Laboratorij za CAD - LECAD
Fakulteta za strojništvo
Univerza v Ljubljani

Identifikacija s kriptografskimi gumbi

Interno poročilo

Avtorja:

Leon Kos, Jože Duhovnik

Ljubljana, junij 1997

Povzetek

Označevanje in identifikacija v okoljih, kjer je potrebno skrbeti za čistočo ali v okoljih z zunanjimi vplivi, je potrebno uporabiti sistem identifikacije, ki je odporen na vplive vlage in umazanije. Prikazan je sistem s kriptografskimi "iButton" gumbi narejeni iz nerjavne pločevine premera 16mm v katerega so vgrajeni Ni-Cd baterija, mikroprocesor in spomin. Zaradi robustnosti je uporaben za vsakodnevno uporabo. Na praktičnem primeru identifikacije je bil narejen sistem kontrole prehoda v laboratoriju za računalniško podprto konstruiranje. Sistem se je pokazal kot uspešen in primeren za uporabo v večjih organizacijah, kot tudi za druge namene.

1 Uvod

Sistem prehoda skozi vrata mora biti zanesljiv, cenen, robusten, hiter in enostaven za uporabo. Za izvedbo so možne naslednje metode kontrole prehoda:

ISO kartica Ima vgrajen procesor in logiko. Uporabno je za večje količine in manjšo frekventnost. Stopnja zaščite je lahko zelo velika. Ker se kartice delajo po naročilu (vsaj nekaj tisoč), to ni uporabno za manjše število kartic.

Magnetna kartica Nosi podatke zapisane na kartici. Predvsem številko. Robustnost je vprašljiva, saj se lahko zaradi frekventnosti magnetna folija poškoduje. Prednost je cenenost. Uporabno pa za manjše frekvence uporabe.

Indukcijska kartica Odpravlja težave magnetne kartice, saj višja frekvenca ne vpliva na obrabo in morebitno poškodbo. So dražje od magnetnih (cca 15 DEM).Imajo vpisano identifikacijsko številko, ki jo detektor spozna in obdela. Robustnost je visoka, vendar je težava podobna kot pri magnetnih karticah njihova velikost. Za detekcijo kartice na razdalji 0.5m je kartica velika kot magnetna. Stopnja zaščite je na nivoju magnetnih kartic. Če pa želimo višjo stopnjo zaščite se lahko vgradi dodatna kriptografska logika, ki pa bistveno podraži kartico in je tako neuporabna pri manjših količinah.

Tipkanjem kode Enostavno, vendar počasno. Možnost odkrivanja s opazovanjem. Obraba tipkovnice. Cenena izvedba sistema. Težava je tudi v tem, da je za večjo stopnjo zaščite potrebno daljše geslo, kar zmanšuje uporabnost. Večje šifre se težko zapomnimo.

Kriptografski gumb Proizvod firme Dallas Semiconductors. Gummb velikosti 16mm iz nerjavnega jekla. Gumb ima dva kontakta za serijsko komunikacijo z zunanjim svetom. Vgrajeno ima litijevo baterijo za hranjenje uporabniških informacij in pogon ure. Za komunikacijo in kodiranje ima vgrajen mikroprocesor. Zelo robustna in cenena varianta ($5). Vsak gumb je

v tovarni narejen s serijsko številko, ki je različna. Tako je že vsak gumb različen in je možna identifikacija na nivoju magnetne ali indukcijske kartice. Poleg tega pa je možno uporabiti še kriptografski procesor in uro v njem.

Za izvedbo identifikacije je po robustnosti in cenenosti kot tudi po visoki stopnji zaščite najbolj primeren *iButton* kriptografski gumb. Obstaja več tipov gumbov, ki s ločijo po vgrajenih funkcijah. Skupna jim je oblika in način komunikacije. Obstajajo variane z različno količino spomina, z ali brez ure, s kriptanjem in celo z vgrajenim termometrom. Tako je možno uporabiti te gumbe v različne namene. Najenostanvejša uporaba je identifikacija, saj so vsi gumbi z unikatno številko. Uporaba spomina daje nešteto možnosti za aplikacije.

Slika 1: Kriptografski gumb v obliki prstana

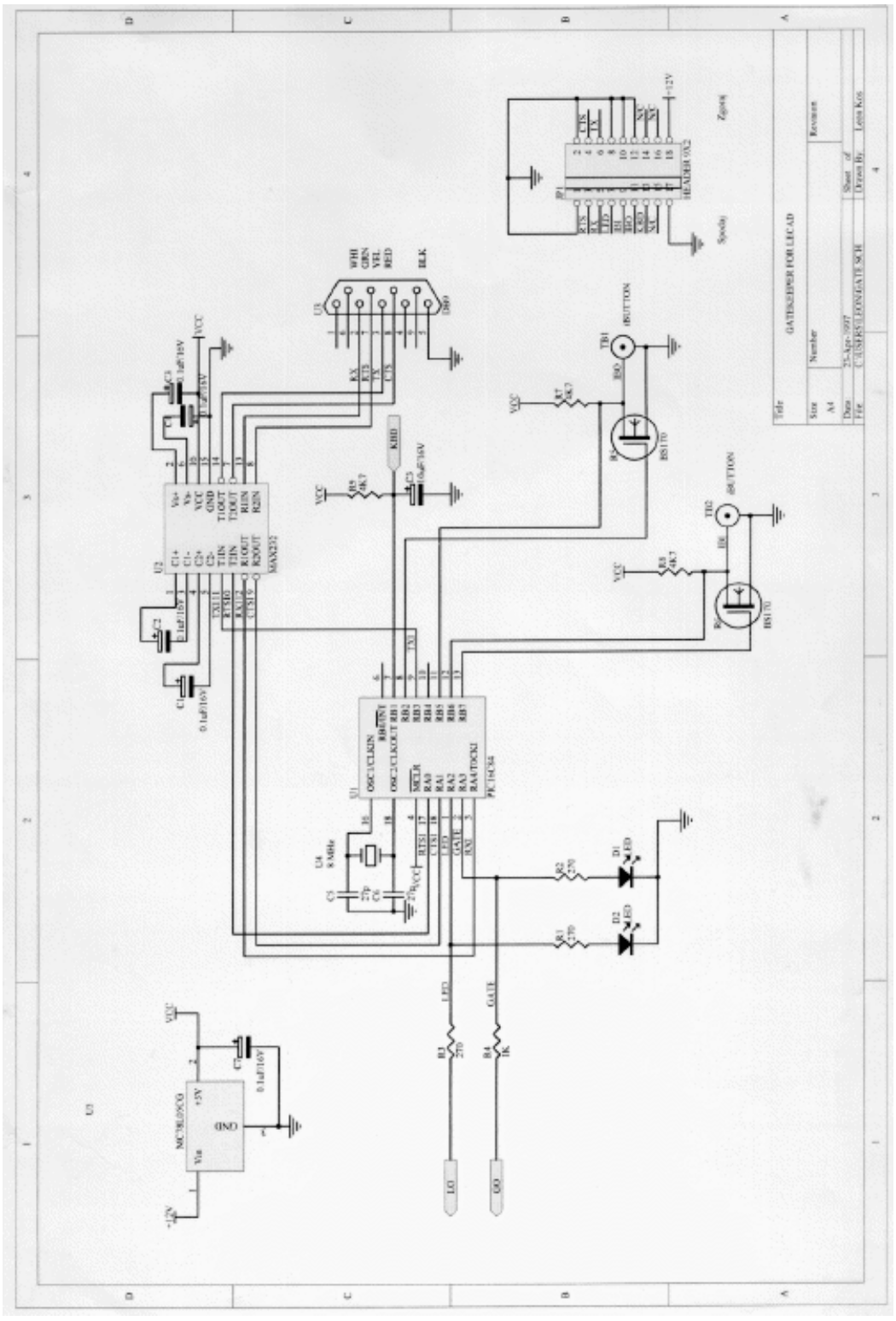
2 Vezje za branje gumbov in komunikacijo z računalnikom

Za komunikacijo z gumbom, po posebnem serijskem protokolu *one wire protocol*, se lahko uporabi tudi serijski vmesnik osebnega računalnika. Pri tem je potrebno nivoje protokola RS-232 prirediti na 5V nivoje protokola za komunikacijo z gumbom.

Ker je nadzor in vstona kontrola predvidena na unix delovni postaji, je bilo potrebno izdelati vezje, ki komunicira z postajo in bere gumbe. Za komunikacijo je bil izbran RS-232 protokol hitrost 4800 baudov. Vezje, ki pa komunicira z gumbi je moralo biti dovolj majhno, da se ga lahko vgraditi v kovinski profil vrat.

Mikrokontroler PIC 16C86 je bil izbran kot najprimernejši mikrokrmilnik za komunikacijski vmesnik. To je računalnik v enem čipu, ki ima RISC arhitekturo, vgrajen watch dog timer in je dovolj hiter tudi za programsko izvedbo protokola RS-232.

Slika 2 prikazuje shemo vezja, ki je bilo izvedeno velikosti TIC-TAC ohišja (1x5x3cm). Za vstop je bil izveden tudi paralelni sistem dveh tipkovnic ULSI-4,



Slika 2: Shema vezja za komunikacijo in branje gumbov in zunanje tipkovnice

ki delujeta tudi ob izpadu elektrike. Tako se je povečala funkcionalnost celotnega sistema in s tem tudi zanesljivost.

2.1 Komunikacija z gumbi

Program za komunikacijo z gumbi je pisan v asemblerju mikroprocesorja. Za razumevanje delovanja je potrebno razumevanje protokola *one-wire protocol*, asemblerja in sheme na sliki 2.

```
        subtitle "DALLAS iButton(TM) reader by Leon Kos"


; Outside connection
#define DATA_BIT1_IN  _portb, 5         ; Button 1 in
#define DATA_BIT1_OUT _portb, 2         ; Button 1 out (gate of NMOS)
#define DATA_BIT2_IN  _portb, 6         ; Button 2 in
#define DATA_BIT2_OUT _portb, 1         ; Button 2 out (gate of NMOS)



TouchInit
    bsf     _rp0                    ; Select Page 1 for TrisB access
    bcf     DATA_BIT1_OUT           ; set Pin As Output Pin, by modifying TR
    bcf     DATA_BIT2_OUT           ; set Pin As Output Pin, by modifying TR
    bsf     DATA_BIT1_IN            ; set Pin As Input Pin, by modifying TRI
    bsf     DATA_BIT2_IN            ; set Pin As Input Pin, by modifying TRI
    bcf     _rp0                    ; Select Page 0
        bcf     DATA_BIT1_OUT           ; Release
        bcf     DATA_BIT2_OUT           ; both lines
        clrwdt                          ; Clear WatchDog Timer
        call    ResetButtonDelay
        return


TouchReset      macro data_bit, _in, data_bit2, _out
        local   _tr1, _tr2, waitlow, _wl1, wh, hl, _hl, SHORT


    bcf     _rp0                    ; Select Page 0
        clrwdt                          ; Clear WatchDog Timer
    bsf     data_bit2,_out          ; Start the Reset pulse

        movlw   0x02                    ; Zakasnitev 500us
```

```
        movwf   TEMP
_tr2    movlw   0xfa
        movwf   tmp1
_tr1    decfsz  tmp1, 1
        goto    _tr1
        decfsz  TEMP, 1
        goto    _tr2

        bcf     data_bit2,_out
        movlw   19
        movwf   TEMP
        bcf     _carry                  ; Clear presence flag
waitlow
        movlw   160
        movwf   tmp1
_wl1    btfsc   data_bit,_in            ; Exit loop if line high
        goto    wh                      ; This may be presence pulse
        decfsz  tmp1, 1                 ;180 us inner loop
        goto    _wl1                    ; hang arround for about
        decfsz  TEMP, 1                 ; 3430 us if line is low
        goto    waitlow
        goto   SHORT                    ; line could not go high


wh
        movlw   200                    ; Delay for presence detect
        movwf   TEMP
hl      btfss   data_bit,_in
        bsf     _carry                  ; set carry if presence

_hl     decfsz  TEMP, 1
        goto    hl

        return
SHORT
        movlw   0xf0
        movwf   CRC                     ; Pokvari CRC
        bcf     _carry                  ; what to do on short?
        return

        endm

TouchReset1   TouchReset DATA_BIT1_IN, DATA_BIT1_OUT
```

```
TouchReset2   TouchReset DATA_BIT2_IN, DATA_BIT2_OUT


TouchByte       macro   data_bit,_in, data_bit2,_out
        local bit_loop, touchbit, _tb1
    bcf     _rp0                    ; Select Page 0
        movwf   TEMP                    ; Store byte
        movlw   8                       ; Setup for 8 bits
        movwf   tmp1                    ; loop counter
bit_loop
        rrf     TEMP, 1                 ; 1. Get bit in carry
        call    touchbit                ; 2. Send bit
        decfsz  tmp1, 1                 ; 1. Get next bit
        goto    bit_loop
        rrf     TEMP, 0                 ; get final bit in W
        return
touchbit
        clrwdt                          ; Clear the WatchDog Timer
        bsf     data_bit2,_out         ; 1 Start the time slot
        nop                             ; 1  Delay to make shure
        nop                             ; 1   that
        nop                             ; 1     the Touch Memory
        nop                             ; 1      sees a low
        nop                             ; 1       for at least
        nop                             ; 1        1 microsecond.
        nop                             ; 1.
        nop                             ; 1.
        btfss _carry                    ; 1 Send out
        bsf data_bit2,_out              ; 1 the
        btfsc _carry                    ; 1. data
        bcf data_bit2,_out              ; 1. bit.
        nop                             ; 1. Delay
        nop                             ; 1.  to give
        nop                             ; 1.   the Touch Memory
        nop                             ; 1.    time to settle
        nop                             ; 1.     before reading
        nop                             ; 1.      the bit
        nop
        nop
        nop
        nop
        nop                            ; 1.
        nop
```

```
        btfsc data_bit,_in              ; 1. Sample input
        bsf     _carry                  ; 1. data
        btfss data_bit,_in
        bcf     _carry
        movlw   0x2A                    ; 1. 0x24
        movwf   tmp2                    ; 1.
_tb1   decfsz  tmp2, 1                  ; 1.
        goto    _tb1                    ; 72. Delay until end
        bcf   data_bit2,_out            ;  Terminate time slot.
        return                          ; Return to the caller.
        endm

; IMPLEMENTATION
TouchByte1      TouchByte       DATA_BIT1_IN, DATA_BIT1_OUT
TouchByte2      TouchByte       DATA_BIT2_IN, DATA_BIT2_OUT


DoCRC   ; Izracuna DOW CRC vsoto. Ohrani W.
        movwf   crca
        movlw   8
        movwf   crcb
        movf    crca, W
crc_loop
        xorwf   CRC, W          ; Calculate CRC
        movwf   crcl
        rrf     crcl, 1         ; move it to the carry
        movf    CRC, W          ; Get the last CRC
        btfsc   _carry          ; Skip if data=0
        xorlw   0x18
        movwf   CRC             ; store the new CRC
        rrf     CRC, 1
        movf    CRC, W          ; za izpis
        bcf     _carry
        rrf     crca, W         ; Get the remaining bits
        btfsc   _carry
        iorlw   0x80
        movwf   crca
        decfsz  crcb, 1         ; Repeat for eight bits
        goto    crc_loop
        movf    crca, W
        clrwdt                   ; Clear the WatchDog Timer
        return
```

```
; Prebere ROM, ga shrani in izracuna CRC
; eliminirati mora tudi button 00 00 00 00 00 00, ki ima CRC=0
ReadRom macro   TouchReset, TouchByte
        local _sernum
        clrf    CRC             ; Brisi CRC
        call    TouchReset
        movlw   8               ; 8 znakov serijske kode
        movwf   BREG
        movlw    0x33
        call    TouchByte

        movlw   IBUTTONROM
        movwf   _fsr

_sernum movlw   0xFF
        call    TouchByte
        call    DoCRC
        movwf    _indf
        incf    _fsr, 1
        decfsz BREG, 1
        goto    _sernum
        movf    IBUTTONROM, 1
        btfss   _z
        return                  ; Vse OK vrni se
        movlw   0xFF
        movwf   CRC             ; pokvari CRC, ker device ni pravi. JE 00
        return

        endm

; IMPLEMENTATION
ReadRom1        ReadRom TouchReset1, TouchByte1
ReadRom2        ReadRom TouchReset2, TouchByte2


PrintRom   ; izpise ROM kodo na UART
        movlw   6               ; 8 znakov serijske kode
        movwf   BREG
        movlw   IBUTTONROM+7    ; Izpisujemo rikverc
        movwf   _fsr
_sprint decf    _fsr, 1
        movf    _indf, W
```

```
        call    PrintHex
        decfsz BREG, 1
        goto    _sprint
        return


CheckButtons;   Preveri ali je kak button pritisnjen
                ; vrne 1 ce je bil odprt sicer 0
        clrwdt                   ; Clear the WatchDog Timer
        movf    BDLY, W
        iorwf   BDLY+1, W
        iorwf   BDLY+2, W
        btfsc   _z
        goto    test_button1

        decfsz  BDLY, 1
        retlw   0
        decfsz  BDLY+1, 1
        retlw   0
        decfsz  BDLY+2, 1
        retlw   0

test_button1
        clrwdt                  ; Clear the WatchDog Timer
        call    TouchReset1     ; button 1
        btfss   _carry
        goto    test_button2
        call    ReadRom1
        movf    CRC, 1
        btfss   _z
        goto    test_button2    ; CRC error
        movlw   'i'
        call    uarttx
        movlw   'b'
        call    uarttx
        call    PrintRom
        call    printcrlf
        call    SetButtonDelay
        retlw   1
test_button2
        clrwdt                  ; Clear the WatchDog Timer
        call    TouchReset2     ; button 2
        btfss   _carry
```

```
        retlw   0
        call    ReadRom2
        movf    CRC, 1
        btfss   _z
        retlw   0
        movlw   'o'
        call    uarttx
        movlw   'b'
        call    uarttx
        call    PrintRom
        call    printcrlf
        call    SetButtonDelay
        retlw   1


SetButtonDelay
        clrf    BDLY
        clrf    BDLY+1
        movlw   0x10            ; zakasnitev med branji buttonov
        movwf   BDLY+2
        return


ResetButtonDelay:
        clrf    BDLY
        clrf    BDLY+1
        clrf    BDLY+2
        return
```

2.2 Glavni program

Poleg komunikacije s hardverskim rokovanjem (*handshaking*), hitrosti 4800 budov, je bilo izvedeno tudi branje signalov iz paralelnega sistema numerične tipkovnice preko ožičenega ali vezja. Dolžina signala v milisekundah pomeni, katera koda je bila odtipkana.

Ker v procesorju poganjamo partralelno več programo je bilo izvedeno multitasking procesiranje na dva načina:

interrupt S prekinitvami komuniciramo z zunanjim računalnikom in mu pošiljamo RS232 signale.

pooling Branje gumbov znotraj in zunaj vrat ter zunanji tuipkovnici beremo z zaporednim poganjanjem ustreznih podprogramov.

Naslednja koda predstavlja srce programa:

```
    TITLE           "GATEKEEPER iButton(TM) lecadgated RS232 Communications
    SUBTITLE        "Main Program"

;*************************************************************************
; Program za komunikacijo s postajo (aphro.fs.uni-lj.si), ki vodi nadzor na
; Dallas iButton kljucev na notranji in zunaji strani vrat. Racunalnika kom
; baudov, 8 bit, no parity, 1 stop bit. Handshaking mora biti RTC-CTS kot j
; komunikacijo (4 + 1 zica). iButtona se bereta le po serijski stevilki. Ho
; skrbi za odpiranje vrat.
;
; Serijski prenos je v interrupt nacinu. iButtona sta v pool nacinu.
; Zunanja tipkovnica je v pool nacinu.
;
;
; April 1997
;       Leon Kos
;
;TODO:
; 1. Zunanji keyboard detect
; 2. EEPROM athentication
; 3. Serial EEPROM programming
; 4. Tamper detect
; 5. WatchDog timer                         ; OK 18.4.1997
;*************************************************************************

        Processor       16C84
                __config 0x3FFE

        Radix   DEC
        EXPAND

        include         "16Cxx.h"

;*************************************************************************
;                               Setup RS-232 Parameters
;*************************************************************************

_ClkIn          equ     8000000         ; Input Clock Frequency is 8 Mhz
_BaudRate       set     4800            ; Baud Rate (bits per second) is 12
_DataBits       set     8               ; 8 bit data, can be 1 to 8
_StopBits       set     1               ; 1 Stop Bit, 2 Stop Bits is not im
```

```
#define _PARITY_ENABLE  FALSE           ; NO Parity
#define _ODD_PARITY     FALSE           ; EVEN Parity, if Parity enabled
#define _USE_RTSCTS     TRUE            ; NO Hardware Handshaking is Used

#define GATE    _porta, 3           ; GATE Pin, RA3, Output signal
#define LED     _porta, 2               ; LED diode for status
#define KEYBOARD _portb, 4              ; Impulzi zunanje tipkovnice

        include         "rs232.h"

;*****************************************************************************
;

    ORG     _ResetVector
    goto    Start


    ORG     _IntVector
    goto    Interrupt
;




;
;*****************************************************************************
;                                       Main Program Loop
;
;  After appropriate initilization, The main program wait for a command fro
;  The command is 0, 1, 2 or 3. This command/data represents the A/D Channe
;  After a command is received, the appropriate A/D Channel is seleted and
;  completed the A/D Data is transmitted back to the Host. The controller n
;  command.
;*****************************************************************************

stringtable
    addwf   _pcl,1              ; Add W value to PC for look-up table
stringbase
logo    DT      "LECAD - Gatekeeper READY\r\n\0"
retstring
    DT      "Pritisnil si naslednjo tipko: \0"
presence
```

```
        DT      "iButton present\r\n\0"

  cblock 0x16
        TEMP    ; Zacasni pomnilnik
        CHARBUF ; pomnilnik za znake
        tmp1, tmp2 ; zacasni spremenljivki za zanke
        BREG            ; General register
        CRC, crca, crcb, crcl   ; Cyclic Redundancy Check of the ROM
        IBUTTONROM:8    ; iButton ROM buffer
        A               ; A register
        BDLY:3          ; Button Check Delay
        KBTIMER:4       ; Keyboard timer
        endc

uarttx:
    movwf   TxReg
   if _USE_RTSCTS
    bsf     _RTS                    ; Half duplex mode, transmission mode, a
    btfsc   _CTS                    ; Check CTS signal if host ready to acc
    goto    $-1
   endif
    call    PutChar
    btfsc   _txmtProgress
    goto    $-1                     ; Loop Until Transmission Over, User Ca
        return

; ********************************************************************
; * printstring - print out a string of chars                        *
; ********************************************************************
printstring
    movwf   TEMP            ; Place string offset into temp
loopprint
    movf    TEMP,W          ; Place next char to be sent into W
    call    stringtable     ; Look up the next char to send
    movwf   CHARBUF         ; Place char into CHARBUF for temp storage
    xorlw   '\0'            ; Place end of string char into W
    btfsc   _z              ; Skip if not at end of string
    retlw   0               ; At end of string - done so go back!
    incf    TEMP,F          ; Point to next character
    movf    CHARBUF,W       ; Place print char into W
    call    uarttx          ; Send char to the screen
        goto    loopprint       ; Loop back for the next char
```

```
; ******************************************************************
; * printcrlf routine - send carrage return and line feed          *
; ******************************************************************
printcrlf
    movlw   .13             ; Place value for carrage return into W
    call    uarttx          ; Send it to the RS-232 port
    movlw   .10             ; Place value for the Line Feed into W
    call    uarttx          ; Send it to the RS-232 port
    retlw   0               ; Done, so return!


TestChar        macro  character
        local char_loop, char_received, _tstendr
   if _USE_RTSCTS
    bcf     _rp0
    bcf     _RTS                    ; ready to accept data from host
   endif
    call    GetChar         ; Get a char from the terminal
char_loop
   if _USE_RTSCTS
        btfsc   _rcvProgress
    bsf     _RTS            ; Deny host to send data
   endif

        call    CheckKeyboard
        xorlw   0x02            ; ali je tiskal button
        btfsc   _z
        goto    getnextchar     ; da resetiraj


        call    CheckButtons
        xorlw   0x01            ; ali je tiskal button
        btfsc   _z
        goto    getnextchar     ; da resetiraj

_tstendr
        btfsc   _rcvOver
        goto    char_loop
char_received
   if _USE_RTSCTS
    bcf     _rp0
```

```
    bsf     _RTS            ; Deny host to send data
   endif

        movf    RxReg, W
    xorlw   character       ; Place end of string char into W
    btfss   _z              ; Skip if character is OK
        goto    getnextchar
        endm


Start:
    bsf     _rp0                    ; Select Page 1 for TrisB access
    bcf     GATE                    ; set GATE Pin As Output Pin, by modify
        bcf     LED
    bcf     _rp0                    ; select Page 0 for Port Access
    bcf     GATE                    ; make sure GATE
        bcf     LED
    call    InitSerialPort
        call    SetupKBtimer

    bcf     _rp0                    ; make sure to select Page 0
    movlw   logo-stringbase         ; Place offset address of string into W
        call    printstring

        call    TouchInit               ; Initialize Touch Button ports
        ; call    TouchReset1
        ; btfss   _carry
        ; goto    getnextchar
    ; movlw   presence-stringbase ; Place offset address of string into W
        ; call    printstring

;*****************************************************************************
; MAIN LOOP
;*****************************************************************************
getnextchar
        call AbortGetChar
        clrwdt                  ; CXlear WatchDog Timer
        TestChar 'o'

   if _USE_RTSCTS
    bsf     _RTS                    ; Ask Host not to send data
   endif
        bsf     GATE            ; Open gate
```

```
        bsf     LED             ; Set indicator light
        movlw   'O'
        call    uarttx
        movlw   'K'
        call    uarttx
        call    printcrlf
        movlw 100               ; Gate open delay (10 seconds)
        call Delay01
        bcf  GATE               ; Close the gate
        bcf  LED                ; Dim gate indicator
        call    ResetButtonDelay
    goto    getnextchar     ; Jump back and do it again!

;
;*****************************************************************
;                               RS-232 Routines
;*****************************************************************
;                               Interrupt Service Routine
;
; Only RTCC Inturrupt Is used. RTCC Inturrupt is used as timing for Serial
; Since RS-232 is implemented only as a Half Duplex System, The RTCC is sha
; Transmit Modules.
;       Transmission :
;                       RTCC is setup for Internal Clock increments and int
;                       RTCC overflows. Prescaler is assigned, depending on
;                       desired BAUD RATE.
;       Reception :
;                       When put in receive mode, RTCC is setup for externa
;                       and preloaded with 0xFF. When a Falling Edge is det
;                       rolls over and an Interrupt is generated (thus Star
;                       bit is detected, RTCC is changed to INTERNAL CLOCK
;                       with a certain value for regular timing interrupts
;
;*****************************************************************

Interrupt:
    btfss   _rtif
    retfie                          ; other interrupt, simply return & enab
;
; Save Status On INT : WREG & STATUS Regs
;
    movwf   SaveWReg
    swapf   _status,w               ; affects no STATUS bits : Only way OUT
```

```
    movwf   SaveStatus
;
    btfsc   _txmtProgress
    goto    _TxmtNextBit            ; Txmt Next Bit
    btfsc   _rcvProgress
    goto    _RcvNextBit             ; Receive Next Bit
    goto    _SBitDetected           ; Must be start Bit
;
RestoreIntStatus:
    swapf   SaveStatus,w
    movwf   _status                 ; restore STATUS Reg
    swapf   SaveWReg, F             ; save WREG
    swapf   SaveWReg,w              ; restore WREG
    bcf     _rtif
    retfie
;
;*****************************************************************************
;
;
;
; Configure TX Pin as output, make sure TX Pin Comes up in high state on Res
; Configure, RX_Pin (RTCC pin) as Input, which is used to poll data on recep
;
;  Program Memory :     9 locations
;  Cycles         :     10
;*****************************************************************************

InitSerialPort:
    clrf    SerialStatus
;
    bcf     _rp0                            ; select Page 0 for Port Access
    bsf     TX                              ; make sure TX Pin is high on po
    bsf     _rp0                            ; Select Page 1 for TrisB access
    bcf     TX                              ; set TX Pin As Output Pin, by m
  if _USE_RTSCTS
    bcf     _RTS                            ; RTS is output signal, controll
    bsf     _CTS                            ; CTS is Input signal, controlle
  endif
    bsf     RX_Pin                          ; set RX Pin As Input for recept
    bcf     _rp0                            ; select Page 0 for Port Access
    return

;*****************************************************************************
```

```
Delay01 movwf   TEMP    ; zakasnitve v 0.1 sekundah, kar podamo z W
_d1     movlw   0xff
        movwf   tmp1
_d2     movlw   0xff
        movwf   tmp2
        clrwdt
_d3     decfsz  tmp2, 1
        goto    _d3
        decfsz  tmp1, 1
        goto    _d2
        call    ToggleLed
        decfsz  TEMP, 1
        goto    _d1
        return

ToggleLed       ; Prizge ali ugasne LED
        btfss   LED
        goto    _tLED
        bcf     LED
        return
_tLED   bsf     LED
        return


;Izpise byte v HEX, ki je podan v W na UART v obliki '0x%x '
; W register se ohrani !

PrintHex
        movwf   TEMP
        ifdef   CPRINT
        movlw   '0'
        call    uarttx
        movlw   'x'
        call    uarttx
        endif

        swapf   TEMP, W
        andlw   0x0F
        sublw   0x09
        btfss   _carry ; vec kot 0x0A
        addlw   -( 'A' - '9' - 1)
        sublw   0x09   ; man kot 0x0A
```

```
        addlw   '0'
        call    uarttx

        movf    TEMP, W
        andlw   0x0F
        sublw   0x09
        btfss   _carry ; vec kot 0x0A
        addlw   -( 'A' - '9' - 1)
        sublw   0x09   ; man kot 0x0A
        addlw   '0'
        call    uarttx
        ifdef   CPRINT
        movlw   ' '
        call    uarttx
        endif
        movf    TEMP, W
        return
;
;*********************************************************************


SetupKBtimer:    ; Inicializira stevec
        bcf     _rb0
        bsf     KEYBOARD
        bsf     _rb0
        bsf     KEYBOARD
        bcf     _rb0
ResetKBtimer:   ; Resetira stevec
        clrf    KBTIMER+0       ; LSB
        clrf    KBTIMER+1
        clrf    KBTIMER+2
        clrf    KBTIMER+3       ; MSB
        return


CheckKeyboard:        ; Preveri stevec in poslje ce ni prazen
        btfsc   KEYBOARD
        goto    _kbflushtest
        call    _incKBtimer  ; Poveca stevec za 1 in se vrne : PIN LOW
        retlw    0x01
_kbflushtest            ; Testira ali je prazen buffer   : PIN HIGH
        movf    KBTIMER+0, W
        iorwf   KBTIMER+1, W
```

```
        iorwf   KBTIMER+2, W
        iorwf   KBTIMER+3, W
        btfss   _z
        goto    _kbflush
        retlw   0x01
        movf    KBTIMER+1, 1
        btfss   _z
        goto    _kbflush
        movf    KBTIMER+2, 1
        btfss   _z
        goto    _kbflush
        movf    KBTIMER+3, 1
        btfss   _z
        goto    _kbflush
        retlw   0x01          ; Already flushed
_kbflush        ; Flush keyboard
        movlw   'k'
        call    uarttx
        movlw   'b'
        call    uarttx
        movf    KBTIMER+3, W
        call    PrintHex
        movf    KBTIMER+2, W
        call    PrintHex
        movf    KBTIMER+1, W
        call    PrintHex
        movf    KBTIMER+0, W
        call    PrintHex
        call    printcrlf
        call    ResetKBtimer
        retlw   0x02

_incKBtimer             ; Poveca stevec za 1 in se vrne : PIN LOW
        incf    KBTIMER+0, 1
        btfss   _z
        retlw   0x00
        incf    KBTIMER+1, 1
        btfss   _z
        retlw   0x00
        incf    KBTIMER+2, 1
        btfss   _z
        retlw   0x00
        incf    KBTIMER+3, 1
```

```
        retlw   0x00


AbortGetChar:
   if _USE_RTSCTS
    bsf     _RTS                    ; Half duplex mode, transmission mode, a
   endif
    bcf _rp0
    bcf _rtie           ; disable further interrupts
    bcf _rcvProgress
    bcf _rcvOver        ; Byte Received, Can RCV/TXMT an other Byte
    return

;
;*****************************************************************************

    include "txmtr.asm"                 ; The Transmit routines are in file
    include "rcvr.asm"                  ; The Receiver Routines are in File
        include "ibutton.asm"                   ; The Touch Memory


        ;; EEPROM Info
        org     H'2100'
        de     "Lecad Gatekeeper v0.1.2 (C) 1997  Leon.Kos@uni-lj.si ", 0

        END
```

3 Zaključek

Prikazani so le bistveni deli programa. Izpuščen je del za komunikacijo z računalnikom. Protokol komunikacije je razviden iz program. V praksi se je izkazalo, da je odziv sistema zelo hiter in zanesliv ter tako primeren tudi za industrijske aplikacije. V diskusiji o uporabnisti sistema, se je pokazala možnost industrijske aplikacije sistema z nadomestitvijo navadne z elektronsko ključavnico. Prozvajalec Titan kamnik bi bil zainteresiran za takšno izvedvo, vendar bi bilo potrebno predhodno rešiti sistem samonapajanja sistema in mehanika izvedbe električnega mehanizma odpiranja.

Literatura

[1] Dallas Semiconductors. Ds1992, 1993, 1994 1kbit/4kbit touch memory. Technical report, Dallas Semiconductors, 1996. http://www.dalsemi.com/DocControl/PDFs/pdfindex.html.

[2] Dallas Semiconductors. Ds1994 memory/time ibutton. Technical Report App Note 60, Dallas Semiconductors, 1996. http://www.dalsemi.com/DocControl/PDFs/appindex.html.

[3] Dallas Semiconductors. Front panel ibutton holder. Technical Report DS1401, Dallas Semiconductors, 1996. http://www.dalsemi.com/DocControl/PDFs/pdfindex.html.

[4] Dallas Semiconductors. Reading and writing ibuttons via serial interfaces. Technical Report App Note 74, Dallas Semiconductors, 1996. http://www.dalsemi.com/DocControl/PDFs/appindex.html.

[5] Dallas Semiconductors. Understanding and using cyclic redundancy checks with dallas semiconductor ibutton products. Technical Report App Note 27, Dallas Semiconductors, 1996. http://www.dalsemi.com/DocControl/PDFs/appindex.html.

[6] Microchip Semiconductors. Pic 16c84 risc microcontroller. Technical report, Microchip Semiconductors, 1996. http://www.microchip2.com/products/micros/base/index.htm.

[]