

## A Tilt Detector

### **Abstract**

This project completes a digital tilt detector based on 80HC908QT4. Although this detector is small, it can do lots of things. It can measure  $360^\circ$  of tilt, and display the result on the LCD. It can store the tilt data in its body without worrying about power failure, and show these history data on the screen. It can talk with PC and do what PC is required, or send stored data to PC by a key pressing. In order to improve the measurement accuracy, this gadget can calibrate the sensor and save the calibration parameters for better performance. It also can measure the acceleration in range from -2g to +2g. All these functions are completed by the tiny 8 pin microcontroller - 80HC908QT4.

This device is handy to operate with only two buttons. Pressing the Fkey, user can select one of the 5 function modes, which are tilt detecting, stored data browsing, sensor calibrating, PC co-working, and acceleration measuring. In each function mode, Skey completes different tasks, such as data saving, parameters confirming, data sending, and etc.

There is an easy reading numerical LCD in this detector. All working status of this device can be read out from it, such as the tilt angle value, history data, acceleration and etc. Lots of prompts are displayed on the LCD for user's handy operation. Owing to the serial techniques, this LCD is controlled by only two pins.

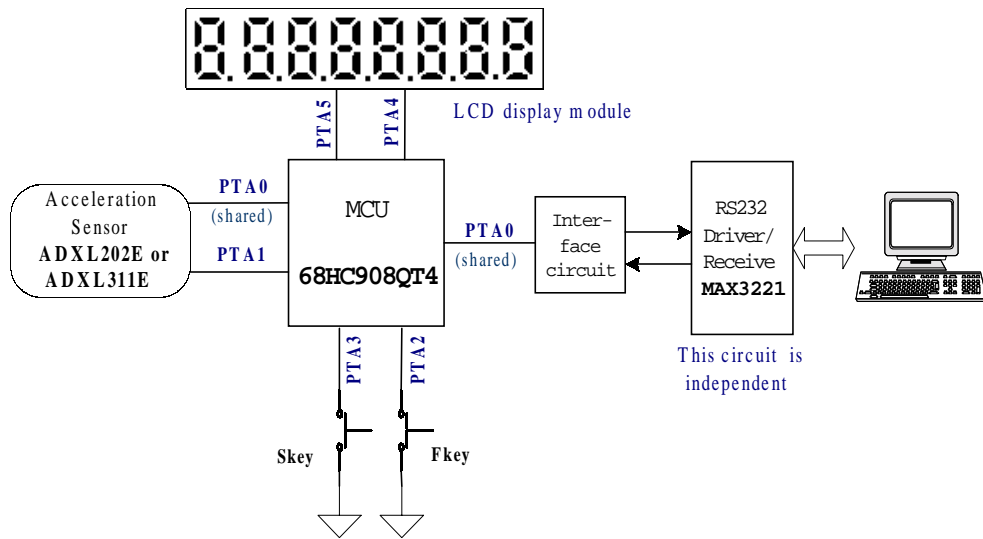
Most features of 80HC908QT4 are dug out in this application, such as ADC module for analog acceleration signals conversion, input capture function for sensor's digital outputs decoding, key board interrupt for button handling, powerful I/O pins for LCD driving and serial communication, flash for calibration parameters and tilt data saving, and low power mode software system for energy saving. An ADC channel and serial communication share one I/O pin - PTA0. A jumper is employed to switch between them. Additionally, some subroutines in 908QT4's ROM are used to simplify the program.

The software is driven by keyboard, ADC or input capture and timer overflow interrupts. When the 908QT4 is wakened up by interrupts, it can do the routine work of the selected function mode. After that, the MCU goes to sleep again. Accordingly, in most of the time the MCU stays in low power mode (WAIT). But it can complete all missions effectively. Even a single battery can power the device for a long time.

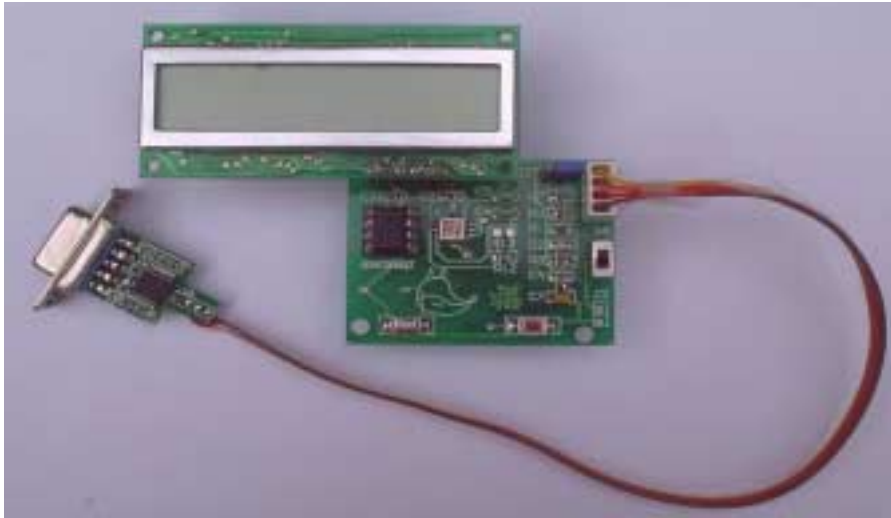
In this project, the tilt angle is derived from the acceleration in two directions through gravity. A smart algorithm is worked out to simplify the converting calculation and improve the precision. With this skill, satisfied resolution can be obtained from 0 to 360 degree.

This detector is a useful cute gadget. It can be used in many applications. There are two versions of the circuit. One is digital version which is powered by input capture function with high accuracy; the other is an analog version which is powered by the inner ADC with low cost.

### Block Diagram



### ***Photos of the circuit***



**Whole project view**



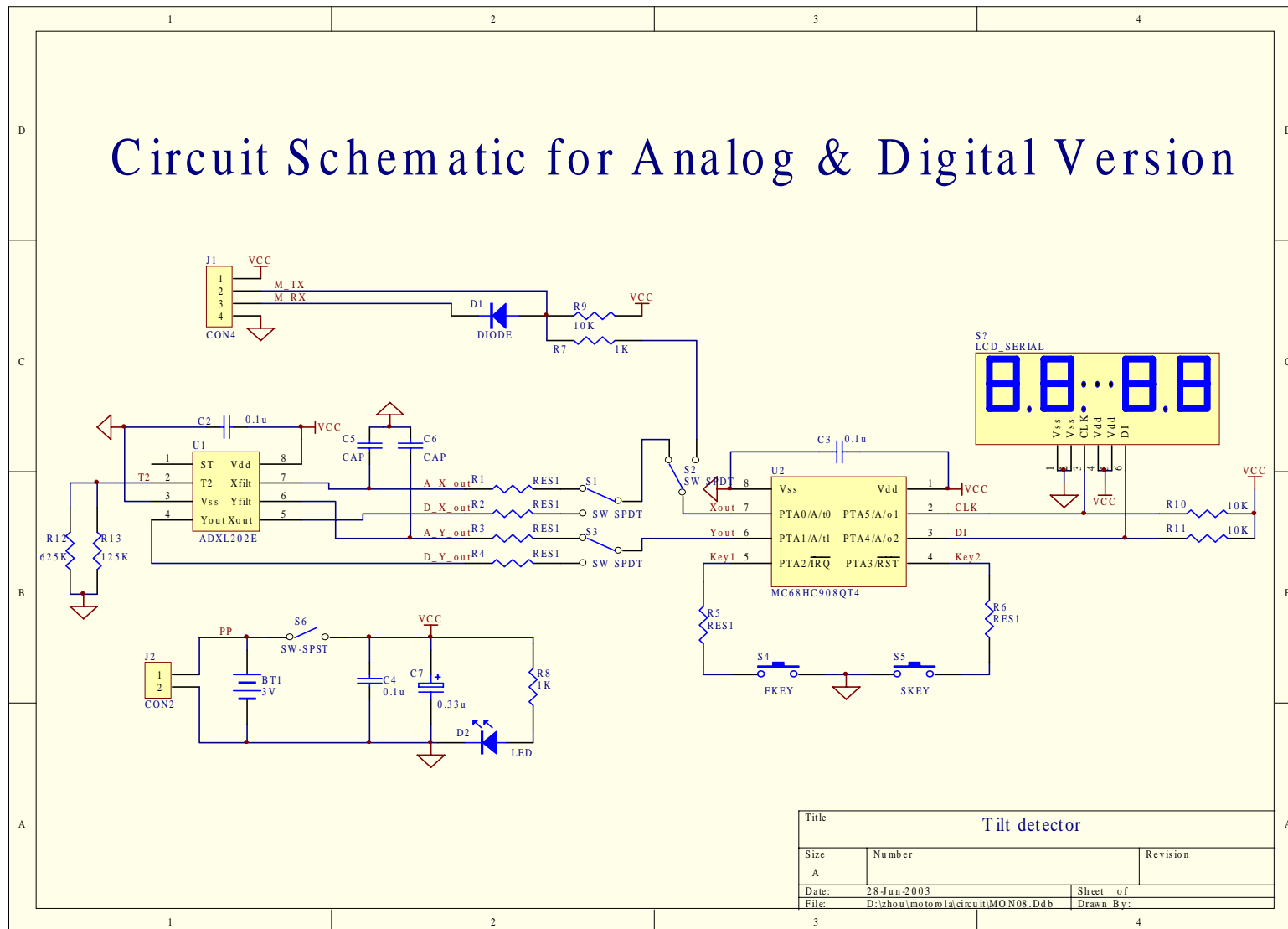
**Run in hand**

### ***Circuit schematic chart***

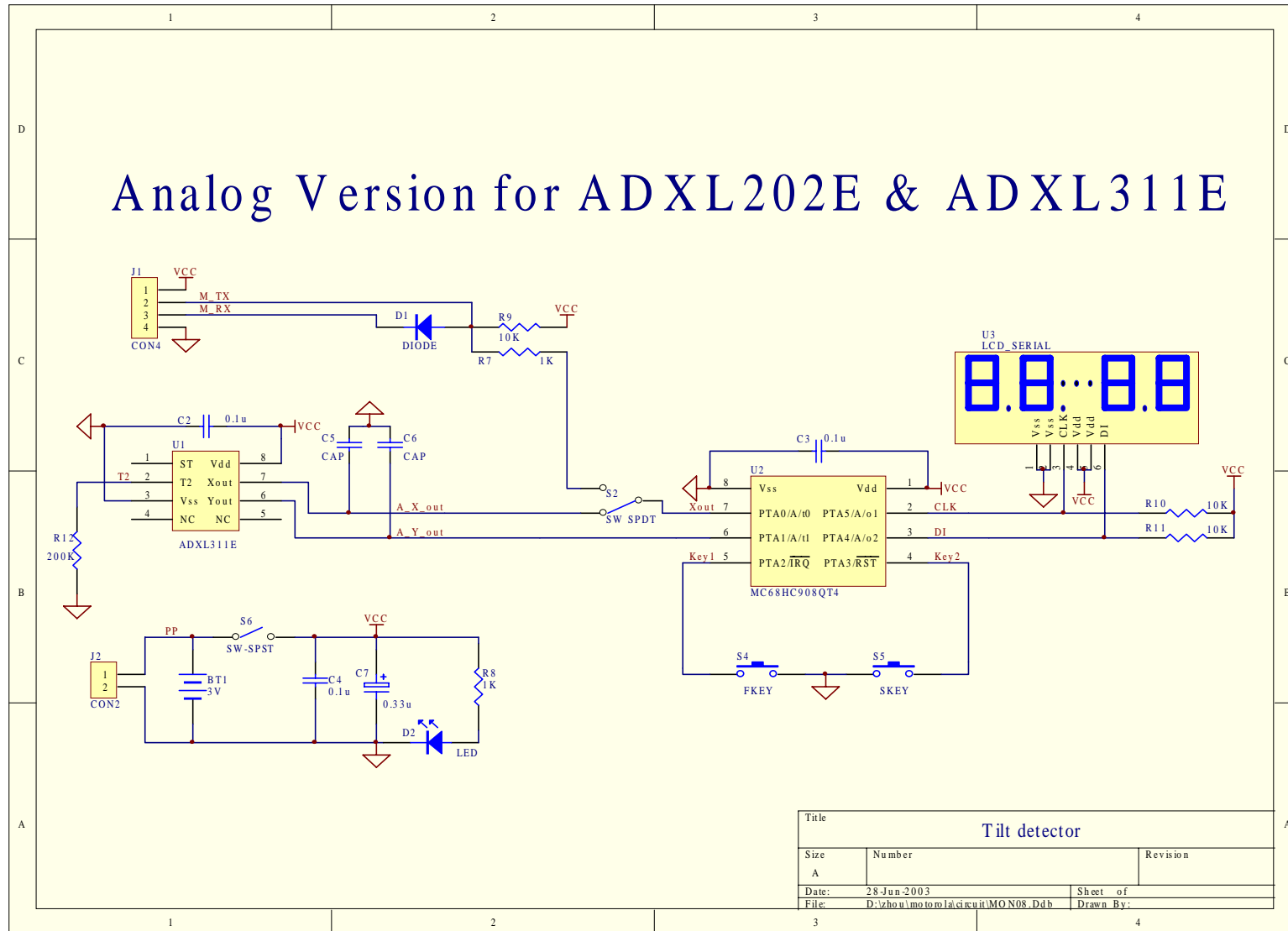
In order to try the ADC function and input capture function of 908QT4. A printed circuit board is carefully designed for both analog and digital version. Configuring the resistors and jumpers correctly, you can get either analog or digital version circuit.

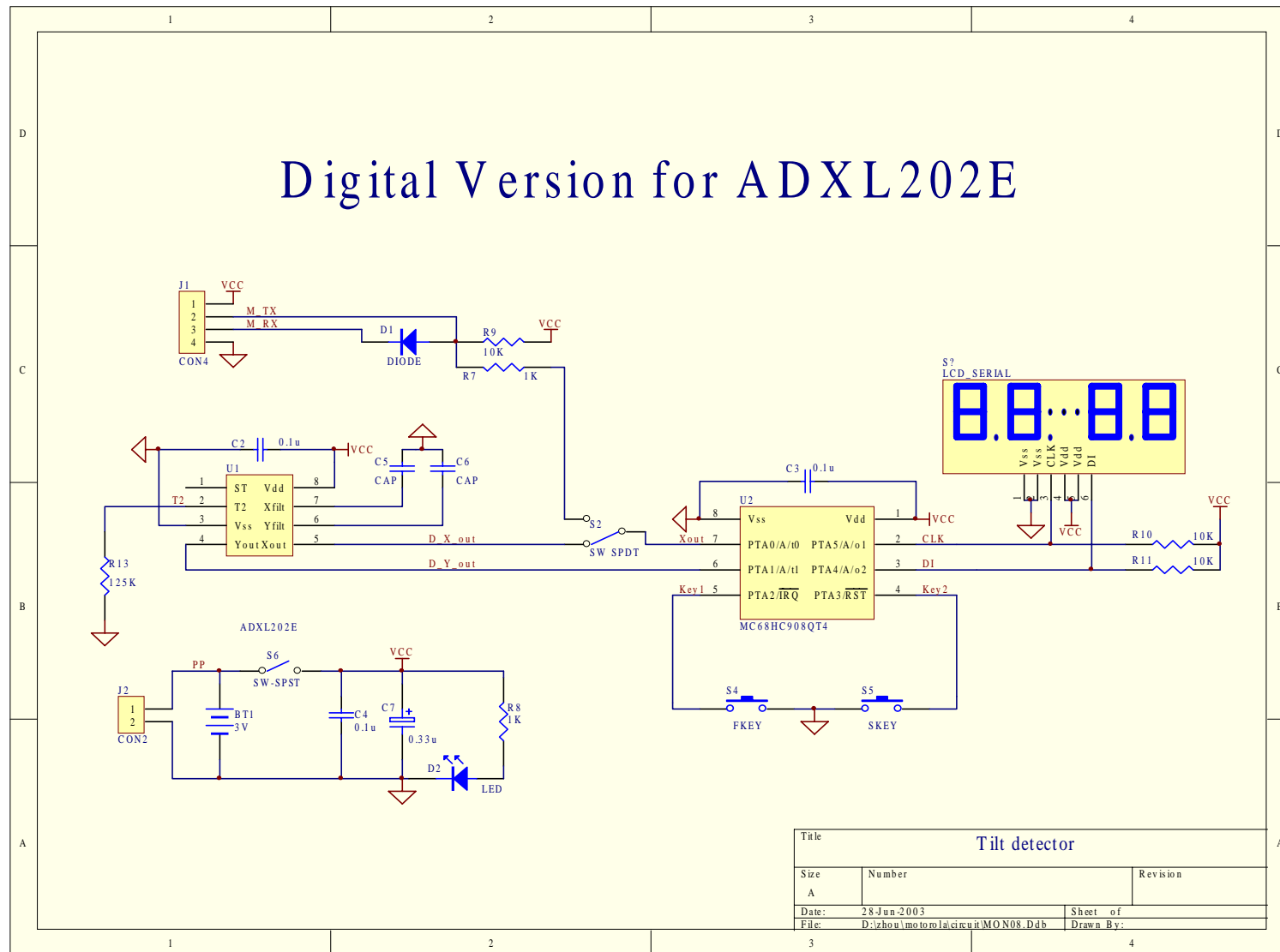
In order to explain the schematic chart clearly, three charts are listed here. The first is the chart used to make the printed circuit board (show in the photos) for both version. The second chart illustrates the configuration of the analog version. The third demonstrates the configuration of the digital version.

# Circuit Schematic for Analog & Digital Version



Title		
Tilt detector		
Size	Number	Revision
A		
Date:	28 Jun-2003	Sheet of
File:	D:\zhou\moto\mla\circuit\MON08.Ddb	Drawn By:





**Software code samples**

```

;*****
;*      Flsh_Erase_Page: Erase a flash page
;*      Entry parameter table
;*      1)      H:X          The any address in the erasing flash page
;*      2)      $0088       The control byte address in RAM
;*                                     if the 6th bit =1 : erase all
;*                                     if the 6th bit =0 : erase a flash page
;* A subroutine (ERAGNGE) in ROM is invoked
;* The entry address is $2A71 (verified in 908QY4, 908QT4)
;*****
Flash_Erase_Page:
    LDHX    #FPARASADD+2      ; The any address in the erasing page
    BCLR   Bit6,FlashEraseCon ; Erase a pages, FlashEraseCon --$88 in ROM
    JSR    ERARNGE           ; Call the subroutine in the ROM
    RTS

;*****
;* Pre_Calculate calculate the parameters (Xuni and Yuni) for the angle conversion
;* Xuni = (|x-X0g|/X1g) * factor (factor = 32)
;* Yuni = (|y-Y0g|/Y1g) * factor (factor = 32)
;* Through this work,the Xuni & Yuni are in same scale and can be compared
;* Inputs: adc_result_x, adc_result_y
;* Outputs: Xabs,Yabs,Xuni, Yuni
;* Used RAM registers: Temp, X1g, Y1g
;*****
Pre_Calculate:
    PSHA
    CLRH
    LDA    adc_result_x      ; Load x_result to A
    LDX    X0g               ; Load x0 to X
    JSR    ABS               ; |x-x0|
    STA    Xabs
    LDX    X1g               ; Load the divisor
    JSR    Unity             ; (Xabx * 32) / X1g
    STA    Xuni
    LDA    adc_result_y      ; Loda y_result to A
    LDX    Y0g               ; Load y0 to X
    JSR    ABS               ; |y-y0|
    STA    Yabs
    LDX    Y1g               ; Load the divisor
    JSR    Unity             ; (Xabx * 32) / X1g
    STA    Yuni
    PULA
    RTS

;*****
;* ABS calculate the absolute value A=|A-X|
;* input: A,X
;* output: A
;* used RAM variable : Temp
;*****
ABS:
    STX    Temp              ; X -> Temp
    CMP    Temp              ; A >= X?
    BHI    ABS_plus          ; if A<X,the A<->X
    PSHA
    LDA    Temp              ; Save the A in the stack
    PULX
    RTS

```

```

    STX          Temp          ; X -> temp (A->temp)
ABS_plus:
    SUB          Temp          ; A>=X, A=A-X
    RTS

;*****
;* Unite the different value of two axes eg: (Xabs / X1g) * 32
;* 32 is a unity factor. This factor can be changed as required
;* Through this calculation, the values between X and Y can be compared.
;* The formula is A = A*32/X
;* input: A,X , A = Xabs or Y abs, X= Xabs or Yabs
;* output: A
;* used RAM variable : Temp
;*****
Unity:
    STX          Temp          ; Save the X -> Temp
    LDX          #32
    MUL          ; A * 32 -> X:A
    PSHX
    PULH          ; X -> H, (A*32)-> H:A
    LDX          Temp
    DIV          ; (H:A) / X -> A
    RTS

;*****
;* Function_Routine: Complete the routine work in each functional mode
;* When the MCU is wakened up by the interrupts, this routine would be run at
;* once. All the subroutine in this routine is grouped by the function.
;* The "Index" indicates which functional mode the program should be
;* enter
;*****
Function_Routine:
    CLRH
    LDX  Index   ; Load the Index number
    LSLX          ; X = X * 2
    JMP  Function_Branch, X
Function_Branch:
    BRA  Fun1
    BRA  Fun2
    BRA  Fun3
    BRA  Fun4
    BRA  Fun5
Fun1:   ; Function mode ( Tilt detector)
    JSR  Tilt_Detect
    MOV  #20,timeout1          ; the frequency of A/D is decided by timeout1
    BRA  FR_over
Fun2:   ; Function mode ( Data Browse)
    JSR  Data_Browse          ; Browse the stored data in the Flash
    BRA  FR_over
Fun3:   ; Function mode ( Parameters calibration)
    JSR  Parameter_Calibration
    MOV  #20,timeout1          ; the frequency of A/D is decided by timeout1
    BRA  FR_over
Fun4:   ; Function mode ( Parameters Browse)
    JSR  Parameters_Browse    ; Browse the stored data in the Flash
    BRA  FR_over
Fun5:   ; Function mode
    JSR  Display_G
    MOV  #20,timeout1          ; the frequency of A/D is decided by timeout1
FR_over: RTS

```