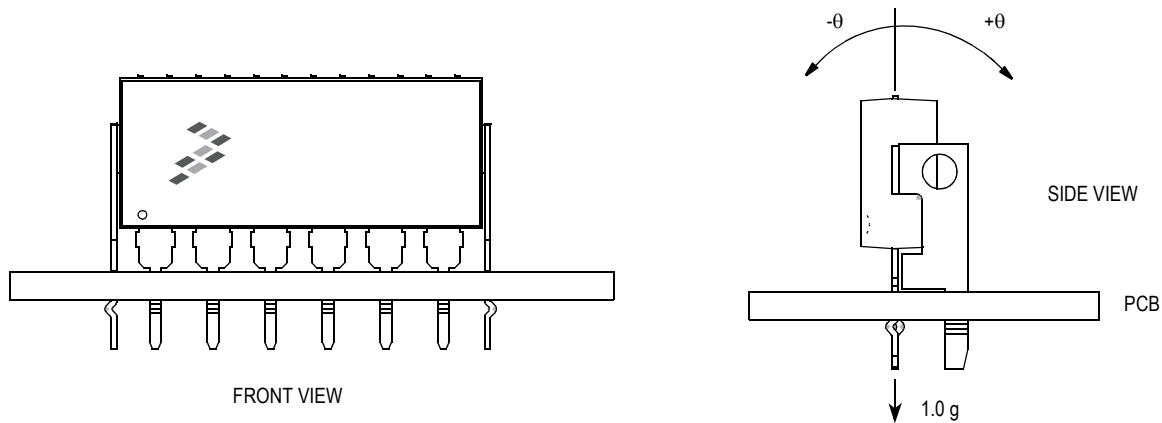
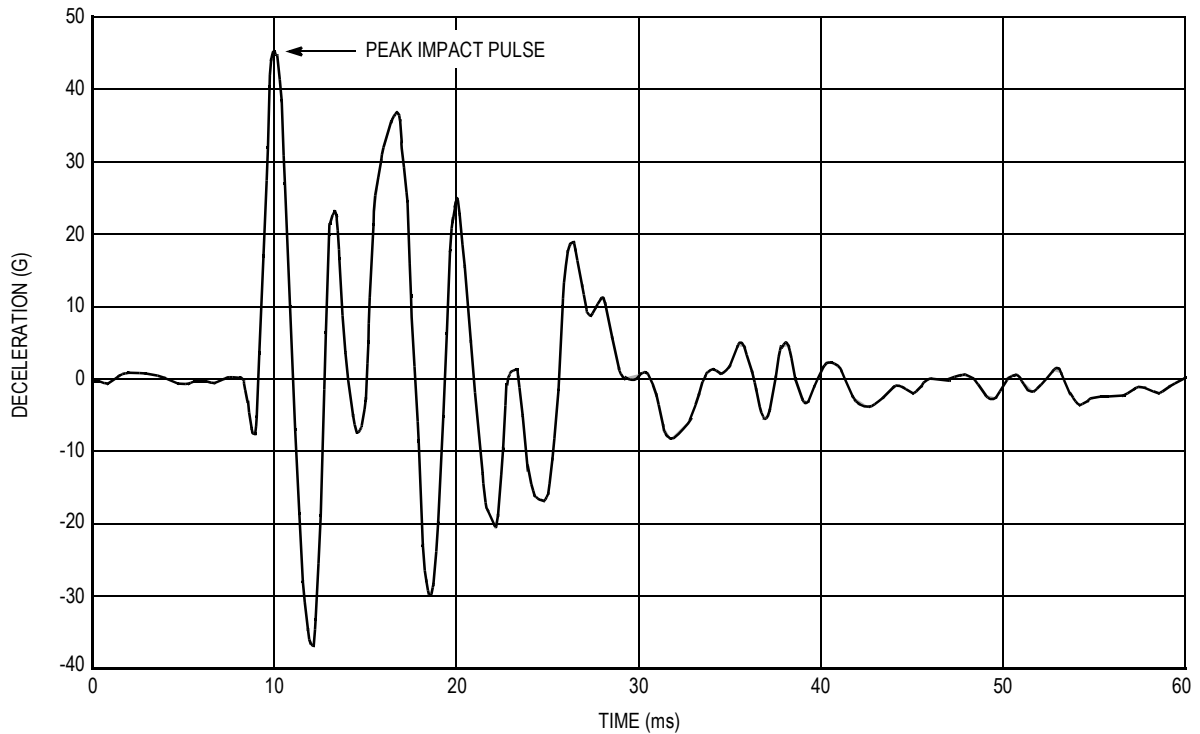


Figure 1. Orientation of Accelerometer

## CONCEPT OF IMPACT MEASUREMENT

During an impact, the accelerometer will be oriented as shown in [Figure 1](#) to measure the deceleration experienced by the object from dc to 400 Hz. Normally, the peak impact pulse is in the order of a few miniseconds. [Figure 2](#) shows a typical crash waveform of a toy car having a stiff bumper.



**Figure 2. Typical Crash Pattern**

### HARDWARE DESCRIPTION AND OPERATION

Since the accelerometer is fully signal-conditioned by its internal op-amp and temperature compensation, the output of the accelerometer can be directly interfaced with an analog-to-digital (A/D) converter for digitization. A filter consists of one RC network should be added if the connection between the output of the accelerometer and the A/D converter is a long track or cable. This stray capacitance may change the position of the internal pole which would drive the output amplifier of the accelerometer into oscillation or instability. In this design, the cut-off frequency is chosen to be 15.9 kHz which also acts as an anti-alias filter for the A/D converter. The 3 dB frequency can be approximated by the following equation.

$$f_{-3\text{db}} = \frac{1}{2\pi RC}$$

Referring to the schematic, [Figure 3](#), the accelerometer is connected to PORT D bit 5 and the output of the amplifier is connected to PORT D bit 6 of the microcontroller. This port is an input to the on-chip 8-bit analog-to-digital (A/D) converter. Typically, the accelerometer provides a signal output to the microprocessor of approximately 0.3 Vdc at -55g to 4.7 Vdc at +55g of acceleration. However, Freescale only guarantees the accuracy within  $\pm 40\text{g}$  range. Using the same reference voltage for the A/D converter and accelerometer minimizes the number of additional components, but does sacrifice resolution. The resolution is defined by the following:

$$\text{count} = \frac{V_{\text{out}}}{5} \times 255$$

The count at 0g =  $[2.5/5] \times 255 \approx 128$

The count at +25g =  $[3.5/5] \times 255 \approx 179$

The count at -25g =  $[1.5/5] \times 255 \approx 77$

Therefore the resolution 0.5g/count

The output of the accelerometer is ratiometric to the voltage applied to it. The accelerometer and the reference voltages are connected to a common supply; this yields a system that is ratiometric. By nature of this ratiometric system, variations in the voltage of the power supplied to the system will have no effect on the system accuracy.

The liquid crystal display (LCD) is directly driven from I/O ports A, B, and C on the microcontroller. The operation of a LCD requires that the data and backplane (BP) pins must be driven by an alternating signal. This function is provided by a software routine that toggles the data and backplane at approximately a 30 Hz rate. Other than the LCD, one light emitting diode (LED) are connected to the pulse length converter (PLM) of the microcontroller. This LED will light up for 3 seconds when an impact greater or equal to 7g is detected.

The microcontroller section of the system requires certain support hardware to allow it to function. The MC34064P-5 provides an undervoltage sense function which is used to reset the microprocessor at system power-up. The 4 MHz crystal provides the external portion of the oscillator function for clocking the microcontroller and provides a stable base for time bases functions, for instance calculation of pulse rate.

## SOFTWARE DESCRIPTION

Upon power-up of the system, the LCD will display CAL for approximately four seconds. During this period, the output of the accelerometer are sampled and averaged to obtain the zero offset voltage or zero acceleration. This value will be saved in the RAM which is used by the equation below to calculate the impact in term of g-force. One point to note is that the accelerometer should remain stationary during the zero calibration.

$$\text{Impact} = [\text{count} - \text{count}_{\text{offset}}] \times \text{resolution}$$

In this software program, the output of the accelerometer is calculated every 650  $\mu\text{s}$ . During an impact, the peak

deceleration is measured and displayed on the LCD for three seconds before resetting it to zero. In the mean time, if a higher impact is detected, the value on the LCD will be updated accordingly.

However, when a low g is detected (e.g. 1.0g), the value will not be displayed. Instead, more samples will be taken for further averaging to eliminate the random noise and high frequency component. Due to the fact that tilting is a low g and low frequency signal, large number of sampling is preferred to avoid unstable display. Moreover, the display value will not hold for three seconds as in the case of an impact.

Figure 4 is a flowchart for the program that controls the system.

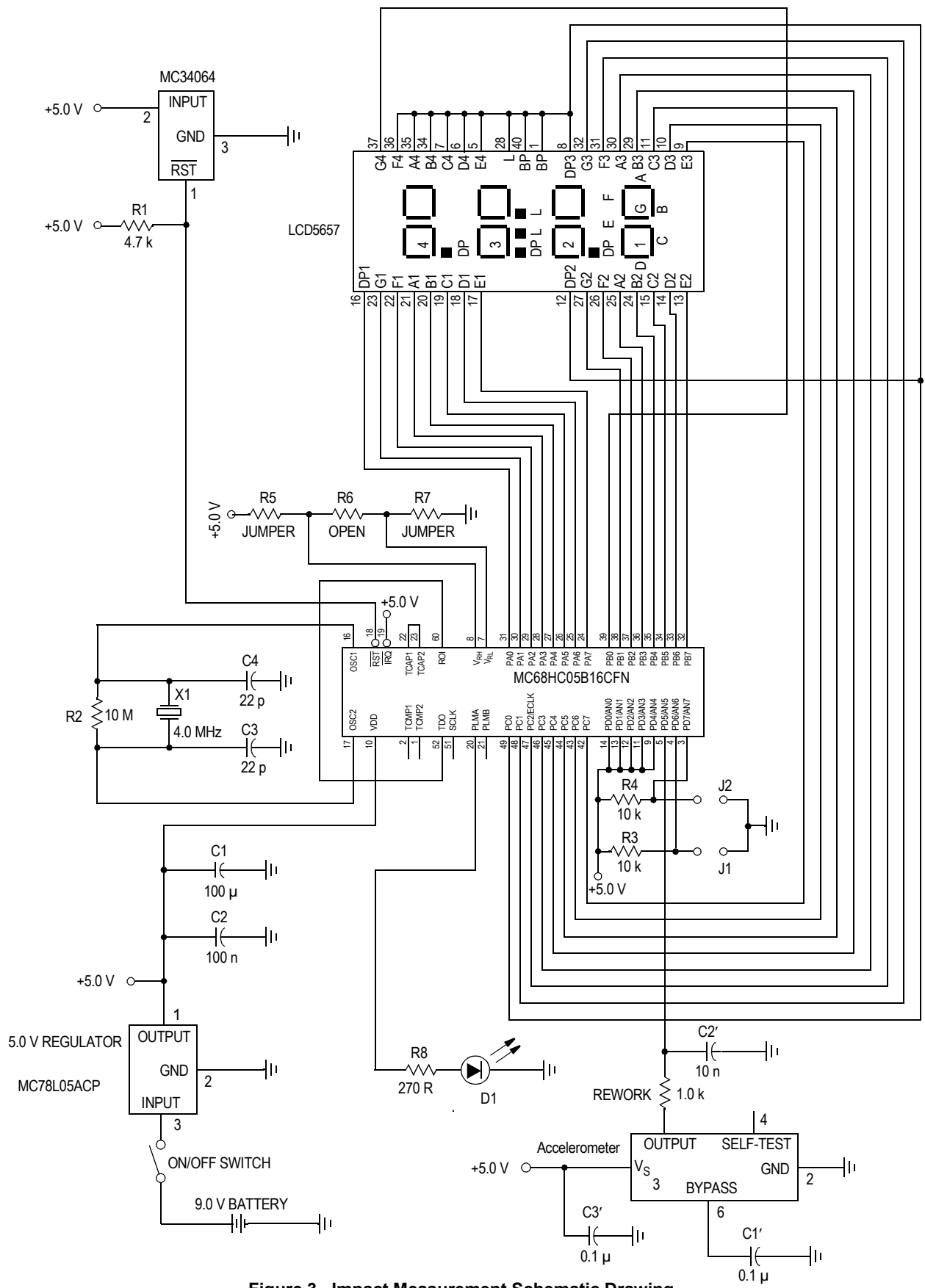


Figure 3. Impact Measurement Schematic Drawing

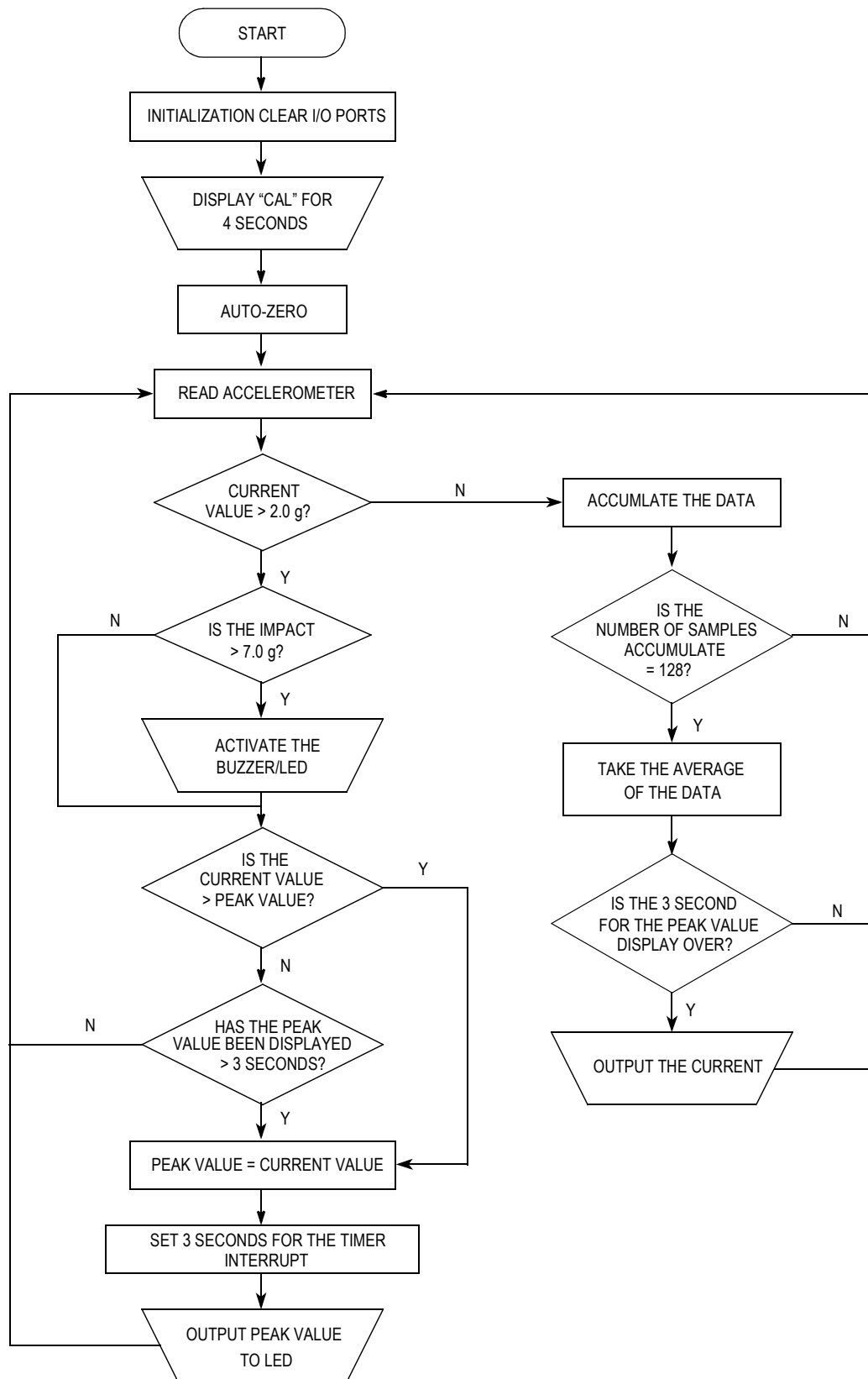


Figure 4. Main Program Flowchart

## SOFTWARE SOURCE/ASSEMBLY PROGRAM CODE

```

*****
*
*           Accelerometer Demo Car Version 2.0
*
*   The following code is written for MC68HC705B16 using MMDS05 software
*   Version 1.01
*   CASM05 - Command line assembler Version 3.04
*   P & E Microcomputer Systems, Inc.
*
*           Written by : C.S. Chua
*           29 August 1996
*
*           Copyright Freescale Electronics Pte Ltd 1996
*           All rights Reserved
*
*   This software is the property of Freescale Electronics Pte Ltd.
*
*   Any usage or redistribution of this software without the express
*   written consent of Freescale is strictly prohibited.
*
*   Freescale reserves the right to make changes without notice to any
*   products herein to improve reliability, function, or design. Freescale
*   does not assume liability arising out of the application or use of any
*   product or circuit described herein, neither does it convey license
*   under its patents rights nor the rights of others. Freescale products are
*   not designed, intended or authorised for use as component in systems
*   intended to support or sustain life or for any other application in
*   which the failure of the Freescale product could create a situation
*   a situation where personal injury or death may occur. Should the buyer
*   shall indemnify and hold Freescale products for any such unintended or
*   unauthorised application, buyer shall indemnify and hold Freescale and
*   its officers, employees, subsidiaries, affiliates, and distributors
*   harmless against all claims, costs, damages, expenses and reasonable
*   attorney fees arising out of, directly or indirectly, any claim of
*   personal injury or death associated with such unintended or unauthorised
*   use, even if such claim alleges that Freescale was negligent regarding
*   the design or manufacture of the part.
*
*   Freescale and the Freescale logo are registered trademarks of Freescale Inc.*
*
*   Freescale Inc. is an equal opportunity/affirmative action employer.
*
*****
*****
*
*           Software Description
*
*   This software is used to read the output of the accelerometer MMA2200W
*   and display it to a LCD as gravity force. It ranges from -55g to +55g
*   with 0g as zero acceleration or constant velocity. The resolution is
*   0.5g.
*
*   The program will read from the accelerometer and hold the maximum
*   deceleration value for about 3.0 seconds before resetting. At the same
*   time, the buzzer/LED is activated if the impact is more than 7.0g.
*   However, if the maximum deceleration changes before 3.0 seconds, it
*   will update the display using the new value. Note that positive value
*   implies deceleration whereas negative value implies acceleration
*
*****
*****
*
*           Initialisation
*
*****
PORTA      EQU      $00      ; Last digit

PORTB      EQU      $01      ; Second digit (and negative sign)
PORTC      EQU      $02      ; First digit (and decimal point)
ADDATA     EQU      $08      ; ADC Data
ADSTAT     EQU      $09      ; ADC Status
PLMA       EQU      $0A      ; Pulse Length Modulator (Output to Buzzer)
MISC       EQU      $0C      ; Miscellaneous Register (slow/fast mode)
TCONTROL   EQU      $12      ; Timer control register
TSTATUS    EQU      $13      ; Timer Status Register
OCMPH11    EQU      $16      ; Output Compare Register 1 High Byte

```

```

OCMPLO1    EQU    $17    ; Output Compare Register 1 Low Byte
TCNTHI     EQU    $18    ; Timer Count Register High Byte
TCNTLO     EQU    $19    ; Timer Count Register Low Byte
OCMPHI2    EQU    $1E    ; Output Compare Register 2 High Byte
OCMPLO2    EQU    $1F    ; Output Compare Register 2 Low Byte
*****
*
*          User-defined RAM
*
*****
SIGN        EQU    $54    ; Acceleration (-) or deceleration (+)
PRESHI2    EQU    $55    ; MSB of accumulated acceleration
PRESHI     EQU    $56
PRESLO     EQU    $57    ; LSB of accumulated acceleration
PTEMPHI    EQU    $58    ; Acceleration High Byte (Temp storage)
PTEMPLO    EQU    $59    ; Acceleration Low Byte (Temp storage)
ACCHI      EQU    $5A    ; Temp storage of acc value (High byte)
ACCLO      EQU    $5B    ; (Low byte)
ADCOUNTER  EQU    $5C    ; Sampling Counter
AVERAGE_H EQU    $5D    ; MSB of the accumulated data of low g
AVERAGE_M EQU    $5E
AVERAGE_L EQU    $5F    ; LSB of the accumulated data of low g
SHIFT_CNT  EQU    $60    ; Counter for shifting the accumulated data
AVE_CNT1   EQU    $61    ; Number of samples in the accumulated data
AVE_CNT2   EQU    $75
TEMPTCNTHI EQU    $62    ; Temp storage for Timer count register
TEMPTCNTLO EQU    $63    ; Temp storage for Timer count register
DECHI      EQU    $64    ; Decimal digit high byte
DECLO      EQU    $65    ; Decimal digit low byte
DCOFFSETHI EQU    $66    ; DC offset of the output (high byte)
DCOFFSETLO EQU    $67    ; DC offset of the output (low byte)
MAXACC     EQU    $68    ; Maximum acceleration
TEMPHI     EQU    $69
TEMPLO     EQU    $6A
TEMP1      EQU    $6B    ; Temporary location for ACC during delay
TEMP2      EQU    $6C    ; Temporary location for ACC during ISR
DIV_LO     EQU    $6D    ; No of sampling (low byte)
DIV_HI     EQU    $6E    ; No of sampling (high byte)
NO_SHIFT   EQU    $6F    ; No of right shift to get average value
ZERO_ACC   EQU    $70    ; Zero acceleration in no of ADC steps
HOLD_CNT   EQU    $71    ; Hold time counter
HOLD_DONE  EQU    $72    ; Hold time up flag
START_TIME EQU    $73    ; Start of count down flag
RSHIFT     EQU    $74    ; No of shifting required for division
ORG        $300
DB         $FC    ; Display "0"
DB         $30    ; Display "1"
DB         $DA    ; Display "2"
DB         $7A    ; Display "3"
DB         $36    ; Display "4"
DB         $6E    ; Display "5"
DB         $EE    ; Display "6"
DB         $38    ; Display "7"
DB         $FE    ; Display "8"
DB         $7E    ; Display "9"
HUNDREDHI  DB         $00    ; High byte of hundreds
HUNDREDLO  DB         $64    ; Low byte of hundreds
TENHI      DB         $00    ; High byte of tens
TENLO      DB         $0A    ; Low byte of tens
*****
*
*   Program starts here upon hard reset
*
*****
RESET      CLR     PORTC    ; Port C = 0
           CLR     PORTB    ; Port B = 0
           CLR     PORTA    ; Port A = 0
           LDA     #$FF
           STA     $06    ; Port C as output
           STA     $05    ; Port B as output
           STA     $04    ; Port A as output
           LDA     TSTATUS  ; Dummy read the timer status register
           CLR     OCMPII2  ; so as to clear the OCF
           CLR     OCMPII1
           LDA     OCMPII2
           JSR     COMPRGT
           CLR     START_TIME

```

```

LDA    #$40          ; Enable the output compare interrupt
STA    TCONTROL
CLI    ; Interrupt begins here
LDA    #$CC          ; Port C = 1100 1100 Letter "C"
STA    PORTC
LDA    #$BE          ; Port B = 1011 1110 Letter "A"
STA    PORTB
LDA    #$C4          ; Port A = 1100 0100 Letter "L"
STA    PORTA
LDA    #16
IDLE   JSR    DLY20    ; Idling for a while (16*0.125 = 2 sec)
DECA   ; for the zero offset to stabilize
BNE    IDLE          ; before perform auto-zero
LDA    #$00          ; Sample the data 32,768 times and take
STA    DIV_LO        ; the average 8000 H = 32,768
LDA    #$80          ; Right shift of 15 equivalent to divide
STA    DIV_HI        ; by 32,768
LDA    #!15         ; Overall sampling time = 1.033 s)
STA    NO_SHIFT
JSR    READAD        ; Zero acceleration calibration
LDX    #5            ; Calculate the zero offset
LDA    PTEMPLO       ; DC offset = PTEMPLO * 5
STA    ZERO_ACC
MUL
STA    DCOFFSETLO    ; Save the zero offset in the RAM
TXA
STA    DCOFFSETHI
CLR    HOLD_CNT
LDA    #$10          ; Sample the data 16 times and take
STA    DIV_LO        ; the average 0100 H = 16
LDA    #$00          ; Right shift of 4 equivalent to divide
STA    DIV_HI        ; by 16
LDA    #$4           ; Overall sampling time = 650 us
STA    NO_SHIFT
LDA    ZERO_ACC      ; Display 0.0g at the start
STA    MAXACC
JSR    ADTOLCD
CLR    START_TIME
CLR    AVE_CNT1
CLR    AVE_CNT2
CLR    SHIFT_CNT
CLR    AVERAGE_L
CLR    AVERAGE_M
CLR    AVERAGE_H
REPEAT JSR    READAD    ; Read acceleration from ADC
LDA    ZERO_ACC
ADD    #$04
CMP    PTEMPLO
BLO    CRASH         ; If the acceleration < 2.0g
LDA    PTEMPLO       ; Accumulate the averaged results
ADD    AVERAGE_L    ; for 128 times and take the averaging
STA    AVERAGE_L    ; again to achieve more stable
CLRA   ; reading at low g
ADC    AVERAGE_M
STA    AVERAGE_M
CLRA
ADC    AVERAGE_H
STA    AVERAGE_H
LDA    #$01
ADD    AVE_CNT1
STA    AVE_CNT1
CLRA
ADC    AVE_CNT2
STA    AVE_CNT2
CMP    #$04
BNE    REPEAT
LDA    AVE_CNT1
CMP    #$00
BNE    REPEAT
SHIFTING INC  SHIFT_CNT ; Take the average of the 128 samples
LSR    AVERAGE_H
ROR    AVERAGE_M
ROR    AVERAGE_L
LDA    SHIFT_CNT
CMP    #$0A
BLO    SHIFTING
LDA    AVERAGE_L

```



```

        STA     PTEMPLO
        LDA     HOLD_CNT      ; Check if the hold time of crash data
        CMP     #$00         ; is up
        BNE     NON-CRASH
        LDA     PTEMPLO      ; If yes, display the current acceleration
        STA     MAXACC       ; value
        JSR     ADTOLCD
        BRA     NON-CRASH
CRASH   LDA     ZERO_ACC
        ADD     #$0E         ; If the crash is more than 7g
        CMP     PTEMPLO      ; 7g = 0E H * 0.5
        BHS     NO_INFLATE
        LDA     #$FF        ; activate the LED
        STA     PLMA
NO_INFLATE JSR     MAXVALUE   ; Display the peak acceleration
        JSR     ADTOLCD
NON-CRASH CLR     SHIFT_CNT
        CLR     AVE_CNT1
        CLR     AVE_CNT2
        CLR     AVERAGE_L
        CLR     AVERAGE_M
        CLR     AVERAGE_H
        BRA     REPEAT      ; Repeat the whole process
*****
*                               *
*      Delay Subroutine         *
*      (162 * 0.7725 ms = 0.125 sec) *
*                               *
*****
DLY20   STA     TEMP1
        LDA     #!162       ; 1 unit = 0.7725 ms
OUTLP   CLRX
INNRLP  DECX
        BNE     INNRLP
        DECA
        BNE     OUTLP
        LDA     TEMP1
        RTS
*****
*                               *
*      Reading the ADC data X times *
*      and take the average        *
*      X is defined by DIV_HI and DIV_LO *
*                               *
*****
READAD  LDA     #$25
        STA     ADSTAT      ; AD status = 25H
        CLR     PRESHI2
        CLR     PRESHI     ; Clear the memory
        CLR     PRESLO
        CLRX
        CLR     ADCOUNTER
LOOP128 TXA
        CMP     #$FF
        BEQ     INC_COUNT
        BRA     CONT
INC_COUNT INC    ADCOUNTER
CONT    LDA     ADCOUNTER   ; If ADCOUNTER = X
        CMP     DIV_HI     ; Clear bit = 0
        BEQ     CHECK_X    ; Branch to END100
        BRA     ENDREAD
CHECK_X TXA
        CMP     DIV_LO
        BEQ     END128
ENDREAD BRCLR   7,ADSTAT,ENDREAD ; Halt here till AD read is finished
        LDA     ADDATA     ; Read the AD register
        ADD     PRESLO     ; PRES = PRES + ADDATA
        STA     PRESLO
        CLRA
        ADC     PRESHI
        STA     PRESHI
        CLRA
        ADC     PRESHI2
        STA     PRESHI2
        INCX           ; Increase the AD counter by 1
        BRA     LOOP128   ; Branch to Loop128
END128  CLR     RSHIFT     ; Reset the right shift counter

```

```

DIVIDE    INC    RSHIFT          ; Increase the right counter
          LSR    PRESHI2
          ROR    PRESHI          ; Right shift the high byte
          ROR    PRESLO          ; Right shift the low byte
          LDA    RSHIFT
          CMP    NO_SHIFT        ; If the right shift counter >= NO_SHIFT
          BHS    ENDDIVIDE      ; End the shifting
          JMP    DIVIDE          ; otherwise continue the shifting
ENDDIVIDE LDA    PRESLO
          STA    PTEMPLO
          RTS

*****
*
*      Timer service interrupt
*      Alternates the Port data and
*      backplane of LCD
*
*****
TIMERCMP  STA    TEMP2           ; Push Accumulator
          COM    PORTC           ; Port C = - (Port C)
          COM    PORTB          ; Port B = - (Port B)
          COM    PORTA          ; Port A = - (Port A)
          LDA    START_TIME     ; Start to count down the hold time
          CMP    #$FF           ; if START_TIME = FF
          BNE    SKIP_TIME
          JSR    CHECK_HOLD
SKIP_TIME BSR    COMPRT         ; Branch to subroutine compare register
          LDA    TEMP2           ; Pop Accumulator
          RTI

*****
*
*      Check whether the hold time
*      of crash impact is due
*
*****
CHECK_HOLD DEC   HOLD_CNT
          LDA    HOLD_CNT
          CMP    #$00           ; Is the hold time up?
          BNE    NOT_YET
          LDA    #$00           ; If yes,
          STA    PLMA           ; stop buzzer
          LDA    #$FF           ; Set HOLD_DONE to FF indicate that the
          STA    HOLD_DONE      ; hold time is up
          CLR    START_TIME     ; Stop the counting down of hold time
NOT_YET   RTS

*****
*
*      Subroutine reset
*      the timer compare register
*
*****
COMPRT    LDA    TCNTHI         ; Read Timer count register
          STA    TEMPTCNTHI     ; and store it in the RAM
          LDA    TCNTLO
          STA    TEMPTCNTLO
          ADD    #$4C           ; Add 1D4C H = 7500 periods
          STA    TEMPTCNTLO     ; with the current timer count
          LDA    TEMPTCNTHI     ; 1 period = 2 us
          ADC    #$1D
          STA    TEMPTCNTHI     ; Save the next count to the register
          STA    OCMPHI1
          LDA    TSTATUS        ; Clear the output compare flag
          LDA    TEMPTCNTLO     ; by access the timer status register
          STA    OCMPL01        ; and then access the output compare
          RTS                   ; register

*****
*
*      Determine which is the next
*      acceleration value to be display
*
*****
MAXVALUE  LDA    PTEMPLO
          CMP    MAXACC         ; Compare the current acceleration with
          BLS    OLDMAX         ; the memory, branch if it is <= maxacc
          BRA    NEWMAX1
OLDMAX    LDA    HOLD_DONE      ; Decrease the Holdtime when
          CMP    #$FF           ; the maximum value remain unchanged

```

```

        BEQ     NEWMAX1      ; Branch if the Holdtime is due
        LDA     MAXACC       ; otherwise use the current value
        BRA     NEWMAX2
NEWMAX1 LDA     #$C8         ; Hold time = 200 * 15 ms = 3 sec
        STA     HOLD_CNT    ; Reload the hold time for the next
        CLR     HOLD_DONE   ; maximum value
        LDA     #$FF
        STA     START_TIME  ; Start to count down the hold time
        LDA     PTEMPLO     ; Take the current value as maximum
NEWMAX2 STA     MAXACC
        RTS

*****
*
*   This subroutine is to convert
*   the AD data to the LCD
*   Save the data to be diplayed
*   in MAXACC
*
*****
ADTOLCD SEI     ; Disable the Timer Interrupt !!
        LDA     #$00       ; Load 0000 into the memory
        STA     DECHI
        LDA     #$00
        STA     DECL0
        LDA     MAXACC
        LDX     #5
        MUL                    ; Acceleration = AD x 5
        ADD     DECL0      ; Acceleration is stored as DECHI
        STA     DECL0      ; and DECL0
        STA     ACCLO      ; Temporary storage
        LDA     #$00       ; Assume positive deceleration
        STA     SIGN       ; "00" positive ; "01" negative
        CLRA
        TXA
        ADC     DECHI
        STA     DECHI
        STA     ACCHI      ; Temporary storage
        LDA     DECL0
        SUB     DCOFFSETLO ; Deceleration = Dec - DC offset
        STA     DECL0
        LDA     DECHI
        SBC     DCOFFSETHI
        STA     DECHI
        BCS     NEGATIVE   ; Branch if the result is negative
        BRA     SEARCH
NEGATIVE LDA     DCOFFSETLO ; Acceleration = DC offset - Dec
        SUB     ACCLO
        STA     DECL0
        LDA     DCOFFSETHI
        SBC     ACCHI
        STA     DECHI
        LDA     #$01       ; Assign a negative sign
        STA     SIGN
SEARCH   CLRX              ; Start the search for hundred digit
LOOP100 LDA     DECL0      ; Acceleration = Acceleration - 100
        SUB     HUNDREDLO
        STA     DECL0
        LDA     DECHI
        SBC     HUNDREDHI
        STA     DECHI
        INCX              ; X = X + 1
        BCC     LOOP100   ; if acceleration >= 100, continue the
        DECX              ; loop100, otherwise X = X - 1
        LDA     DECL0      ; Acceleration = Acceleration + 100
        ADD     HUNDREDLO
        STA     DECL0
        LDA     DECHI
        ADC     HUNDREDHI
        STA     DECHI
        TXA              ; Check if the MSD is zero
        AND     #$FF
        BEQ     NOZERO    ; If MSD is zero, branch to NOZERO
        LDA     $0300,X    ; Output the first second digit
        STA     PORTC
        BRA     STARTTEN
NOZERO  LDA     #$00       ; Display blank if MSD is zero
        STA     PORTC

```

```

STARTTEN  CLRX                ; Start to search for ten digit
LOOP10    LDA  DECLO          ; acceleration = acceleration - 10
          SUB  TENLO
          STA  DECLO
          LDA  DECHI
          SEC  TENHI
          STA  DECHI
          INCX
          BCC  LOOP10        ; if acceleration >= 10 continue the
          DECX              ; loop, otherwise end
          LDA  DECLO          ; acceleration = acceleration + 10
          ADD  TENLO
          STA  DECLO
          LDA  DECHI
          ADC  TENHI
          STA  DECHI
          LDA  $0300,X        ; Output the last second digit
          EOR  SIGN          ; Display the sign
          STA  PORTB
          CLRX              ; Start to search for the last digit
          LDA  DECLO          ; declo = declo - 1
          TAX
          LDA  $0300,X        ; Output the last digit
          EOR  #$01          ; Add a decimal point in the display
          STA  PORTA
          CLI                ; Enable Interrupt again !
          RTS

```

\*\*\*\*\*

```

*
*   This subroutine provides services
*   for those unintended interrupts
*
*
*****

```

```

SWI      RTI                ; Software interrupt return
IRQ      RTI                ; Hardware interrupt
TIMERCAP RTI                ; Timer input capture
TIMERROV RTI                ; Timer overflow
SCI      RTI                ; Serial communication Interface
          ; Interrupt
          ORG  $3FF2        ; For 68HC05B16, the vector location
          FDB  SCI          ; starts at 3FF2
          FDB  TIMERROV     ; For 68HC05B5, the address starts
          FDB  TIMERCOMP    ; 1FF2
          FDB  TIMERCAP
          FDB  IRQ
          FDB  SWI
          FDB  RESET

```

---

**NOTES**

---

**NOTES**

---

**NOTES**

## **How to Reach Us:**

**Home Page:**  
www.freescale.com

**E-mail:**  
support@freescale.com

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

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

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

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

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

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

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

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

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

