Arquitetura de Computadores

Trabalho Prático 3

Data de entrega e discussão: 24-05-2024

Faculdade de Ciências Exatas e da Engenharia

Ano letivo 2023/2024

2º Ano 2º Semestre

Pedro Brito 2022622

Índice

Introdução

Este relatório foi realizado no âmbito de descrever o trabalho realizado para o 3º trabalho prático de Arquitetura de Computadores. Este trabalho teve como principal objetivo implementar um programa em assembly e C que é capaz de realizar uma contagem de tempo e também é capaz de registar as respostas do utilizador. Este sistema seria desenvolvido no âmbito de, por exemplo, um concurso de respostas. Para isto foi utilizado o software Keil Uvision 5 para produzir o código do programa e consequente simulação do mesmo.

Foi desenvolvimento pelo grupo docente um circuito que trava de realizar as ligações entre o microcontrolador, os displays de 7 segmentos e os botões necessários. Este circuito está representado na figura 1.

O botão B1 é responsável por iniciar a contagem de 5 segundos até 0 segundos. Caso não haja nenhuma interrupção através dos botões de resposta BA, BB, BC e BD é apresentado, alternadamente em 1 segundo, o tempo “0.0” e a resposta “-.-“. Para reiniciar o programa o utilizador clica novamente no botão B1 onde é mostrado os 5 segundos novamente. Caso o utilizador clique nalgum botão de respostas durante a contagem, inicia-se a alternação entre resposta/tempo com 1 segundo em que a resposta corresponde ao botão clicado e o tempo corresponde ao tempo restante que o utilizador

tinha para responder, ou seja, corresponde ao tempo que estava representado no display aquando da seleção de resposta.

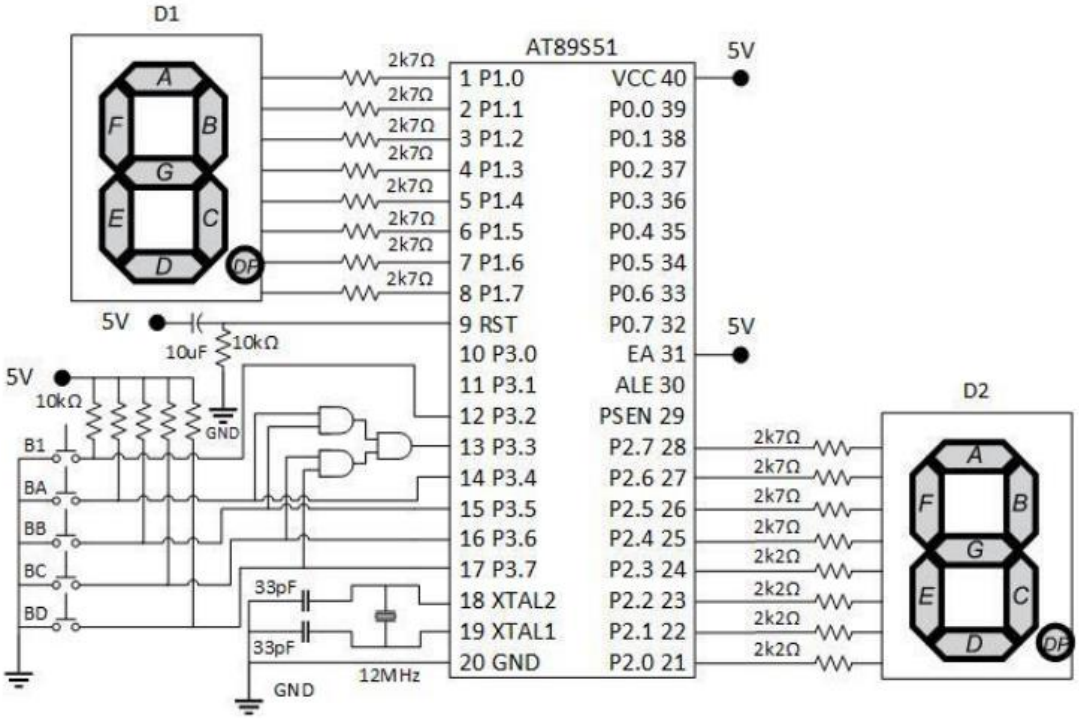


Figura 1 - Esquema de ligações do microcontrolador

Tabela 1 - Mapeamento dos Pinos do microcontrolador para o Display

Objeto	Pino do microcontrolador
DISPLAY 1	
A	P1.0
B	P1.1
C	P1.2
D	P1.3
E	P1.4
F	P1.5
G	P1.6
DP	P1.7
DISPLAY 2	
A	P2.0
B	P2.1
C	P2.2
D	P2.3
E	P2.4
F	P2.5
G	P2.6
DP	P2.7

Objetivos

- Breve estudo dos requisitos de software para a elaboração do projeto
- Desenho dos principais fluxogramas do controlo e das rotinas secundárias
- Programação em linguagem assembly e linguagem C
- Verificação experimental do programa

Desenvolvimento

Este projeto foi desenvolvimento baseado numa noção de estados. Assim definiu-se 3 estados fundamentais:

- Estado waiting: corresponde ao programa estar com o display “5.0” e à espera que o utilizador carregue no botão B1 para iniciar a contagem
- Estado time: corresponde ao programa estar a realizar a contagem desde os 5 segundos até aos 0 segundos. Este estado pode ser interrompido através do utilizador quando este carrega num botão corresponde a uma das quatro respostas, passo para o estado de resposta.
- Estado answer: corresponde ao programa estar a mostrar a resposta (ou resposta nula (“-.-“)) alternadamente com o tempo restante de 1 em 1 segundos. Este estado é interrompido quando o utilizador carrega no botão B1, voltando ao estado waiting.

Assim, como é percebido, é necessário utilizar expressões condicionais para saber se o estado está ativo ou desativo para poder executar as instruções correspondentes.

Depois, foi necessário corresponder os bits dos segmentos para um valor em hexadecimal. Assim, esta correspondência é descrita na tabela 2.

Tabela 2 - Valores em binário e hexadecimal dos símbolos do Display

Símbolo	Valor em Binário	Valor em Hexadecimal
0.	01000000	40
1.	01111001	79
2.	00100100	24
3.	00110000	30
4.	00011001	19
5.	00010010	12
-.	00111111	3F
0	11000000	C0
1	11111001	F9
2	10100100	A4
3	10110000	B0
4	10011001	99
5	10010010	92
6	10000010	82

7	11111000	F8
8	10000000	80
9	10010000	90
-	10111111	BF
A	10001000	88
B	10000011	83
C	11000110	C6
d	10100001	A1

Interrupções

As interrupções foram uma funcionalidade bastante importante neste trabalho. Assim, decidiu-se ativar 4 interrupções (2 timer e 2 externas). Para fazer isto, foi necessário ativar as interrupções globalmente, ativar cada interrupção e de seguida configurar essas interrupções.

As interrupções timers foram configuradas com um debounce de 50ms para evitar o ruído quando o utilizador pressiona os botões. Assim, como sabe-se que os timers estiveram no modo 1 com 16 bits:

$$65536 - 50000 = 15536\ (3CB0h)$$

Logo, TH = 0x3C e TL=0xB0.

A interrupção Timer 0 foi responsável por fazer contagens de segundos e por ativar a mudança do display 1 (correspondente aos segundos). Também foi responsável por verificar se a contagem tinha excedido os 5 segundos no estado time de modo a dar reset no timer, ativar o estado answer e mostrar no display do tempo restante “0.0”.

A interrupção Timer 1 foi responsável por fazer contagens de 0.1 segundos e por ativar a mudança do display 2 (correspondente às decimas de segundo).

A interrupção externa 0 (ativada quando o utilizador carrega no botão B1) tem 2 funções. Caso o estado ativo seja o estado waiting (estado inicial com “5.0” no display), muda-se para o estado time (inicia-se a contagem). Caso o estado ativo seja o estado answer (alternância de resposta/tempo restante), muda-se para o estado waiting (estado inicial com “5.0” no display).

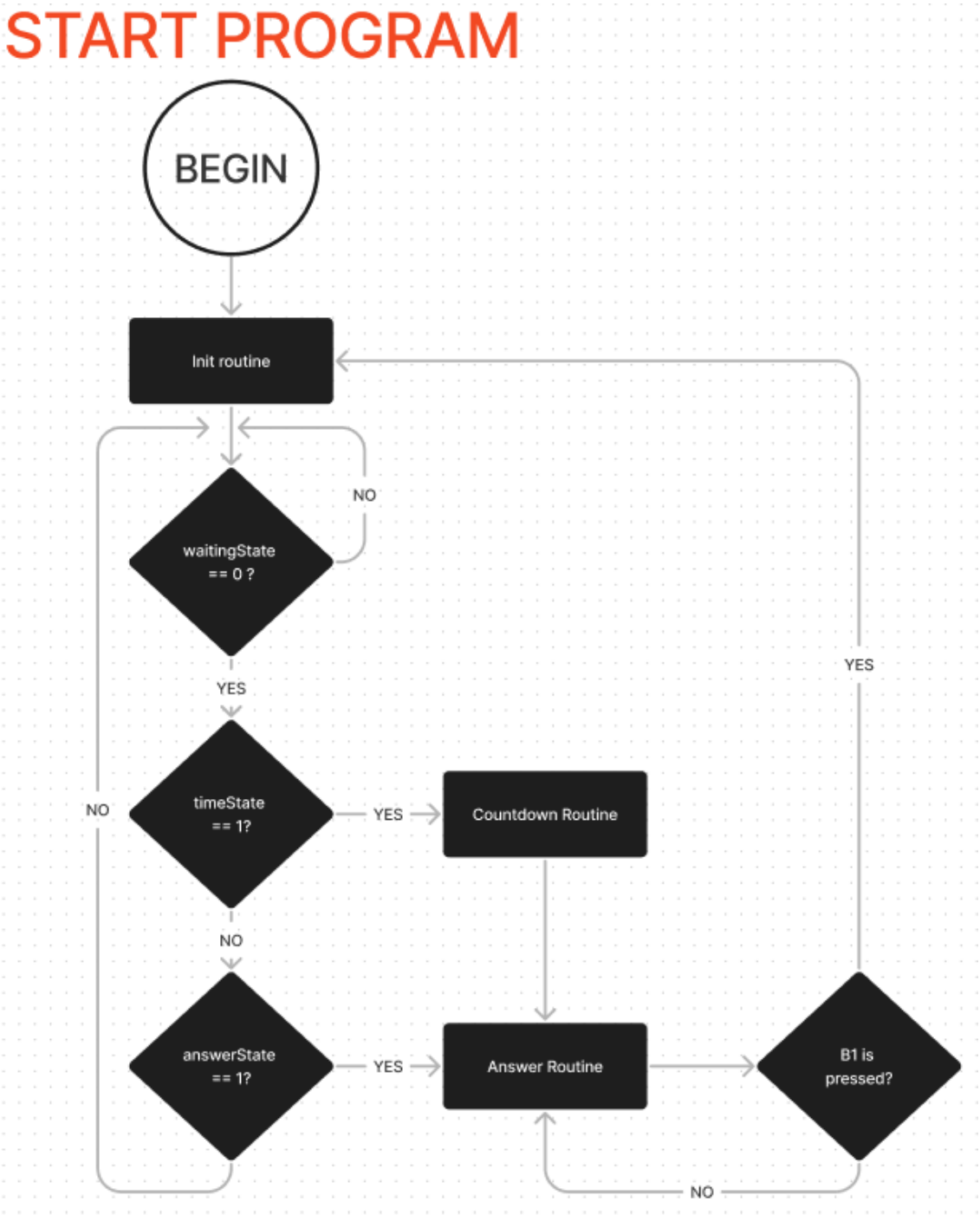
A interrupção externa 1 (ativada quando o utilizador carrega num dos botões de resposta BA, BB, BC ou BD) verifica e ativa a flag desse botão de resposta.

Conclusão

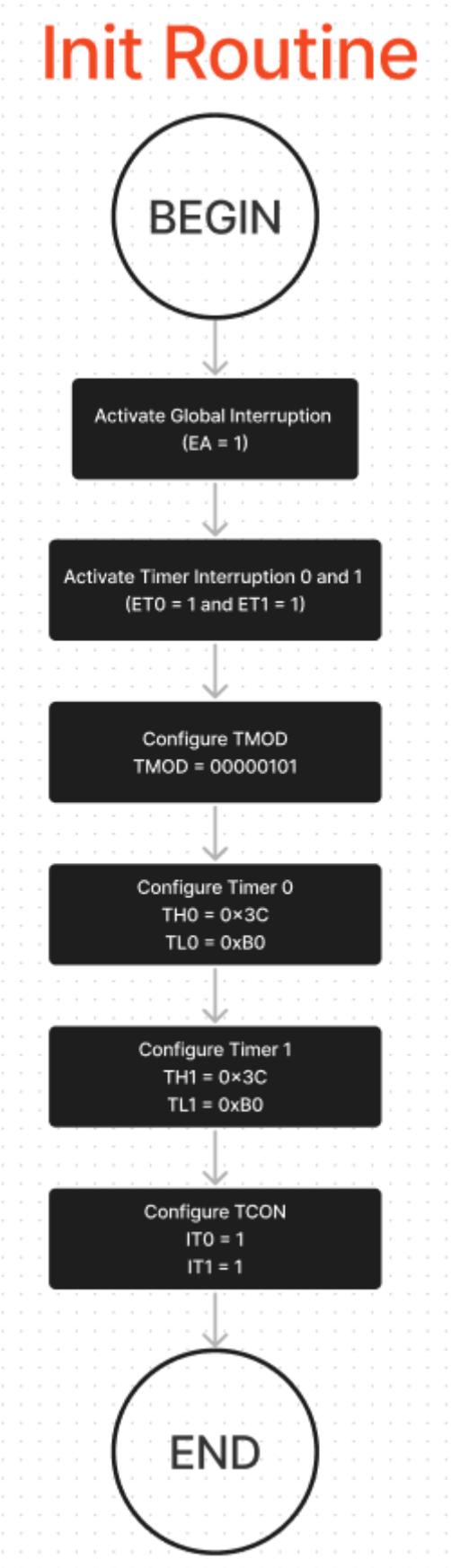
Este trabalho permitiu ter uma melhor noção de como trabalhar e gerir interrupções. Permitiu, embora com mais dificuldade, compreender melhor a linguagem assembly e como gerir registos e memória. A linguagem C é, sem dúvida, mais fácil de trabalhar do que a linguagem Assembly. No entanto, percebe-se que com Assembly o programador tem muito mais controlo sobre a memória e o próprio programa.

Anexo A – Fluxogramas

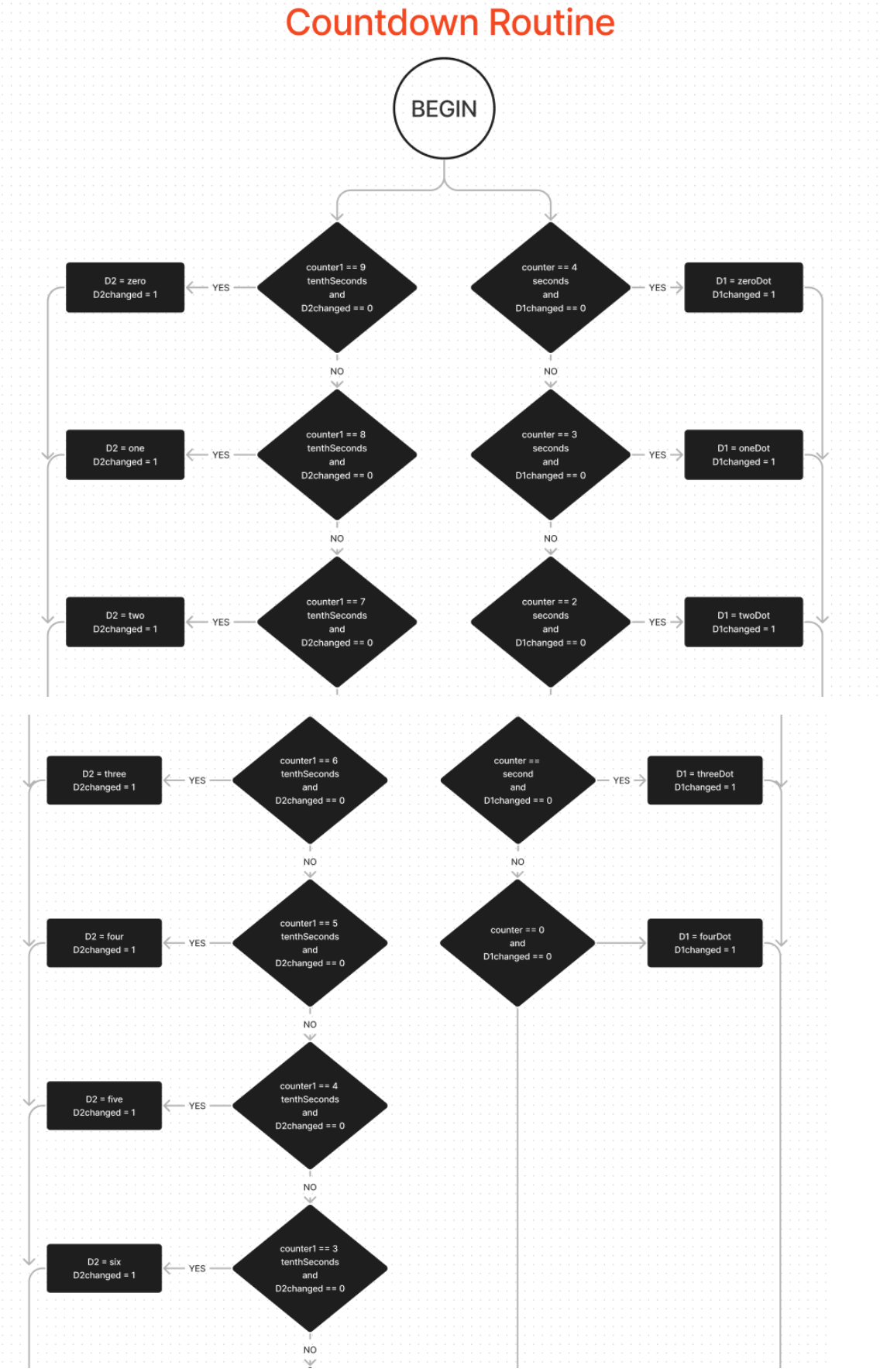
Start Program

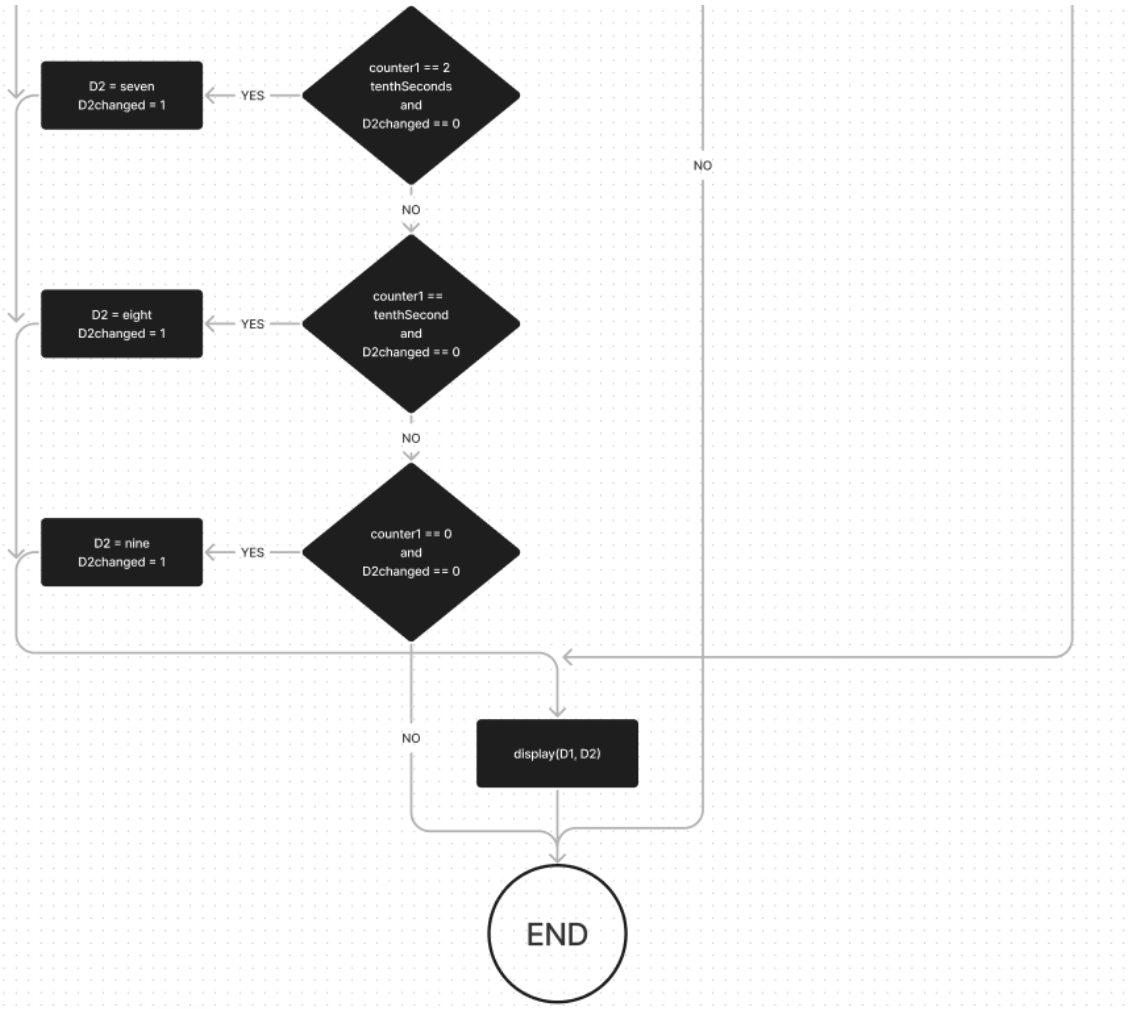


Init Routine

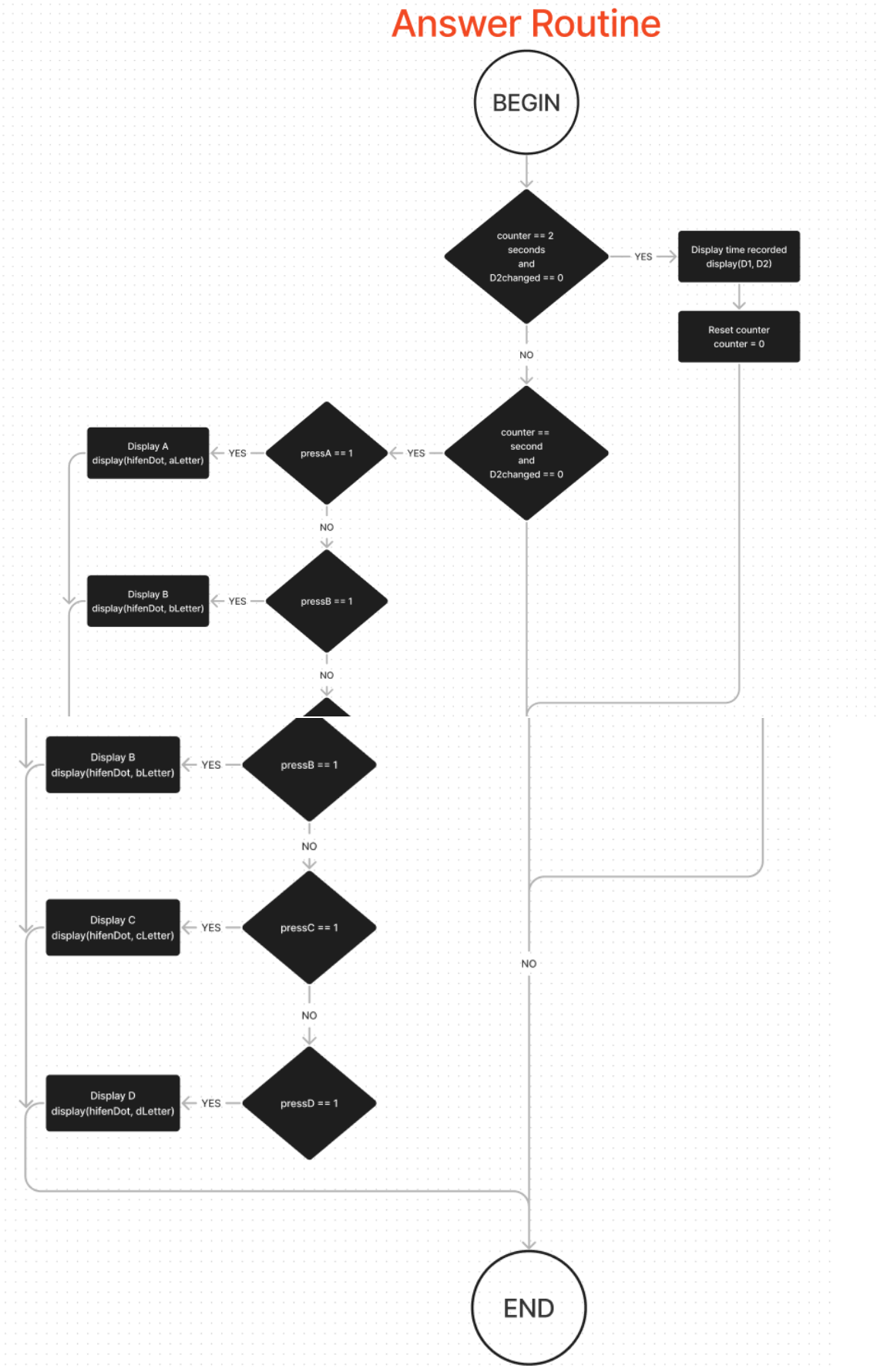


Countdown Routine

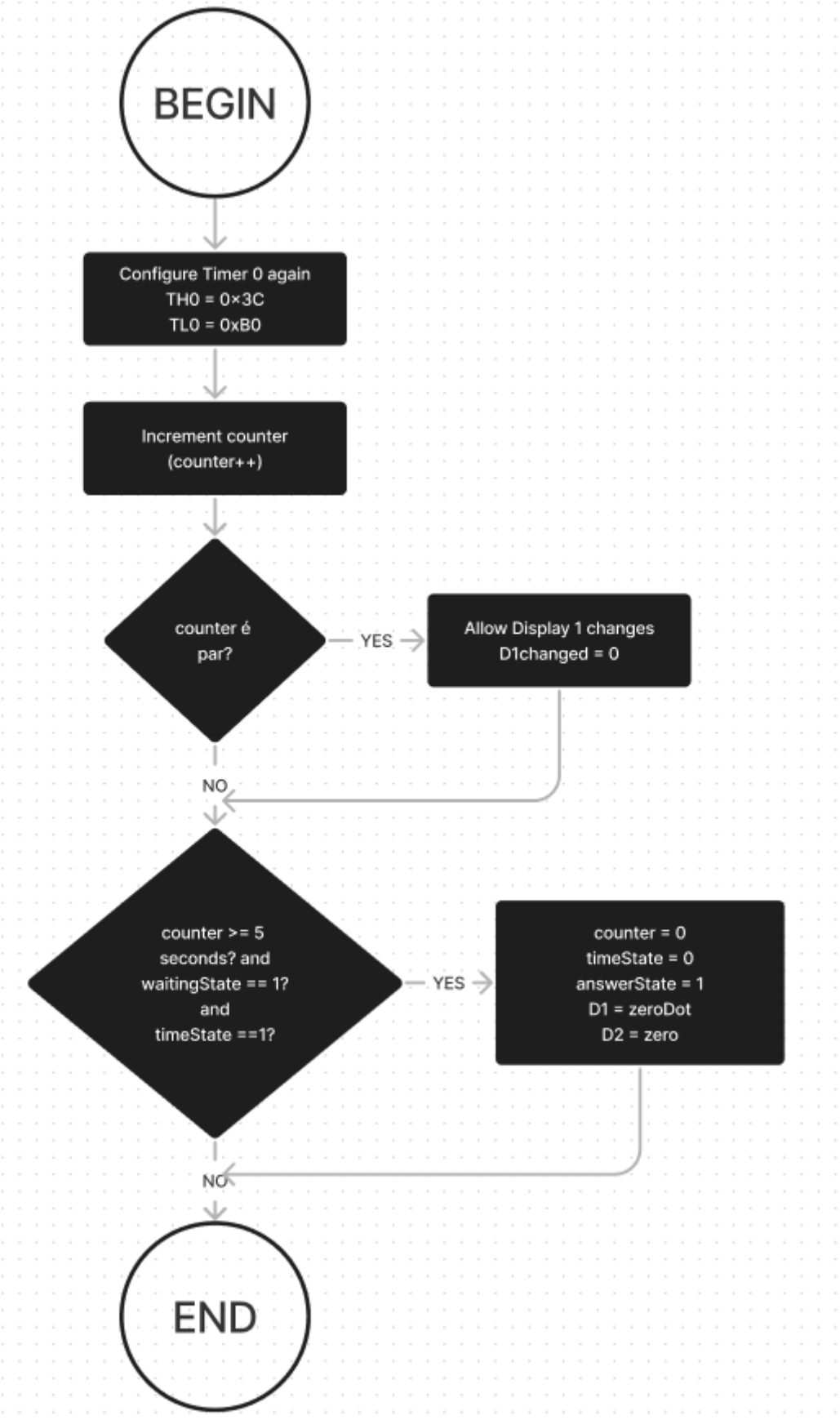



counter1 == 2
tenthSeconds
and
D2changed == 0
YES
D2 = seven
D2changed = 1
NO
counter1 ==
tenthSecond
and
D2changed == 0
YES
D2 = eight
D2changed = 1
NO
counter1 == 0
and
D2changed == 0
YES
D2 = nine
D2changed = 1
NO
NO
display(D1, D2)
END

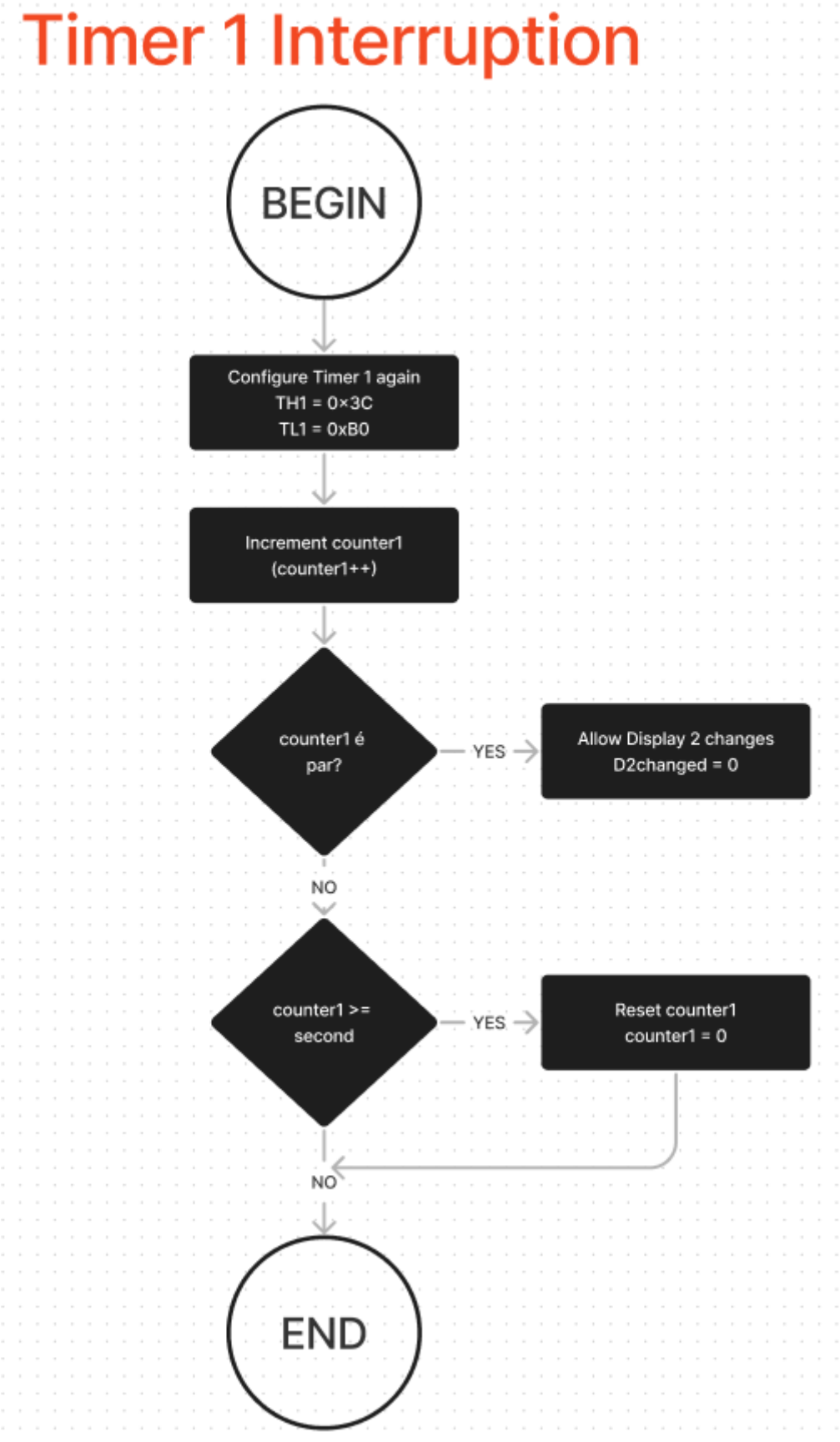
Answer Routine



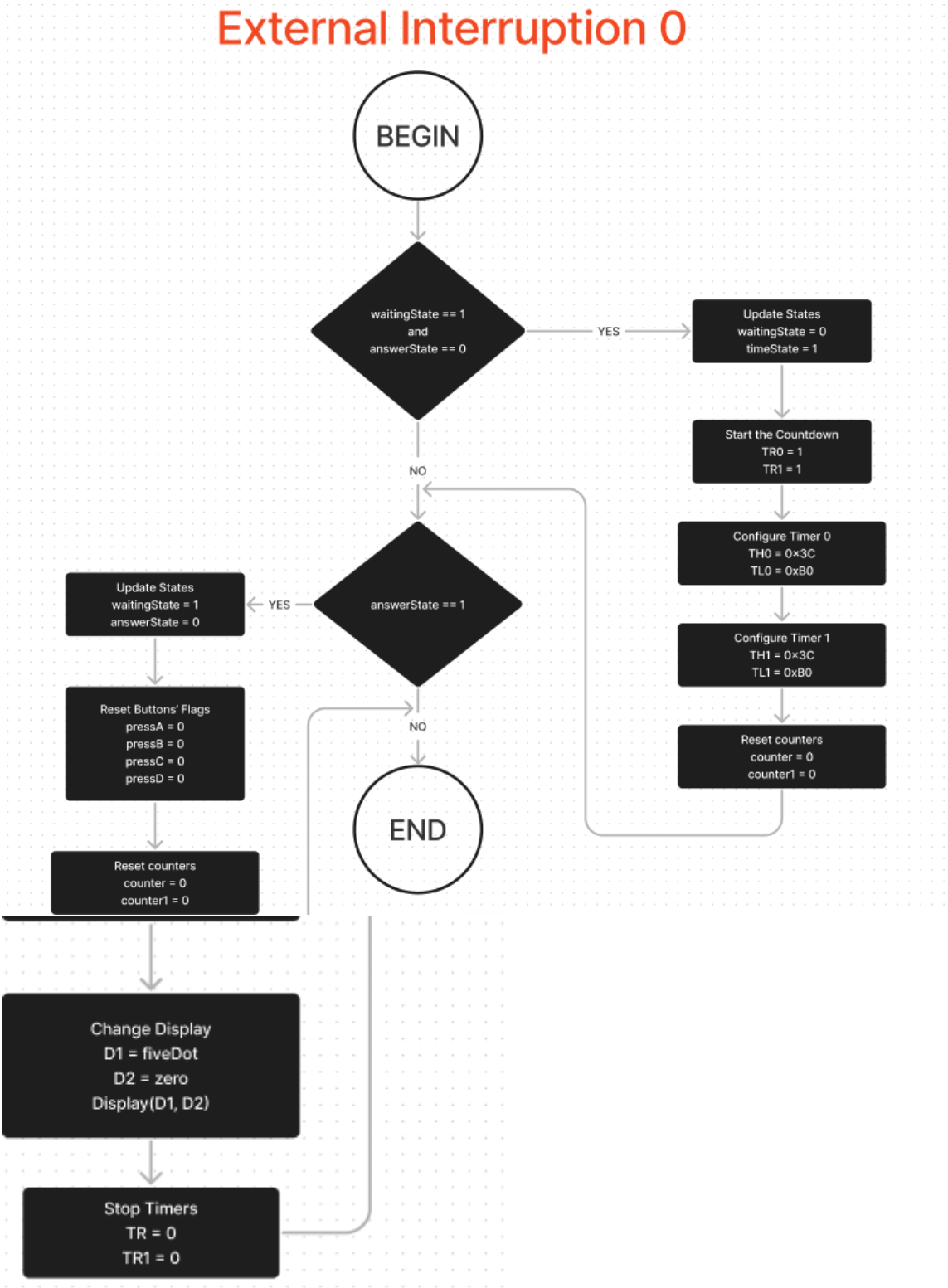
Timer 0 Interruption



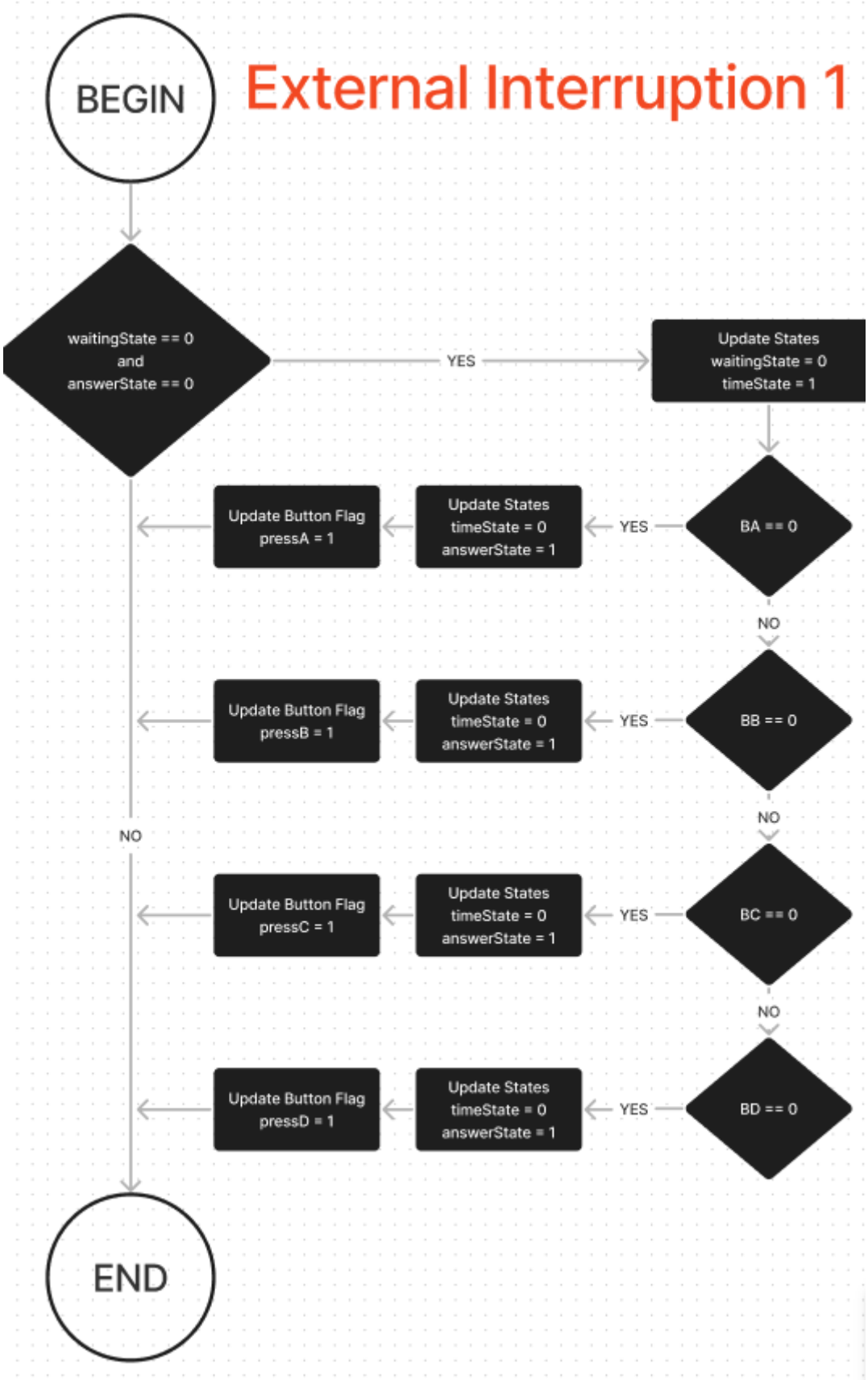
Timer 1 Interruption



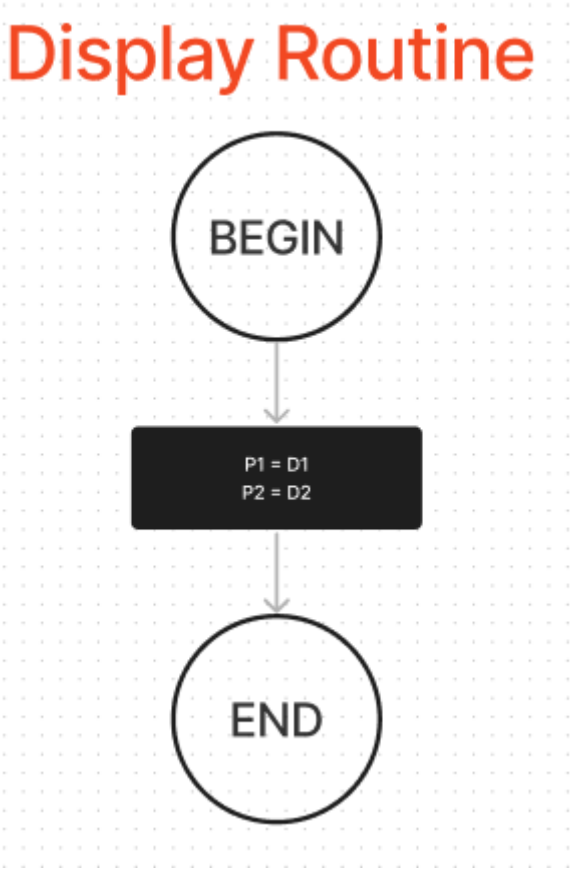
External Interruption 0



External Interruption 1



Display Routine



Anexo B1: Código em linguagem Assembly

```
tenthSecond         EQU     2           ;time to count 0,1 seconds (2 x 50ms)
twoTenthSeconds     EQU     4
threeTenthSeconds EQU   6
fourTenthSeconds    EQU 8
fiveTenthSeconds EQU    10
sixTenthSeconds     EQU     12
sevenTenthSeconds EQU   14
eightTenthSeconds EQU   16
nineTenthSeconds EQU    18
second              EQU     20          ;time to count 1 second (20 x 50ms)
twoSeconds          EQU 40
threeSeconds  EQU       60
fourSeconds   EQU 80
fiveSeconds   EQU 100
THtimer             EQU 0x3C ;Timer 0 - 50ms -> (65536(10000h) - 50000(3E8h) = 15536(3CB0h))
TLtimer             EQU 0xB0 ;Timer 0 - 50ms -> (65536(10000h) - 50000(3E8h) = 15536(3CB0h))

; values for display
null                EQU     0xFF
zeroDot             EQU     0x40
oneDot              EQU     0x79
twoDot              EQU     0x24
threeDot            EQU     0x30
fourDot             EQU     0x19
fiveDot             EQU 0x12
hifenDot            EQU     0x3F
zero                EQU     0xC0
one                 EQU     0xF9
```

```
two                     EQU         0xA4
three           EQU         0xB0
four            EQU         0x99
five            EQU         0x92
six                     EQU         0x82
seven           EQU         0xF8
eight           EQU         0x80
nine            EQU         0x90
hifen           EQU         0xBF
aLetter         EQU         0x88
bLetter         EQU         0x83
cLetter         EQU         0xC6
dLetter         EQU         0xA1



; Buttons definition
BA          EQU     P3^4    ;
BB                      EQU     P3^5    ;
BC                      EQU     P3^6    ;
BD                      EQU     P3^7    ;

; states
waitingState    EQU     1           ; R0 bank 0
timeState               EQU     0           ; R1 bank 0
answerState     EQU     0           ; R2 bank 0
D1changed               EQU     0           ; R3 bank 0
D2changed               EQU     0           ; R4 bank 0

counter                 EQU     0           ; R5 bank 0
counter1                EQU     0           ; R6 bank 0
```

```
D1                      EQU     0               ; R0 bank 1
D2                      EQU     0               ; R1 bank 1
pressA          EQU     0               ; R2 bank 1
pressB          EQU     0               ; R3 bank 1
pressC                  EQU     0               ; R4 bank 1
pressD                  EQU     0               ; R5 bank 1

StackPointer    EQU     0x0F    ; need to point to a new location because of the usage of 2 banks

CSEG AT 0000H
        JMP Main

CSEG AT 0003H
        JMP ExternalInterruption0

CSEG AT 000BH
        JMP TimerInterruption0

CSEG AT 0013H
        JMP ExternalInterruption1

CSEG AT 001BH
        JMP TimerInterruption1

CSEG AT 1000H

Main:

        MOV SP, #StackPointer
        CALL Initializations            ; routine to set registers
```

```
CALL EnableInterruptions       ; routine to enable interruptions and set priorities
CALL ConfigureInterruptions ; routine to configure tmod, timers and tcon

; select bank 1
SETB PSW.3

MOV R0, #fiveDot
MOV R1, #zero

; select bank 0
CLR PSW.3

CALL Display

For:

CJNE R0, #0, CheckAnswerState1; waitingState == 0

CJNE R1, #1, CheckAnswerState1; timeState == 1

CJNE R3, #0, OneSecondR; D1Changed == 0
MOV A, R5                                                    ; A = counter
CJNE A, #fourSeconds, OneSecondR          ; counter == 4 * seconds

        ; select bank 1
        SETB PSW.3


        MOV R0, #zeroDot

        ; select bank 0
```

```
        CLR PSW.3

        MOV R3, #1
        CALL Display
        JMP ZeroTenthSecondsR

OneSecondR:
CJNE R3, #0, TwoSecondsR; D1Changed == 0
MOV A, R5                                   ; A = counter
CJNE A, #threeSeconds, TwoSecondsR          ; counter == 3 * seconds

        ; select bank 1
        SETB PSW.3


        MOV R0, #oneDot

        ; select bank 0

        CLR PSW.3

        MOV R3, #1
        CALL Display
        JMP ZeroTenthSecondsR

TwoSecondsR:
CJNE R3, #0, ThreeSecondsR; D1Changed == 0
MOV A, R5                                   ; A = counter
CJNE A, #twoSeconds, ThreeSecondsR          ; counter == 2 * seconds
```

```
            ; select bank 1
            SETB PSW.3


            MOV R0, #twoDot

            ; select bank 0

            CLR PSW.3

            MOV R3, #1
            CALL Display
            JMP ZeroTenthSecondsR

ThreeSecondsR:
CJNE R3, #0, FourSecondsR; D1Changed == 0
CJNE R5, #second, FourSecondsR ; counter == second

            ; select bank 1
            SETB PSW.3


            MOV R0, #threeDot

            ; select bank 0

            CLR PSW.3

            MOV R3, #1 ; D1changed = 1
            CALL Display
            JMP ZeroTenthSecondsR
```

```
FourSecondsR:
CJNE R3, #0, ZeroTenthSecondsR      ; D1Changed == 0
CJNE R1, #1, ZeroTenthSecondsR      ; timeState == 1
CJNE R5, #0, ZeroTenthSecondsR      ; counter == 0

        ; select bank 1
        SETB PSW.3


        MOV R0, #fourDot

        ; select bank 0

        CLR PSW.3

        MOV R3, #1
        CALL Display
        JMP ZeroTenthSecondsR

CheckAnswerState1:
        JMP CheckAnswerState

ZeroTenthSecondsR:
        CJNE R4, #0, OneTenthSecondsR; D2Changed == 0
        MOV A, R6
        CJNE A, #nineTenthSeconds, OneTenthSecondsR ; counter1 == 9 *
tenthSeconds

        ; select bank 1
        SETB PSW.3
```

```
                MOV R1, #zero

                ; select bank 0

                CLR PSW.3

                MOV R4, #1
                CALL Display
                JMP CheckAnswerState

        OneTenthSecondsR:
                CJNE R4, #0, TwoTenthSecondsR; D2Changed == 0
                MOV A, R6
                CJNE A, #eightTenthSeconds, TwoTenthSecondsR ; counter1 == 8 *
tenthSeconds

                ; select bank 1
                SETB PSW.3


                MOV R1, #one

                ; select bank 0

                CLR PSW.3

                MOV R4, #1
                CALL Display
                JMP CheckAnswerState
```

```
        TwoTenthSecondsR:
                CJNE R4, #0, ThreeTenthSecondsR; D2Changed == 0
                MOV A, R6
                CJNE A, #sevenTenthSeconds, ThreeTenthSecondsR ; counter1 == 7 *
tenthSeconds

                ; select bank 1
                SETB PSW.3


                MOV R1, #two

                ; select bank 0

                CLR PSW.3

                MOV R4, #1
                CALL Display
                JMP CheckAnswerState

        ThreeTenthSecondsR:
                CJNE R4, #0, FourTenthSecondsR; D2Changed == 0
                MOV A, R6
                CJNE A, #sixTenthSeconds, FourTenthSecondsR ; counter1 == 6 *
tenthSeconds

                ; select bank 1
                SETB PSW.3


                MOV R1, #three
```

```
                ; select bank 0

                CLR PSW.3

                MOV R4, #1
                CALL Display
                JMP CheckAnswerState

        FourTenthSecondsR:
                CJNE R4, #0, FiveTenthSecondsR; D2Changed == 0
                MOV A, R6
                CJNE A, #fiveTenthSeconds, FiveTenthSecondsR ; counter1 == 5 *
tenthSeconds

                ; select bank 1
                SETB PSW.3


                MOV R1, #four

                ; select bank 0

                CLR PSW.3

                MOV R4, #1
                CALL Display
                JMP CheckAnswerState

        FiveTenthSecondsR:
                CJNE R4, #0, SixTenthSecondsR; D2Changed == 0
```

```
                MOV A, R6
                CJNE A, #fourTenthSeconds, SixTenthSecondsR ; counter1 == 4 *
tenthSeconds

                ; select bank 1
                SETB PSW.3


                MOV R1, #five

                ; select bank 0

                CLR PSW.3

                MOV R4, #1
                CALL Display
                JMP CheckAnswerState

        SixTenthSecondsR:
                CJNE R4, #0, SevenTenthSecondsR; D2Changed == 0
                MOV A, R6
                CJNE A, #threeTenthSeconds, SevenTenthSecondsR ; counter1 == 3 *
tenthSeconds

                ; select bank 1
                SETB PSW.3


                MOV R1, #six

                ; select bank 0
```

```
                CLR PSW.3

                MOV R4, #1
                CALL Display
                JMP CheckAnswerState

        SevenTenthSecondsR:
                CJNE R4, #0, EightTenthSecondsR; D2Changed == 0
                MOV A, R6
                CJNE A, #twoTenthSeconds, EightTenthSecondsR ; counter1 == 2 *
tenthSeconds

                ; select bank 1
                SETB PSW.3

                MOV R1, #seven

                ; select bank 0

                CLR PSW.3

                MOV R4, #1
                CALL Display
                JMP CheckAnswerState

        EightTenthSecondsR:
                CJNE R4, #0, NineTenthSecondsR; D2Changed == 0
                CJNE R6, #tenthSecond, NineTenthSecondsR ; counter == tenthSeconds

                ; select bank 1
```

```
            SETB PSW.3



            MOV R1, #eight

            ; select bank 0

            CLR PSW.3

            MOV R4, #1
            CALL Display
            JMP CheckAnswerState

NineTenthSecondsR:

            CJNE R4, #0, CheckAnswerState; D2Changed == 0
            CJNE R6, #0, CheckAnswerState ; counter == 0
            CJNE R1, #1, CheckAnswerState ; timeState == 1

            ; select bank 1
            SETB PSW.3



            MOV R1, #nine

            ; select bank 0

            CLR PSW.3

            MOV R4, #1  ; D2changed = 1
            CALL Display
```

```
        JMP CheckAnswerState



CheckAnswerState:

        CJNE R2, #1, EndCycle1          ; answerState == 1

        CJNE R3, #0, OneSecondAnswer; D1Changed == 0
        MOV A, R5
        CJNE A, #twoSeconds, OneSecondAnswer ; counter == 2 * seconds

        CALL Display
        MOV R5, #0                              ; reset counter

        JMP EndCycle

        EndCycle1:
                JMP EndCycle

        OneSecondAnswer:

                CJNE R3, #0, EndCycle; D1Changed == 0
                CJNE R5, #second, EndCycle ; counter == second

                ; select bank 1
                SETB PSW.3

                CJNE R2, #1, CheckB   ; pressA == 1

                ; select bank 1
                SETB PSW.3
```

```
        MOV A, R0
        MOV R0, #hifenDot
        MOV B, R1
        MOV R1, #aLetter

        CLR PSW.3
        CALL Display

        ; select bank 1
        SETB PSW.3

        MOV R0, A
        MOV R1, B

        ; select bank 0
        CLR PSW.3
        JMP EndCycle

CheckB:

        CJNE R3, #1, CheckC  ; pressB == 1

        ; select bank 1
        SETB PSW.3

        MOV A, R0
        MOV R0, #hifenDot
        MOV B, R1
        MOV R1, #bLetter
```

```
                CLR PSW.3
                CALL Display

                ; select bank 1
                SETB PSW.3

                MOV R0, A
                MOV R1, B

                ; select bank 0
                CLR PSW.3
                JMP EndCycle

CheckC:

                CJNE R4, #1, CheckD  ; pressC == 1

                ; select bank 1
                SETB PSW.3

                MOV A, R0
                MOV R0, #hifenDot
                MOV B, R1
                MOV R1, #cLetter

                CLR PSW.3
                CALL Display

                ; select bank 1
                SETB PSW.3
```

```
                MOV R0, A
                MOV R1, B

                ; select bank 0
                CLR PSW.3
                JMP EndCycle

        CheckD:

                CJNE R5, #1, CheckNone        ; pressD == 1

                ; select bank 1
                SETB PSW.3

                MOV A, R0
                MOV R0, #hifenDot
                MOV B, R1
                MOV R1, #dLetter

                CLR PSW.3
                CALL Display

                ; select bank 1
                SETB PSW.3

                MOV R0, A
                MOV R1, B

                ; select bank 0
                CLR PSW.3
                JMP EndCycle
```

```
            CheckNone:

                ; select bank 1
                SETB PSW.3

                MOV A, R0
                MOV R0, #hifenDot
                MOV B, R1
                MOV R1, #hifen

                CLR PSW.3
                CALL Display

                ; select bank 1
                SETB PSW.3

                MOV R0, A
                MOV R1, B

                ; select bank 0
                CLR PSW.3
                JMP EndCycle

        EndCycle:
            CLR PSW.3
            JMP For                         ; begin again

Initializations:

        MOV R0, #waitingState
```

```
        MOV R1, #timeState
        MOV R2, #answerState
        MOV R3, #D1changed
        MOV R4, #D2changed
        MOV R5, #counter
        MOV R6, #counter1

        ; select bank 1
        SETB PSW.3



        MOV R0, #D1
        MOV R1, #D2
        MOV R2, #pressA
        MOV R3, #pressB
        MOV R4, #pressC
        MOV R5, #pressD

        ; select bank 0
        CLR PSW.3



        RET

EnableInterruptions:
        MOV IE, #8FH                    ; EA=1, ET1=1, EX1=1, ET0=1 e EX0=1 ->
IE=10001111
        MOV IP, #00H                    ; Don't change priorities
        RET

ConfigureInterruptions:
```

```
        ; TMOD Config
        MOV A, TMOD                         ; copy tmod to A
        ANL A,#0x00                         ; set all bits of tmod to 0
        ORL A,#0x11                         ; set timer mods to 1 (16 bits)
        MOV TMOD, A                         ; set tmod

        ; timer 0 config
        MOV TH0, #THtimer           ; set th0
        MOV TL0, #THtimer           ; set tl0

        ; timer 1 config
        MOV TH1, #THtimer           ; set th1
        MOV TL1, #THtimer           ; set tl1

        ; TCON config
        CLR TR0                                 ; timer 0 is turned off
        CLR TR1                                 ; timer 1 is turned off
        SETB IT0                            ; external interruption 0 is activated on
falling edge
        SETB IT1                            ; external interruption 1 is activated on
falling edge

        RET

Display:

        ; select bank 1
        SETB PSW.3



        MOV P1, R0
```

```
        MOV P2, R1

        ; select bank 0

        CLR PSW.3

        RET


ExternalInterruption0:

        PUSH PSW

        ; select bank 0
        CLR PSW.3

        CJNE R0, #1, NotWaiting                 ; skip if the wainting state is 1

        CJNE R2, #0, ExternalInterruption0End; skip if answer state is 0

        ; change states
        MOV R0, #0                                      ; change to non waiting state
        MOV R1, #1                                      ; change to start counting time state

        ; enable timers
        SETB TR0
        SETB TR1

        ; Reload timers
        MOV TH0, #THtimer                       ; set th0
        MOV TL0, #THtimer                       ; set tl0
```

```
MOV TH1, #THtimer                    ; set th1
MOV TL1, #THtimer                    ; set tl1

; reset counters
MOV R5, #0
MOV R6, #0

JMP ExternalInterruption0End

NotWaiting:

        CJNE R2, #1, ExternalInterruption0End ; answerState == 1

        ; change states
        MOV R0, #1                           ; change to waiting state
        MOV R2, #0                           ; change to non answer state

        ; select bank 1
        SETB PSW.3


        MOV R2, #0      ; Reset button press flags
        MOV R3, #0
        MOV R4, #0
        MOV R5, #0

        MOV R0, #fiveDot
        MOV R1, #zero

        ; select bank 0
```

```
                CLR PSW.3

                CALL Display

                ; reset counters
                MOV R5, #0
                MOV R6, #0

                // stop timers
                CLR TR0
                CLR TR1

        ExternalInterruption0End:
                POP PSW
                RETI

TimerInterruption0:

        PUSH PSW

        ; select bank 0
        CLR PSW.3

        ; timer 0 config
        MOV TH0, #THtimer               ; set th0
        MOV TL0, #THtimer               ; set tl0

        INC R5                                  ; Increment counter

        MOV R3, #0                              ; D1changed = 0
```

```
CJNE R5, #fiveSeconds, TimerInterruption0End ; counter == 5*second

CJNE R0, #0, TimerInterruption0End ; waitingState == 0

CJNE R1, #1, TimerInterruption0End ; timeState == 1

; reset counter
MOV R5, #0                          ; counter = 0

; change states
MOV R1, #0                          ; timeState = 0
MOV R2, #1                          ; answerState = 1

; display
; select bank 1
SETB PSW.3


MOV R0, #zeroDot
MOV R1, #zero

; select bank 0

CLR PSW.3

CALL Display

TimerInterruption0End:
        POP PSW
        RETI
```

```
ExternalInterruption1:

        PUSH PSW

        ; select bank 0
        CLR PSW.3

        CJNE R0, #0, ExternalInterruption1End ; waitingState == 0

        CJNE R2, #0, ExternalInterruption1End ; answerState == 0

        JB BA, AnswerB                        ; BA == 0

        ; select bank 1
        SETB PSW.3


        MOV R2, #1

        ; select bank 0

        CLR PSW.3

        ; change states
        MOV R1, #0                            ; timeState = 0
        MOV R2, #1                            ; answerState = 1

        JMP ExternalInterruption1End

        AnswerB:
                JB BB, AnswerC                ; BB == 0
```

```
        ; select bank 1
        SETB PSW.3


        MOV R3, #1

        ; select bank 0

        CLR PSW.3

        ; change states
        MOV R1, #0                      ; timeState = 0
        MOV R2, #1                      ; answerState = 1

        JMP ExternalInterruption1End

AnswerC:
        JB BC, AnswerD                  ; BC == 0

        ; select bank 1
        SETB PSW.3


        MOV R4, #1

        ; select bank 0
        CLR PSW.3

        ; change states
        MOV R1, #0                      ; timeState = 0
```

```
                MOV R2, #1                                  ; answerState = 1

                JMP ExternalInterruption1End
        AnswerD:
                JB BD, ExternalInterruption1End; BD == 0

                ; select bank 1
                SETB PSW.3


                MOV R5, #1

                ; select bank 0
                CLR PSW.3

                ; change states
                MOV R1, #0                                  ; timeState = 0
                MOV R2, #1                                  ; answerState = 1

                JMP ExternalInterruption1End

        ExternalInterruption1End:
                POP PSW
                RETI

TimerInterruption1:

        PUSH PSW

        ; select bank 0
        CLR PSW.3
```

```
        ; timer 1 config
        MOV TH1, #THtimer                   ; set th1
        MOV TL1, #THtimer                   ; set tl1

        INC R6                                      ; counter1++

        MOV R4, #0                                  ; D2changed = 0

        CJNE R6, #second, TimerInterruption1End ; counter1 == second

        MOV R6, #0                                  ; counter1 = 0

        TimerInterruption1End:
                POP PSW
                RETI

END
```

Anexo B2: Código em Linguagem C

```
#include <reg51.h>

#define tenthSecond                 2 //time to count 0,1 seconds (2 x
50ms)
#define second                          20 // time to count 1 second (20
x 50ms)

// values for display
#define null                                0xFF
#define zeroDot                         0x40
#define oneDot                          0x79
#define twoDot                          0x24
#define threeDot                        0x30
#define fourDot                         0x19
#define fiveDot                         0x12
#define hifenDot                        0x3F
#define zero                                0xC0
#define one                                 0xF9
#define two                                 0xA4
#define three                               0xB0
#define four                                0x99
#define five                                0x92
#define six                                 0x82
#define seven                               0xF8
#define eight                               0x80
#define nine                                0x90
#define hifen                               0xBF
#define aLetter                         0x88
#define bLetter                         0x83
#define cLetter                         0xC6
#define dLetter                         0xA1


// Buttons definition
sbit BA = P3^4;
sbit BB = P3^5;
sbit BC = P3^6;
sbit BD = P3^7;

bit pressA = 0;
bit pressB = 0;
bit pressC = 0;
bit pressD = 0;
bit waitingState = 1;
bit timeState = 0;
bit answerState = 0;
bit D1changed = 0;
```

```
bit D2changed = 0;

unsigned int counter = 0;
unsigned int counter1 = 0;

unsigned char D1;
unsigned char D2;


//Function declarations
void Init(void);
void display(unsigned char Display1, unsigned char Display2);

void main (void)
{
    Init();

    while(1) {

            if (waitingState == 0) {

                // button has been pressed to start counting

                if (timeState == 1) {

                    if (counter == (4*second) && D1changed == 0) {

                        D1 = zeroDot;
                        D1changed = 1;

                    } else if (counter == (3*second) && D1changed == 0) {

                        D1 = oneDot;
                        D1changed = 1;

                    } else if (counter == (2*second) && D1changed == 0) {

                        D1 = twoDot;
                        D1changed = 1;

                    } else if (counter == second && D1changed == 0) {

                        D1 = threeDot;
                        D1changed = 1;

                    } else if (counter == 0 && D1changed == 0 &&
timeState == 1) {

                        D1 = fourDot;
```

```
                        D1changed = 1;

                    }

                    if (counter1 == (9*tenthSecond) && D2changed == 0) {

                        D2 = zero;
                        D2changed = 1;

                    } else if (counter1 == (8*tenthSecond) && D2changed
== 0) {

                        D2 = one;
                        D2changed = 1;

                    } else if (counter1 == (7*tenthSecond) && D2changed
== 0) {

                        D2 = two;
                        D2changed = 1;

                    } else if (counter1 == (6*tenthSecond) && D2changed
== 0) {

                        D2 = three;
                        D2changed = 1;

                    } else if (counter1 == (5*tenthSecond) && D2changed
== 0) {

                        D2 = four;
                        D2changed = 1;

                    } else if (counter1 == (4*tenthSecond) && D2changed
== 0) {

                        D2 = five;
                        D2changed = 1;

                    } else if (counter1 == (3*tenthSecond) && D2changed
== 0) {

                        D2 = six;
                        D2changed = 1;

                    } else if (counter1 == (2*tenthSecond) && D2changed
== 0) {

                        D2 = seven;
```

```
                        D2changed = 1;

                    } else if (counter1 == tenthSecond && D2changed == 0)
{

                        D2 = eight;
                        D2changed = 1;

                    } else if (counter1 == 0 && D2changed == 0 &&
timeState == 1) {

                        D2 = nine;
                        D2changed = 1;

                    }

                    if (D2changed == 1 || D2changed == 1) {

                        display(D1, D2);

                    }

                }

                // answer state
                if (answerState == 1) {

                    if (counter == (2*second) && D2changed == 0) {

                            display(D1, D2);

                            counter = 0; // repeat the cycle


                    } else if (counter == second && D2changed == 0) {

                        if (pressA == 1) {

                            display(hifenDot, aLetter);

                        } else if (pressB == 1) {

                            display(hifenDot, bLetter);

                        } else if (pressC == 1) {

                            display(hifenDot, cLetter);

                        } else if (pressD == 1) {
```

```
                            display(hifenDot, dLetter);
                        } else {
                            display(hifenDot, hifen);
                        }
                    }
                }
            }
        }
}
void Init(void){

    //Configuration of interruptions
    EA = 1; //ativate global interruptions
    ET0 = 1; // activate timer interruption 0
    ET1 = 1; // activate timer interruption 1
    EX0 = 1; // activate external interruption 0
    EX1 = 1, // activate external interruption 1

    //Configure TMOD
    TMOD &= 0x00; //Clean initial 4 bits of timer 0 and 1
    TMOD |= 0x11; //Set timer 0 and 1 with 16 bits

    //Configure timer 0
    //Timer 0 - 50ms -> (65536(10000h) - 50000(3E8h) = 15536(3CB0h))
    TH0 = 0x3C;
    TL0 = 0xB0;
    //Timer 1 - 50ms -> (65536(10000h) - 50000(3E8h) = 15536(3CB0h))
    TH1 = 0x3C;
    TL1 = 0xB0;


    //Configure TCON
    IT0 = 1; //specify falling edge trigger on external interruption 0
    IT1 = 1; //specify falling edge trigger on external interruption 1

    display(fiveDot, zero);
}

void Timer0_ISR (void) interrupt 1 {

    //Configure timer 0
```

```
    //Timer 0 - 50ms -> (65536(10000h) - 50000(C350h)= 15536(3CB0h))
    TH0 = 0x3C;
    TL0 = 0xB0;

    counter++;

    if (counter % second == 0) {

        D1changed = 0;

    }



    // start showing answer state with no answer
    if(counter >= (5*second) && waitingState == 0 && timeState == 1){

        // reset timer
        counter = 0;

        // change stats
        timeState = 0;
        answerState = 1;

        // bug fix
        D1 = zeroDot;
        D2 = zero;

    }

}

void Timer1_ISR (void) interrupt 3 {

    //Timer 1 - 50ms -> (65536(10000h) - 50000(3E8h) = 15536(3CB0h))
    TH1 = 0x3C;
    TL1 = 0xB0;

    // increase counter
    counter1++;

    if (counter1 % tenthSecond == 0) {

        D2changed = 0;

    }


    if (counter1 >= second) {
```

```
        counter1 = 0;

    }


}

void External0_ISR (void) interrupt 0 {

    // change for non waiting state and start counting timer
    if (waitingState == 1 && answerState == 0) {

        waitingState = 0;
        timeState = 1;

        TR0 = 1;
        TR1 = 1;

        //Timer 0 - 50ms -> (65536(10000h) - 50000(3E8h) = 15536(3CB0h))
        TH0 = 0x3C;
        TL0 = 0xB0;
        //Timer 1 - 50ms -> (65536(10000h) - 50000(3E8h) = 15536(3CB0h))
        TH1 = 0x3C;
        TL1 = 0xB0;

        counter = 0;
        counter1 = 0;

    }

    // beginning state with 5.0s
    if (answerState == 1) {

        answerState = 0;
        waitingState = 1;

        pressA = 0;
        pressB = 0;
        pressC = 0;
        pressD = 0;

        counter = 0;
        counter1 =0;

        // bug fix
        D1 = fiveDot;
        D2 = zero;
```

```
        display(D1, D2);

        // stop timers
        TR0 = 0;
        TR1 = 0;

    }


}

void External1_ISR (void) interrupt 2 {

    // check which button has been pressed
    if (waitingState == 0 && answerState == 0) {

        if (~BA) {

            pressA = 1;

            // change states
            timeState = 0;
            answerState = 1;

        } else if (~BB) {

            pressB = 1;

            // change states
            timeState = 0;
            answerState = 1;

        } else if (~BC) {

            pressC = 1;

            // change states
            timeState = 0;
            answerState = 1;

        } else if (~BD) {

            pressD = 1;

            // change states
            timeState = 0;
            answerState = 1;
        }
```

```
        counter = 0;
        counter1 = 0;

    }


}

void display(unsigned char Display1, unsigned char Display2)
{

    P1 = Display1;
    P2 = Display2;

}
```