```
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        ; WONG KAH QUAN
        ; HA18015
        ; MIKRORECHNER
        ; nelsonwongisme@gmail.com

        ; 015_LetzteHausfgabe_FinalEx
        ; Datum: 12/10/2020
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

#include <p16f648A.inc> ;MC type
    errorlevel -302     ;supress the 'not in bank0' warning

    cblock  0x20    ; Start variables @ 0x20
TicCnt          ; Tick Counter
Temp            ; Temporary Register to store temporary data
FSMStateP       ; FSM's State Pointer, indicates the current state of the system.
FSMTime         ; Time delay duration in FSM, used for Timeout down counting.
InPort          ; Input Port Shadows
OutPort         ; OutPut Port Shadows
    endc

    cblock  0x70    ; Flags and save
w_save              ; save workreg.
STATUS_save         ; save Flags Z, C etc.
Flags               ; System flags
OutFlags            ; Write to ports by drivers
InFlags             ; Read from Input Ports
    endc

;   define Flags inside Flags register as Flags,#
; in file register of Flags:
; Bit 0 = TicFlag, Bit 1 - SekFlag, Bit 2 = FSMF
TicFlag     equ     0
SekF        equ     1
FSMF        equ     2   ; if State transition is taken place ---> FSMF = 1
                        ; Previous State != Next State ---------> FSMF = 1
;
;   Portbelegung
; Bb, Bell Button, is a stimulus, a sensor.
Bb      equ     0   ; Pin 0 of PORTA activ high, RA0.

;
;  define event bits in InFlags
BbF         equ     0  ; Bit 0 of InFlags is BbF
nBbF        equ     1  ; Bit 1 of InFlags is nBbF

; Define Outputs to OutFlags --> OutPort --> PORTB
; Below literals are the Bit addresses
; Define bits in OutFlags
; Output Port: Bit 0 = DoorBell, Bit 1 = HL, Bit 2 = DL.
Doorbell    equ     0   ; Door Bell: DoorBell = 1, it rings, else, not ringing.
HL          equ     1   ; Hallway Light: HL = 1, Lights on, else, off.
DL          equ     2   ; Door Lock: DL = 1, Door is unlock, else, lock.
; More about the Output in the Table below.

;   define constant values
T0reload    equ     0x00    ; TMR0 reload value
Ticload     equ     d'255'  ; Downcounts from 255
;-------------------------------------------------------------
    org 0   ;start with code memory adr. 0
    goto    Start   ; jump to Start
    nop             ; No operation
    nop
    nop
;-------------------------------------------------------------
ISR         ;Interrupt service at adr. 4
    movwf   w_save  ;no Z Flag change
    movf    STATUS,w
    movwf   STATUS_save
ServiceT0   ; no other source enabled
```

```
    bcf     INTCON,T0IF     ; TMR0 Overflow Interrupt Flag bit must be cleared in software
                            ; Timer0 interrupt is generated when the TMR0 register
                            ; timer/counter overflows from FFh to 00h. This overflow
                            ; sets the T0IF bit. The interrupt can be masked by
                            ; clearing the T0IE bit (INTCON<5>). The T0IF bit
                            ; (INTCON<2>) must be cleared in software by the
                            ; Timer0 module interrupt service routine before reenabling this
interrupt.
    bsf     Flags,TicFlag
    movlw   T0reload        ; T0reload = 0 --> WREG --> TMR0
    movwf   TMR0            ; TMR 0 = T0reload = 0x00.
Exit
    movf    STATUS_save,w
    movwf   STATUS      ;from here no change of Z flag
    swapf   w_save,f
    swapf   w_save,w
    retfie
ISR_e
;----------------------------------------------------------------

Start

PortInit             ; Initialize PortA as Input and PORTB as Output
    banksel PORTA    ; Select Bank where PORTA resides.
    clrf    PORTA
    clrf    PORTB
    banksel TRISA
    movlw   0xff
    movwf   TRISA
    clrf    TRISB
    banksel CMCON  ;switch to page with comparator contrl
    movlw   0x07
    movwf   CMCON
    banksel VRCON
    clrf    VRCON  ;deactivate voltage reference
    banksel OPTION_REG
    movlw   B'11000000' ; maskword to disable PORTB pullups,
                        ; Interrupt on rising edge of RB0/INT pin.
    iorwf   OPTION_REG,f
    banksel PORTA
PorInit_e

Timer0Init          ; Initialize TMR0
    banksel OPTION_REG  ; Select bank of register of OPTION_REG
    movlw   B'11000000'
    andwf   OPTION_REG,f
    movlw   B'00000111'
    iorwf   OPTION_REG,f ; OPTION_REG's current bits: 11000111
                         ; use Internal instruction cycle clock (CLKOUT)
                         ; Prescaler is assigned to the Timer0 module
                         ; Prescaler Rate for TMR0 --> 1:256
    movlw   B'10100000'
    movwf   INTCON       ; INTCON's current bits: 10100000
                         ; All un-masked interrupts ENABLED
                         ;      > Un-masked interrupt is a hardware interrupt,
                         ;        in our case, Bell Button(Bb))
                         ;
                         ; All peripheral interrupts DISABLED. Peripheral interrupt is any interrupt
other than TMR0, INT, or PORTB change.
                         ; TMR0 interrupt ENABLED
                         ;
                         ; Note that Interrupt requests are asynchronous events which means that an
interrupt request
                         ; can occur at any time during the execution of a program.
    banksel TMR0         ; Select bank of register of TMR0, then copy literal of T0reload to TMR0
    movlw   T0reload
    movwf   TMR0         ; T0reload --> WREG --> TMR0
                         ; TMR 0 = T0reload = 0x00.
Timer0Init_e

TicTacInit  ; Initialize TicTac
```

```
    movlw   Ticload
    movwf   TicCnt  ; TicCnt = Ticload
TictacInit_e

FSMInit     ; Initialize Finite State Machine, initially the State is RESET.
            ; Clear the Flags
    clrf    FSMStateP
    clrf    Flags
FSMInit_e

loop        ; Loop forever

ShadowIn                ; PORTA is the Input of this microcontroller.
    movf    PORTA,w
    movwf   InPort      ; InPort = PORTA
                        ; It reads data from PORTA, then copy to InPort for further processes.
ShadowIn_e

TicTac                      ; Countinuos down counting.
    bcf     Flags,SekF
    btfss   Flags,TicFlag   ; Initially, TicFlag = 1. Will skip "goto TicTac_e", to start to
decrement.
                            ; If TicFlag = 0, means down counting already started. 2nd loop and more,
not at the first loop of the counting cycles.
    goto    TicTac_e
    bcf     Flags,TicFlag   ; Clear TicFlag to note Down Counting starts.
    decfsz  TicCnt,f        ; Decrement the Tick Counter. iF zero, will skip "goto TicTac_e" to
reload the count(Literal from Ticload), as counting is finished.
    goto    TicTac_e
    movlw   Ticload         ; This line executes when TicCnt = 0.
                            ; Reload the Tick Counter, by load literal of TicLoad to WREG then to
TicCnt. Then, set SekF bit in Flags f register.
    movwf   TicCnt          ; TicLoad --> WREG --> TicCnt
                            ; TicCnt = Ticload(d'255' here, it will downcount from 255).
    bsf     Flags,SekF
TicTac_e

BellDrv       ; Bell Driver, to check for Bell Button pressed or not.
    bcf     InFlags,BbF
    bsf     InFlags,nBbF    ; First, clear the BbF(Bb Flag) and set nBbF(Not Bell Button Flag)
                            ;  Then go to next line to test if the Bb is pressed/fired or not.
    btfss   InPort,Bb       ; gedruckt = 1, nicht gedruckt = 0. Always check if the Bb is pressed.
                            ; If pressed, it will skip "goto BellDrv_e" and Set BbF = 1.
    goto    BellDrv_e
    bsf     InFlags,BbF
    bcf     InFlags,nBbF    ; This and previous line executed when Bb is fired.
                            ; Once Bb is fired, Bb = 1, it will set the Bell Button's Flag
                            ; Notifying the Microcontroller that Bell Button is fired.
BellDrv_e

; in FSM:
; Initially, it checks for FSMF, check whether there's state transition or not.
; If the current state is same as the previous state, will not execute FSMdo1, here, it will set the
duration of the timeout delay of the
; current state into FSMTime and return Outpattern to WREG then to OutFlags(which thne later will
copy to PORTB as output of the system).
; Then, goto FSMd01.
; But when the FSMF = 0, the previous state and the current state is same, means there's no state
transition.
; will jump to FSMdo1 straight away.
; More explanation below.

FSM         ;Tabellen gesteuerter Zustandsautomat/Finite State Machine
    movf    FSMStateP,w     ; initially FSM's State Pointer is zero. Start from S0.
    movwf   Temp            ; Copy data from WREG to Temp to store temporary. Data to be passed to
WREG back for W's offset in table of OutPattern
    btfsc   Flags,FSMF      ; Test bit of FSMF.
                            ; If FSMF = 1: There's state transition, the previous and next state is
different.
                            ; Skip "goto    FSMdo1" when there's no state transition, namely FSMF = 0.
                            ; When there's no state transition, no skip to FSMdo1, will set big of
```

```
FSMF and so on.
                            ; Initially FSMF = 0, then once start looping, FSMF will flag depends on
state change at Code Block of FSMexit,
                            ; then FSMF = 1, will loop back here to execute FSMdo1 and so on.
    goto    FSMdo1
    bsf     Flags,FSMF      ; executed if FSMF = 0 in bit test of FSMF in Flags, i.e., execute when
no state transition occurs.
                            ; It will set FSMF = 1. Then invoke Zeit table and produce its
corresponding output. If the state not depends on
                            ; delay timeout to change state, it will return the same literal of state
number in TimeExit.
                            ; For example, S1 --> S1 if it only listen to stimulus such as button.
    call    Zeit
    movwf   FSMTime         ; Time obtained from Zeit's Table will copy to WREG then FSMTime. That
Particular State's Time.
    movf    Temp,w
    call    Outpattern      ; initially StatePointer is zero. Start from S0. So will call Outpattern
for S0.
    movwf   OutFlags        ; RETLW Literal from Table to WREG then to OutFlags.

FSMdo1                  ; Test timeout & go to next state(depends on stimulus)
    movf    FSMTime,f   ; Copy FSMTime to W. Then test bit.
    btfss   STATUS,Z    ; Z flag is set if f == 0(FSMTime is f register here).
                        ; Z = 1, result of logical or operation is zero, means that FSMTime = 0,i.e.,
the Timeout Delay finished down counting.
                        ; It then will skip "goto FSMdo2" to "call TimeExit", which is the next state
after delay timeout.
    goto    FSMdo2
    movf    Temp,w      ; execute this if Z is set.  The result of an arithmetic or logic operation
is zero
    call    TimeExit    ; get next state for timeout. The next state's state's number is then store
to WREG
    movwf   FSMStateP   ; from WREG, which contains the Next state, now move to FSMStateP.

FSMdo2                  ; Test for Downcount lifetime & Down counting
    btfss   Flags,SekF
    goto    FSMdo3
    decf    FSMTime,f   ; Decrement FSMTime if SekFlag(1st bit of Flags) is zero
                        ; It will downcount depends on the duration of the timeout delay of that
current state.
                        ; the duration information/data obtain from FSMTime.

FSMdo3                  ; Test Bell Button & go to next state(depends on stimulus)
    movf    Temp,w
    call    EventMask   ; Call EventMask table to check which state and check if that state listen to
Bb or not.
    andwf   InFlags,w   ; InFlags: Bit 0 = BbF, Bit 1 = nBbF
    btfsc   STATUS,Z
    goto    FSMexit     ; Skip if Z = 0. I.e, when the result of operation is not zero. Both WREG and
InFlags are set in the LSB.
                        ; Execute this, when Z = 1, where BbF AND W is 0. Means, either BbF or W are
0, or both. WREG is from EventMask from Temp.
                        ; Z must be 0 to skip this and call next state.
    movf    Temp,w      ; Bb is fired and EventMask = 1
                        ; Execute this if result of the logical operation is not zero, where Z = 0.
                        ; In others word, BbF = 1. Bb is pressed and the state is a state that listen
to Bb Stimulus.
    call    EventExit   ; Next state after event/condition
    movwf   FSMStateP   ; EventExit -> FSMStateP. Now system/machine is in that state.
                        ; This code block is to change to FSMStateP to the apporpriate State.

FSMexit                 ; Check Previous State & Next State same or not, then exit FSM and produce
output and loop again.
                        ; FSMF is the FSM's Flag.
                        ; No state transition is taken place --> FSMF = 0    [PS  = NS]
                        ; State transition is taken place    --> FSMF = 1    [PS != NS]
    movf    Temp,w
    xorwf   FSMStateP,w ; Z Flag if no change
    btfsc   STATUS,Z    ; FSMStateP XOR W , will skip "goto FSM_e" if WREG = FSMStateP
                        ; else FSMStateP != WREG, will execute "goto FSM_e"
                        ; Exit FSM when FSMStateP != WREG, as Z = 1. WREG is the StatePointer of
```

```
previous state.
                        ; If Previous State is different than the next state, means there is state
change.
                        ; Then exit FSM and loop again with FSMF = 1.
    goto    FSM_e       ; Execute if Next State != Previous State, means state transition happened.
    bcf     Flags,FSMF  ; If FSMStateP XOR W = 1, Z = 0, this line will be executed., clear FSMF in
Flags. Not Flagged, still at S0.
                        ; Executed when FSM Previous State is same as the Next state. (NS = PS)
                        ; No state transition is taken place,then FSMF = 0, where FSMF is the FSM's
Flag.
FSM_e

OutDrv
    movf    OutFlags,w
    movwf   OutPort     ; OutFlags -> WREG -> OutPort
OutDrv_e

ShadowOut
    movf    OutPort,w
    movwf   PORTB       ; OutPort -> WREG -> PORTB
                        ; PORTB is the Output of this microcontroller.
ShadowOut_e

    goto loop
;
;-----------------------------------Tabellen;-----------------------------------
;           States
;           S0      S1      S2      S3      S4      S5      S6      S7
; Literal are in HEX, otherwise stated.

Zeit        addwf   PCL,f
    dt      0,      3,      7,       5,      3,      2,     d'20',   2

    ; Values are in Seconds.
    ; These are the duration needed to change state depends for the state that
    ; uses timer as interrupt.

Outpattern  addwf   PCL,f
    dt      0,      3,      2,       0,      0,      4,      0,      0

    ; Outpattern that will be the output at PORTB
    ; 0x00 : B'00000000', Hallway Light is OFF and DoorBell is NOT ringing, Door is LOCKED
    ; 0x02 : B'00000010', Hallway Light is ON and DoorBell is NOT ringing, Door is LOCKED
    ; 0x03 : B'00000011', Hallway Light is ON and DoorBell is ringing, Door is LOCKED
    ; 0x04 : B'00000100', Hallway Light is OFF and DoorBell is NOT ringing, Door is UNLOCKED

TimeExit    addwf   PCL,f
    dt      0,      2,      3,       6,      6,      7,      0,      0

    ; State will go to Next State after timeout:
    ;
    ;   S0 :    Only depends on Bb
    ;   S1 :    S1 --> S2
    ;   S2 :    S2 --> S3
    ;   S3 :    S3 --> S6
    ;   S4 :    S4 --> S6
    ;   S5 :    S5 --> S7
    ;   S6 :    S6 --> S0
    ;   S7 :    S7 --> S0
    ;
    ;   The duration needed to change state depends
    ;   on the duration of the delay in table "Zeit"(See above).
    ;   Only depends on Bb: Means that this state only change reacting to Bell Button,
    ;   instead of change state when dleay timeout. Could be either Bb pressed or released.
    ;   This state will waiting for stimulus to be stimulate to change to next state.

EventMask   addwf   PCL,f
    dt      1,      0,      1,       1,      2,      0,      0,      0

    ; Bit 0 of InFlags is BbF
    ; Bit 1 of InFlags is nBbF
    ;
    ; 0x01 == B'00000001'   : BbF = 1, nBbF = 0
    ; 0x02 == B'00000010'   : BbF = 0, nBbF = 1
    ; 0x00 == B'00000000'   : Not stimulated by Bell Button(Bb), means this state does not listne to
Bb

EventExit   addwf   PCL,f
    dt      1,      0,      1,       4,      5,      0,      0,      0

    ; Event Exit
    ; EvEx(Event Exit): Next state after event/condition(e.g.Bb executed)
    ;
    ; When Bb is fired or BbF = 1:
    ;   S0 :    S0 --> S1
    ;   S1 :    Does not listen to Bb
    ;   S2 :    S2 --> S1
    ;   S3 :    S3 --> S4
    ;   S4 :    S4 --> S5
    ;   S5 :    Does not listen to Bb
    ;   S6 :    Does not listen to Bb
    ;   S7 :    Does not listen to Bb
    ;
    ; Listen/Stimulate : this state won't chnage state when Bb is fired or not fired
    ;                    Only depends on the delay timeout to change state.

    END

; Some Explanations for Table:
; 1 - The OutPattern to be invoke will RETLW to OutFLags then OutPort then PORTB, then output.
;     means if bit of, say DL(4th bit of OutFLags/OutPort/PORTB), is 00001000, only DL is HIGH, means
DoorLock enabled.
;     [Reference: .pdf of this Hausaufgabe, Page ]

; 2 - For the addwf PCL,f ==> add W to PCL and store in PCL. PCL is program counter. So to jump to
the specify column of
;     table. PCL + W = Address location of that RETLW, so the literal can be return to the WREG.
;     Hence, the WREG before invoking the table is equal to the column's index.
;     In other words, offsetting in the Program Counter with the help of WREG(WREG here/now stores
the FSMStateP, which is the current state).
;     For example: If FSMStateP or the WREG = 2 (i.e., State 2/S2), then PCL = PCL + 2. It will
return the literal in the 2nd column
;     of the table.

; Note(to self):
; > Don't put bit in Wtach Tab, only bytes or register
; > Need to take note on register, byte, bits, flag bit and enable etc.
; > Don't forget to disable WDT and use INTOSC for CLKOUT.
; > Don't forget to enable Real-Time updates of simulation
```