APPLICATION OPERATION

PIC12CXXX microcontroller can be used in quite an unexpected area of application - as an intelligent metal detector. Few components are needed to build a hand held (stand alone) or a static, remotely controlled metal sensor connected to a computer or another microcontroller.

As known, the metal objects can change the resonant frequency of an LC circuit. If this circuit is connected to oscillator inputs of PIC, then the operating speed of the microcontroller will be influenced by the metal objects located near inductor L1 (figure 2). The microcontroller must measure its own frequency at start up, save it as reference frequency, and compare it with all other currently measured frequencies. To perform such a type of measurement, a RC circuit (R2,R3,C4) is connected to a GP2 pin and determines a constant time interval for calculations. GP2 has two functions: to discharge the capacitor C4 (as output) and to control its voltage (as input). See Figure 1.

The button S1 (RESET) is needed for periodical (from time to time) calibration of the reference frequency, since the LC oscillator frequency and the RC circuit time interval are a function of the supply voltage, the operating temperature and the stability of the components. The size of metal objects to be detected determine the size and geometry of inductor L1.

If we need stand alone small size metal detector, then outputs GP0 and GP1 can be connected to control LED and head phones (for sound effects).

If we need to get data from the PIC and to transmit some special commands to microcontroller, GP0 & GP1 can be I2C's - DATA and CLOCK pins.

FIGURE 1:
GRAPHICAL HARDWARE REPRESENTATION:

BILL OF MATERIALS (BOM)

<table>
<thead>
<tr>
<th>Part#</th>
<th>Manufacture</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1-PIC12C508</td>
<td>Microchip</td>
</tr>
<tr>
<td>C1-15pf</td>
<td>any</td>
</tr>
<tr>
<td>C2-15pf</td>
<td>any</td>
</tr>
<tr>
<td>C3-15pf</td>
<td>any</td>
</tr>
<tr>
<td>C4-470nf</td>
<td>any</td>
</tr>
<tr>
<td>R1-10K</td>
<td>any</td>
</tr>
<tr>
<td>R2-100K</td>
<td>any</td>
</tr>
<tr>
<td>R3-200</td>
<td>any</td>
</tr>
<tr>
<td>R4-1K</td>
<td>any</td>
</tr>
<tr>
<td>S1-button</td>
<td>any</td>
</tr>
<tr>
<td>D1-LED</td>
<td>any</td>
</tr>
<tr>
<td>L1</td>
<td>unknown</td>
</tr>
</tbody>
</table>
FLOW CHART

There are many intelligent algorithms that can be implemented in PIC12CXXX for metal detection and automatic calibration. The algorithm shown below is just for experiments.

START

Initial setup

Wait for power & oscillator stabilization

CALL CALC_FREQ
Measuring & calculating current oscillator frequency

Reference freq. = current freq.

CALL CALC_FREQ
Measuring & calculating current oscillator frequency

Freq. Offset = | Ref.freq. - curr.freq.|

Freq. Offset > max. limit ?

Y   N

LED = ON   LED = OFF

Y   N

RESET button pressed ?

CALC_FREQ

Reset capacitor of RC group (used for time interval)

Reset timer TMR0
Current freq. = 0

Current freq. = current freq. + TMR0 clocks

RC time interval over ?

Y        N

RETURN

MICROCHIP TOOLS USED

Assembler/Compiler version

MPLAB 3.22, MPASM 1.5
APPENDIX A: SOURCE CODE

;************************************************************************
; Using PIC12CXXX as a sensor interface for metal detection
; Written by Vladimir Velchev 07.1997.
; (C) AVEX - Vladimir Velchev
; Version 1.00
;************************************************************************

; LC osc.: F=2MHz; GP5/GP4 must be configured as XT type OSC1,2 in/outs
; GP0 - LED indicator (output: 0=LED ON, 1=LED OFF)
; GP1 - not used (reserved output for phones or DATA pin [GP0- CLOCK])
; GP2 - RC group - 100mS measurement time (input/output)
; GP3 - RESET button for calibration (input)
; GP4 - LC oscillator (OSC1 input)
; GP5 - LC oscillator (OSC2 output)

LIST P=12C508

#include <p12C508.inc>

;*** Equates
LED_Pin  equ 0  ;LED indicator - GP0
RC_Pin   equ 2  ;RC group
RESET_Pin equ 3  ;RESET button pin
FREQ_OFFS equ D'10'  ;freq. offset limit (threshold) |Fmax-Fmin|
                   ;determines the device sensibility
START_UP_COUNT equ D'10'  ;number of start up false measurements
IOSET equ B'00001000'  ;initial I/O port settings
                   ;GP3-input, others- outputs
RC_MASK equ B'00000100'  ;bit mask for RC pin

;*** RAM locations
Frh equ H'07'  ;reference frequency - MS byte
Frl equ H'08'  ;reference frequency - LS byte
Fch equ H'09'  ;current frequency - MS byte
Fcl equ H'0A'  ;current frequency - LS byte

;*** Vectors
org 0  ;RESET vector
BEGIN:
; Initial setup
movlw IOSET  ;init GPIO
tris GPIO
clrf GPIO  ;reset all outputs (=0)
movlw H'D2'  ;init option register
option
           ;TMR0: int.clock, prescaler 1:8

; Additional delay after start up
movlw START_UP_COUNT  ;read number of start up cycles
movwf Frl  ;use Frl as counter for start up
START_UP_LOOP:
call CALC_FREQ  ;call freq. subroutine
decfsz Fr1,l  ;counter Fr1--, skip if=0 (exit)
goto START_UP_LOOP

; Measurement of the reference frequency
call CALC_FREQ  ;call calculate freq. subroutine
movfr Fch,w  ;copy measured to reference freq.
movwf Frh
movf Fcl,w
movwf Frl

MAIN_LOOP:
call CALC_FREQ ;calculate current freq.

; Calculate absolute value of the frequency offset
; Fc= |Fc - Fb|
movf Frl,W ;read reference freq. LSbyte
subwf Fcl,1 ;sub. from current freq.
btfss STATUS,C ;skip if result is 0 or positive
decf Fch,1
movf Frh,W ;read reference freq. MSbyte
subwf Fch,1 ;sub. from current freq.
btfss Fch,7 ;skip if result is negative
goto CHECK_FREQ
comf Fcl,1 ;convert negative to positive offset
comf Fch,1 ;Fch:Fcl- absolute value of offset

CHECK_FREQ:
movf Fch,1 ;checks freq. offset MSbyte
btfss STATUS,Z ;skip if zero
goto LED_ON ;else - turn LED ON
movlw FREQ_OFFS ;read freq. offset limit
subwf Fcl,W ;compare with result offset
btfsc STATUS,C ;skip if result < limit offset
goto LED_ON ;else - turn LED ON

LED_OFF:
bsf GPIO,LED_Pin ;LED= OFF
goto CHECK_RESET ;go to check the reset button

LED_ON:
bcf GPIO,LED_Pin ;LED= ON

; Checking the RESET button (calibration)
CHECK_RESET:
btfsc GPIO,RESET_Pin ;skip if reset button is pressed
goto MAIN_LOOP ;go to measurement loop
goto BEGIN ;go to begin for calibration

;*** Subroutine - CALC_FREQ
; Input:
; Output: Fch:Fcl- current freq.
; Info: Calculates current frequency of the external oscillator

; Fosc.= 2MHz; Fclk.= 2MHz/4= 500kHz
; TMR0 prescaler: TMR0ps= 1:8
; TMR0freq= Fclk./TMROps= 500kHz/8= 62500Hz
; TMR0tick= 1/TMR0freq= 16 uS
; Measurement interval: (RC circuit) TRC = 100mS
; Frequency counter [max]: Fc= TRC/TMR0tick= 100mS/16uS= 6250
; Frequency counter rate: Frate= Fosc./6250= 320Hz
; |Fmax-Fmin| interval= FREQ_OFFS*Frate= 10*320Hz = 3200Hz
;
; Fch:Fcl = Fosc./320

CALC_FREQ:
clr Fch ;clear current freq. counters
clr Fcl

; Discharging the RC circuit
movlw IOSET&(~RC_MASK)
tris GPIO ;set RC pin as output
bcf GPIO,RC_Pin ;RC pin= 0
clr TMRO ;use TMRO as discharge timer

CALC_FREQ_DISCH:
clrwdt ;clear watchdog timer
movlw H'FF' ;look for TMRO overflow
subwf TMRO,W
btfss STATUS,Z ;skip if TMRO overflowed
goto CALC_FREQ_DISCH
; Enable RC time interval circuit
movlw IOSET|RC_MASK
tris GPIO ; set RC pin as input

; Start counting (measurement of the current frequency)
clf TMR0
CALC_FREQ_LOOP:
clrwdt ; clear watchdog timer
btfsc GPIO,RC_Pin ; continue if RC pin still=0
goto CALC_FREQ_STOP ; else- stop the measurement
movlw H'FF' ; look for TMR0 overflow
subwf TMR0,W
btfsc STATUS,Z ; skip if TMR0 not overflowed
incf Fch,1 ; increment MSbyte freq. counter
goto CALC_FREQ_LOOP
CALC_FREQ_STOP:
movf TMR0,W ; read current value of TMR0
movwf Fcl ; store to LSbyte of freq. counter
return
end ; end of program