Simple Alarm System

The alarm system discussed in this application note may be used to guard cars and homes. It is based on two PIC12C508s (one is used in the transmitter and the other one, in the main unit). The transmitter uses an infrared beam to send code names to the main unit. It has two commands: arm and disarm alarm. The code is fixed and is 64 bits long. When in disarm mode, it works like a central locking mechanism, and in arm mode, this feature is blocked.

Start and stop bits are separate and are 3T. The start bit is used to synchronize two RC generators in the main unit and in the receiver. Code word transmission format is shown in Figure 1. The main period is 400 μs, with 14 μs active pulse (except the first one pulse which is 20 μs). This format is ideal for infrared transmitters, since it saves battery power. With two 3V lithium batteries, the transmitter will work for more than a year with about six transmissions per day. Code word organization is shown in Figure 2. Only two bits of the first four bits are used. It took about 400 ms to send a transmission.

FIGURE 2: CODE WORD ORGANIZATION

<table>
<thead>
<tr>
<th>Fixed Code Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bits for buttons</td>
</tr>
<tr>
<td>60 bit password</td>
</tr>
</tbody>
</table>

FIGURE 1: CODE WORD TRANSMISSION FORMAT

- Period
- LOGIC ‘0’ 1T
- LOGIC ‘1’ 2T

Author: Kirill Yelizarov V.
Moscow Power Engineering Institute
Moscow, Russia
email: tihonov@srv-vmss.mpei.ac.ru
TRANSMITTER PARTS LIST

Capacitors: C1 – 470 µF (electrolytic)
Diodes: D1 – D2 Infrared light emitting diodes
Resistors: R1 – 1 Ω
R2 – 51 Ω
Miscellaneous: U1 – PIC12C508 programmed with transmitter code
U2 – 2N2222
S1 – S2 - normally open push-button switches

MAIN UNIT PARTS LIST

Capacitors: C1 – 47 µF (electrolytic)
C2 – .1 µF
C3 – 2200 µF (electrolytic)
Diodes: D1 – D10 Any type diodes
D11 – D14 high voltage diodes
D15 – 1 amp rectifier
LED – Red light emitting diode
Resistors: R1 – R8 (2 kΩ)
R9 – R11 (100 kΩ)
R12 – R16 (1 kΩ)
Miscellaneous: U1 – PIC12C508 programmed with alarm code
U2 – 78L05
U3 – U7 (2N2222)

The main unit is protected with a small 3V lithium battery. This is needed if thieves try to disconnect the car battery. Disconnecting both batteries and then reconnecting one, automatically arms the main unit. The main unit is shown in Figure 3.
Any amplifier with positive pulses can be connected to the infrared input (TBA2800 for example)

QUICK CODE IDEAS

This code has macro commands that are helpful in PICmicros with no interrupts.

MICROCHIP TOOLS USED

MPLAB™ for Windows/16 Version 3.30.00
Assembler/Compiler version: MPASM v02.00
SOURCE CODE

Car Alarm System – Main Unit

; Car Alarm System (Main Unit) Version 01 K 1997
; Author: Kirill Yelizarov

LIST P=PIC12C508, R=HEX
INCLUDE "p12c508.inc"
INCLUDE "alr97k01.pas" ;get password

__CONFIG _IntRC_OSC & _WDT_OFF & _CP_OFF & _MCLRE_OFF

ALARM_TIME equ 0x0a ;in intrusion mode make 10 signals

;----- Infrared input -----
#define IRinp GPIO,3 ;infrared input is connected to GP3 pin

;----- Alarm Main Functions Timing Table -----
;one tick (tk) equals to 32 ms

;----- LED -----
#define LED GPIO,4 ;LED is connected to GP4 pin
#define LED_d Flags,4 ;LED direction flag
LED_f equ 0x06 ;LED flashing time (6 tk = 192 ms)
LED_p equ 0x19 ;LED pause between flashes time (25 tk = 800 ms)

;----- Lights -----  
#define Lights GPIO,5 ;Lights are connected to GP5 pin
#define Lights_d Flags,5 ;Lights direction flag
Lights_f equ 0x0f ;lights flashing time (15 tk = 480 ms)
Lights_p equ 0x10 ;lights pause between flashes (16 tk = 512 ms)

;----- Beep ----- 
#define Beep GPIO,0 ;Horn is connected to GP0 pin
#define Beep_d Flags,0 ;Beep direction flag
Beep_f equ 0x0f ;beep activate time (15 tk = 480 ms)
Beep_p equ 0x10 ;beep sleep time (16 tk = 512 ms)

;----- Lock doors ----- 
#define DLock GPIO,2 ;Door lock relay and switch are connected to GP2 pin
#define DLock_d Flags,2 ;Lock direction flag
DLock_f equ 0x10 ;Lock activate time (16 tk = 512 ms)
DLock_p equ 0x01 ;should always be one

;----- Unlock doors ----- 
#define DUnlock GPIO,1 ;Door unlock relay and switch are connected to GP1 pin
#define DUnlock_d Flags,1;Unlock direction flag
DUnlock_f equ 0x10 ;Unlock activate time (16 tk = 512 ms)
DUnlock_p equ 0x01 ;should always be one

;----- Input switch ----- 
#define InSw GPIO,0 ;Intrusion switch is connected to GP0
#define InSw_f KeyFlags,0 ;Intrusion flag
#define Intrusion AlarmFlags,2

;----- Open switch ----- 
#define OpenSw GPIO,1 ;Door open switch is connected to GP1
#define OpenSw_f KeyFlags,1 ;Door open flag
#define Open SwitchFlags,1

;----- Close switch ----- 
#define CloseSw GPIO,2 ;Door close switch is connected to GP2
#define CloseSw_f KeyFlags,2 ;Door close flag
#define Close SwitchFlags,2
;----- AlarmFlags bits -----
NullFlag    equ     7
IRFlag      equ     6
Disarm      equ     5
Arm         equ     4
AlarmFlag    equ     3
TimerFlag   equ     1
BeepFlag    equ     0

;----- Local DATA -----
IRCount equ 0x07 ;counter used in IRdelay
IRCorrection equ 0x08 ;correction for infrared intervals
AlarmFlags equ 0x09 ;alarm flags
Flags equ 0x0a ;direction flags
LED_c equ 0x0b ;LED counter
Lights_c equ 0x0c ;Lights counter
Beep_c equ 0x0d ;Beep counter
DLock_c equ 0x0e ;Lock counter
DUnlock_c equ 0x0f ;Unlock counter
Dig1 equ 0x10 ;64 bit code
Dig2 equ 0x11
Dig3 equ 0x12
Dig4 equ 0x13
Dig5 equ 0x14
Dig6 equ 0x15
Dig7 equ 0x16
Dig8 equ 0x17
KeyFlags equ 0x18 ;key flags used by macro TestKey
AlarmCounter equ 0x19 ;alarm counter
AlarmTris equ 0x1a ;alarm TRIS register
SwitchFlags equ 0x1b ;switch flags used by central lock

;----- Macro -----
Check macro  r,rb,d,db,f,p,c
local out,pas
;r - working bit
d - direction flag
f - flashing time (in ticks) (d flag is set)
p - pause between flashes (in ticks) (d flag is cleared)
c - counter
delay 10us max
decfsz c,F ;decrease counter
goto out
btfss d,db ;check direction
goto pas
bcf r,rb ;clear working bit
bcf d,db ;clear direction bit
movlw p ;read pause between flashes
movwf c ;and save it to counter
goto out ;out from macro
pas:
bsf r,rb ;set working bit
bsf d,db ;set direction bit
movlw f ;read flashing time
movwf c ;and save it to counter
out:
endm

;This macro is used to search closed key
;button de bounce is 32 ms
;Delay 7us max
TestKey macro  r,rb,f,fb,a,ab
local out,reset,setact
;r - pin to test
; f - pin flag (if pin r is low then set flag f, and check the next time pin r and flag f)
;a - action bit this pinb should be checked by the program and cleared by it

btfsc r,rb ;if pin is high then out and reset flag
  goto reset
btfsc f,fb ;skip if flag is cleared
  goto setact
btfsc f,fb ;set flag and out
  goto out

setact:
  bsf a,ab ;set action
reset:
  bcf f,fb ;and clear flag
out:
  endm

;----- CODE -----

org 0
  goto Start

;-------------- SUBROUTINES --------------

Receive
btfsc IRinp ;wait till IRinp becomes low
  goto Receive
clrf Dig1 ;~22 us
clrf Dig2
clrf Dig3
clrf Dig4
clrf Dig5
clrf Dig6
clrf Dig7
clrf Dig8
nop
  movlw 0xb8
  call IRDelay ;delay 184*3-1+5=556 us
  movlw 0xb8
  call IRDelay ;delay 184*3-1+5=556 us

;this is a special correction routine
;this is needed to synchronize transmitter and receiver,
because they are clocked with internal RC generators

clrf IRCorrection
btfsc IRinp ;0
  goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;1
  goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;2
  goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;3
  goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;4
  goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;5
  goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;6
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;7
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;8
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;9
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;10
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;11
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;12
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;13
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;14
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;15
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;16
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;17
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;18
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;19
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;20
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;21
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;22
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;23
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;24
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;25
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;26
goto SetCorrection
incf IRCorrection,F
btfsc IRinp ;27
goto SetCorrection
incf IRCorrection, F
btfsc IRinp ;28
goto SetCorrection
incf IRCorrection, F
btfsc IRinp ;29
goto SetCorrection
incf IRCorrection, F
btfsc IRinp ;30
goto SetCorrection
incf IRCorrection, F
btfsc IRinp ;31
goto SetCorrection
incf IRCorrection, F
btfsc IRinp ;32
goto SetCorrection
incf IRCorrection, F
btfsc IRinp ;33
goto SetCorrection
incf IRCorrection, F
btfsc IRinp ;34
goto SetCorrection
incf IRCorrection, F
btfsc IRinp ;35
goto SetCorrection

goto BadRead

SetCorrection:
    movlw 0x7d ;~4 us after start
    call IRDelay ;delay 126*3-1+5=383 us
    nop
    bsf STATUS,C ;this STOP bit will end the loop

NextDigit:
    rrf Dig8, F
    rrf Dig7, F
    rrf Dig6, F
    rrf Dig5, F
    rrf Dig4, F
    rrf Dig3, F
    rrf Dig2, F
    rrf Dig1, F
    btfsc STATUS, C ;check for STOP bit
    goto Compare
    bsf AlarmFlags, NullFlag ;Set the null flag

RetryDigit:
    bcf AlarmFlags, IRFlag ;1 us Reset bit read flag
    nop
    nop
    nop
    btfsc IRinp ;5th us look for a bit
    bsf AlarmFlags, IRFlag ;Set if bit read
    movlw 0x23
    movwf IRCount

StartCorrection: ;Delay 244+IRCorrection us
    movf IRCount, W
    subwf IRCorrection, W ;for T=400 us delay 261 us
    btfsc STATUS, C
    goto AddCorrection
AddCorrection:
    decfsz IRCorrection, F
    goto StartCorrection
    nop
nop
nop
movlw 0x23
call IRDelay ;Delay 35*3-1+5=109 us
btfss AlarmFlags,IRFlag
goto ResetFlag
btfsc AlarmFlags,NullFlag ;If flag is clear it was "1"
goto Set0
nop
bsf STATUS,C ;Set an overflow flag to read "1"
goto NextDigit
Set0:
bcf STATUS,C ;Reset an overflow flag to read "0"
goto NextDigit
ResetFlag:
movlw 0x02
callIRDelay ;Delay 2*3-1+5=10 us
btfss AlarmFlags,NullFlag; Error if an overflow flag already reset
goto BadRead
bcf AlarmFlags,NullFlag
goto RetryDigit
Compare:
nop ;401th us
nop
nop
nop
btfsc IRinp ;405th us watch for clear bit
goto BadRead
movlw 0x23
movwf IRCount
StartCorrection1: ;Delay 244+IRCorrection us
movf IRCount,W ;for T=400 mks delay 261 us
subwf IRCorrection,W
btfsc STATUS,C
goto AddCorrection1
AddCorrection1:
decfsz IRCount,F
goto StartCorrection1
nop
;compare received data with password
movlw PASS8
xorwf Dig8,W
btfss STATUS,Z
goto BadCode
movlw PASS7
xorwf Dig7,W
btfss STATUS,Z
goto BadCode
movlw PASS6
xorwf Dig6,W
btfss STATUS,Z
goto BadCode
movlw PASS5
xorwf Dig5,W
btfss STATUS,Z
goto BadCode
movlw    PASS4
xorwf    Dig4,W
btfss    STATUS,Z
goto     BadCode

movlw    PASS3
xorwf    Dig3,W
btfss    STATUS,Z
goto     BadCode

movlw    PASS2
xorwf    Dig2,W
btfss    STATUS,Z
goto     BadCode

movf     Dig1,W
andlw    b'00001111'
xorwf    PASS1
btfss    STATUS,Z
goto     BadCode

movlw    0x20 ;
call     IRDelay ;delay 32*3-1+5=100 us
nop

btfsc    IRinp ;805 us
goto     BadRead

movlw    0x23
movwf    IRCount

StartCorrection2: ;Delay 244+IRCorrection us
movf     IRCount,W ;for T=400 us delay 261 us
subwf    IRCorrection,W
btfsc    STATUS,C
goto     AddCorrection2

AddCorrection2:
decfsz   IRCount,F
goto     StartCorrection2
nop

movlw    0x2b
call     IRDelay ;delay 43*3-1+5=133 us
nop

btfss    IRinp ;5th us look for a STOP bit
goto     BadRead

movlw    b'00110000'
andwf    Dig1,W
btfsc    STATUS,Z
goto     BadRead
iorwf    AlarmFlags,F ;Save function (ON/OFF)
retlw    0x00

BadCode:
bsf      Intrusion ;Set an intrusion flag for bad code

BadRead:
retlw    0x00

;Delay for timing intervals where actual delay is
;W*3-1+5 us with 4MHz oscillator
IRDelay
movwf    IRCount

DelayStart:
decfsz   IRCount,F
Wireless and Remote Controlled Personal Appliance

goto DelayStart
retlw 0x00

;  ------------ M A I N ------------

Start:
clr GPIO
movlw b'11000110' ;Dissable weak pull-ups and wake up on pin change
option ;and set prescaller to 1:128
movlw b'00001111' ;Set GP4 an GP5 as output and the rest are inputs
movwf AlarmTris ;save data to alarm tris register
tris GPIO

clr AlarmFlags ;clear alarm flags
clr Flags ;clear direction flags
clr SwitchFlags ;clear central lock switch flags
movlw 0x01
movwf Beep_c
movwf Lights_c
movwf LED_c
movwf DLock_c
movwf DUnlock_c

MainLoop:
btfsc IRinp ;Check IR input pin
    call Receive
btfsc TMR0,3 ;if timer0 highest bit is cleared
    goto Skip ;then check flag else goto Skip
btfsc AlarmFlags,TimerFlag
    goto Tick
Skip:
    bsf AlarmFlags,TimerFlag ;set TimerFlag
    btfss TMR0,3 ;if timer0 highest bit is cleared
    bcf AlarmFlags,TimerFlag ;then clear TimerFlag
    goto MainLoop

Tick:
    bcf AlarmFlags,TimerFlag ;clear TimerFlag
btfac IRinp ;Check IR input pin
    call Receive
btfac Open
    goto _Open
btfac Close
    goto _Close
btfss AlarmFlags,AlarmFlag
    goto _Armed ;if falg is cleared it is arm mode
    goto _Disarmed

_Armed:
    btfac Intrusion
    goto _Intrusion
btfac IRinp ;Check IR input pin
call Receive
Check LED,LED_d,LED_f,LED_p,LED_c;check LED (delay 10us)
btfac IRinp ;Check IR input pin
call Receive
TestKey InSw,InSw_f,Intrusion ;test intrusion switch (delay 7us)
movlw ALARM_TIME
movwf AlarmCounter ;set alarm counter
btfac AlarmFlags,Disarm
goto DisarmAlarm
goto MainLoop

_Intrusion:
    btfac AlarmFlags,Disarm
goto DisarmAlarm
    bcf AlarmTris,0 ;set GP0 to be output
movf AlarmTris, W
tris GPIO
bsf LED ;turn on LED
btfsr IRinp ;Check IR input pin
call Receive
Check Lights, Lights_d, Lights_f, Lights_p, Lights_c ;check Lights (delay 10us)
btfsr IRinp ;Check IR input pin
call Receive
Check Beep, Beep_d, Beep_f, Beep_p, Beep_c ;check Beep (delay 10us)
decfsz AlarmCounter, F
goto MainLoop
bcf Intrusion
bsf AlarmTris, 0 ;set GP0 to be input
movf AlarmTris, W
tris GPIO
bcf LED ;turn off LED
goto MainLoop

_disarmed:
bcf LED ;turn LED off
btfsr IRinp ;Check IR input pin
call Receive
TestKey OpenSw, OpenSw_f, Open ;test open switch (delay 7us)
btfsr IRinp ;Check IR input pin
call Receive
TestKey CloseSw, CloseSw_f, Close ;test close switch (delay 7us)
bfsr AlarmFlags, Arm
goto ArmAlarm
goto MainLoop

_Open:
bcf AlarmTris, 1 ;set GP1 to be output
movf AlarmTris, W
tris GPIO
btfsr IRinp ;Check IR input pin
call Receive
Check DUnlock, DUnlock_d, DUnlock_f, DUnlock_p, DUnlock_c; check Unlock (delay 10us)
btsr DUnlock_d ;wait till direction flag changes its state
goto MainLoop
bsf AlarmTris, 1 ;set GP1 to be input
bcf Open
movf AlarmTris, W
tris GPIO
goto MainLoop

_Close:
bcf AlarmTris, 2 ;set GP2 to be output
movf AlarmTris, W
tris GPIO
btfsr IRinp ;Check IR input pin
call Receive
Check DLock, DLock_d, DLock_f, DLock_p, DLock_c; check Lock (delay 10us)
btsr DLock ;wait till direction flag changes its state
goto MainLoop
bsf AlarmTris, 2 ;set GP2 to be input
bcf Close
movf AlarmTris, W
tris GPIO
goto MainLoop

ArmAlarm:
bcf AlarmFlags, Arm
bsf Close
goto MainLoop

DisarmAlarm:
bcf AlarmFlags, Disarm
bsf Open
goto MainLoop

org 0x1ff
movlw b'01110000'
;set OSCCAL

end
Car Alarm System - Transmitter

;Car Alarm System (Transmitter) v1.0 1997
;Author: Kirill Yelizarov

LIST P=PIC12C508, R=HEX
INCLUDE pi2c508.inc
INCLUDE air97k01.pas ;get password

__CONFIG _ExtRC_Osc & _WDT_Off & _CP_Off & _MCLRE_Off

----- GPIO Port bits -----
IRLED equ 2 ;Infrared LED pin

----- Local DATA SFRs -----
Count equ 0x07
DelayCount equ 0x08
DelayCountLow equ 0x09
DelayCountHi equ 0x0a

----- Password SFRs -----
Dig1 equ 0x10
Dig2 equ 0x11
Dig3 equ 0x12
Dig4 equ 0x13
Dig5 equ 0x14
Dig6 equ 0x15
Dig7 equ 0x16
Dig8 equ 0x17

----- CODE -----
org 0
btfsc STATUS,GPWUF
    goto Transmit
clrf GPIO
movlw b'00000000' ;Enable weak pull-up on GP0 and GP1
    movlw b'00111011' ;Set GP2 as output and the rest are inputs
    tris GPIO
    goto _Sleep

Delay for timing intervals where actual delay is
    W*3-1+5 us with 4MHz oscillator
Delay movwf DelayCount
DelayStart:
    decfsz DelayCount,F
    goto DelayStart
    retlw 0x00

;Delay for key de bounce where actual delay is
;W ms with 4MHz oscillator
LongDelay movwf DelayCountHi
    clrf DelayCountLow
DelayLoop:
    nop
    incfsz DelayCountLow,F
    goto DelayLoop
    decfsz DelayCountHi,F
    goto DelayLoop
    retlw 0x00

Transmit:
    comf GPIO,W ;read and invert GPIO
andlw b'00000011'
btfsc STATUS, Z ;check if a button is pressed
goto _Sleep ;if not go to sleep
movlw 0x1e
call LongDelay ;wait for 30 ms
comf GPIO, W ;read and invert GPIO again
andlw b'00000011'
btfsc STATUS, Z ;check if a button is still pressed
goto _Sleep ;if not go to sleep
xorlw PASS1 ;xor pressed buttons with PASS1 digit
movwf Dig1 ;assign password
swapf Dig1, W
movlw PASS2
movwf Dig2
movlw PASS3
movwf Dig3
movlw PASS4
movwf Dig4
movlw PASS5
movwf Dig5
movlw PASS6
movwf Dig6
movlw PASS7
movwf Dig7
movlw PASS8
movwf Dig8
movlw 0x40
movwf Count ;64 bits transmission†
bsf GPIO, IRLED ;Send START bit
nop ;each bit is a 14 us pulse
nop ;but the Start one is 20 us
movlw 0x04
call Delay ;delay 4*3-1+5=16 us
bcf GPIO, IRLED
movlw 0xc3
call Delay ;delay195*3-1+5=589 us
nop
nop
movlw 0xc2
call Delay ;delay194*3-1+5=586 us

NextDig:
bsf GPIO, IRLED ;Send another bit
bcf STATUS, C
rrf Dig8, F
rrf Dig7, F
rrf Dig6, F
rrf Dig5, F
rrf Dig4, F
rrf Dig3, F
rrf Dig2, F
rrf Dig1, F ;rotate 64 bits to get the next bit
nop ;to transmit and place it into STATUS, C
nop
nop
nop
bcf GPIO, IRLED
movlw 0x0a
call Delay ;delay 10*3-1+5=34 us
btfss STATUS, C ;if STATUS, C is low
goto Send0 ;then delay between bits is 400 us
movlw 0x84 ;else add another 400 us
call Delay ;delay 132*3-1+5=400 us

Send0:
movlw 0x71
call Delay ;delay 113*3-1+5=343 us
decfsz Count,F ;last bit is not reached
goto NextDig
nop
bsf GPIO,IRLED ;send END bit
nop
nop
movlw 0x02
call Delay ;delay 2*3-1+5=10 us
bcf GPIO,IRLED
movlw 0xc4
call Delay ;delay196*3-1+5=592 us
nop
nop
movlw 0xc3
call Delay ;delay195*3-1+5=589 us
bsf GPIO,IRLED ;send STOP bit
nop
nop
movlw 0x02
call Delay ;delay 2*3-1+5=10 us
bcf GPIO,IRLED
.Sleep:
sleep ;go to sleep
org 0x1ff
movlw b'01110000' ;set OSCCAL
end
Alarm Pass

PASS1 equ 0x01
PASS2 equ 0x23
PASS3 equ 0x45
PASS4 equ 0x67
PASS5 equ 0x89
PASS6 equ 0xab
PASS7 equ 0xcd
PASS8 equ 0xef