```
; Harsh Savla & TJ Wiegman
; ME 58600
; 2022-09-26
; initint.s

; a subroutine to enable the external IRQ interrupt pin
RCC_APB2ENR     EQU 0x40021018
GPIOA_CRL       EQU 0x40010800
EXTI_IMR        EQU 0x40010400
EXTI_FTSR       EQU 0x4001040C
NVIC_ISER0      EQU 0xE000E100

; program code
        AREA ARMex, CODE, READONLY
        ENTRY
initint PROC
        EXPORT initint
        ; push LR to stack
        push {LR}

        ; Enable GPIO A
        ldr R3, =RCC_APB2ENR
        ldrb R1, [R3]
        orr R1, 0x04  ; enable bit 2 (0100)
        strb R1, [R3]

        ; Configure PA01
        ldr R3, =GPIOA_CRL
        ldrb R1, [R3]
        and R1, #0x0F ; set PA01 to 0000
        orr R1, #0x40 ; set PA01 to 0100
        strb R1, [R3]

        ; Unmask External Interrupt 1
        ldr R3, =EXTI_IMR
        mov R1, #0x02
        str R1, [R3]

        ; Set trigger to falling edge
        ldr R3, =EXTI_FTSR
        mov R1, #0x02
        str R1, [R3]

        ; Enable in NVIC (?)
        ldr R3, =NVIC_ISER0
        mov R1, #0x80
        str R1, [R3]

        ; End subroutine and go back to caller
        pop {LR}
        bx LR
        ENDP
    END
```

```
; Harsh Savla & TJ Wiegman
; ME 58600
; 2022-09-26
; inittime.s

; a subroutine to set up the SysTick timer interrupt for every 100ms
STK_CTRL    EQU 0xE000E010
STK_LOAD    EQU 0xE000E014

; program code
        AREA ARMex, CODE, READONLY
        ENTRY
inittime PROC
        EXPORT inittime
        ; push LR to stack
        push {LR}

        ; Set timer to 100 ms
        ldr R3, =STK_LOAD
        ldr R1, =0x249EFF
        str R1, [R3]

        ; Enable timer, systick interrupt, and 24MHz counter
        ldr R3, =STK_CTRL
        mov R1, #0x07
        str R1, [R3]

        ; End subroutine and go back to caller
        pop {LR}
        bx LR
        ENDP
    END
```

```
; Harsh Savla & TJ Wiegman
; ME 58600
; 2022-09-19
; serialIO.s

RCC_APB2ENR EQU 0x40021018 ; enable APB2 clock for USART1
GPIOA_CRH   EQU 0x40010804 ; configure PA09 and PA10 for tx/rx
USART1_BRR  EQU 0x40013808 ; configure USART1 baud rate
USART1_CR1  EQU 0x4001380C ; enable USART1, set parity, mode
USART1_SR   EQU 0x40013800 ; USART1 satus register
USART1_DR   EQU 0x40013804 ; USART1 data register

USART1baud  EQU 0x00D0 ; hex fraction for setting baud rate

; allocate some RAM for bindec
        AREA MyData, DATA, READWRITE
num3    SPACE 2
array3  SPACE 6

; program code
        AREA ARMex, CODE, READONLY
        ENTRY
initcom     PROC    ; initializes serial channel 1 for asynchronous communcations
        EXPORT initcom
        ; push LR to stack
        push {LR}

        ; turn on APB2 perhipheral clock
        ldr R3, =RCC_APB2ENR
        ldr R1, [R3] ; save current APB2 state
        orr R1, #0x4000
                orr R1, #0x0005
        str R1, [R3]

        ; configure port A - PA09 output, PA10 input
        ldr R3, =GPIOA_CRH
        ldr R1, =0x444444B4
        str R1, [R3]

        ; configure USART1 baud rate
        ldr R3, =USART1_BRR
        mov R1, #USART1baud ; as close as possible to 115,200 -- impossible to get exact
at 24MHz
        str R1, [R3]

        ; enable USART1 for 8 data bits. disable parity and interrupts
        ldr R3, =USART1_CR1
        mov R1, #0x200C
        str R1, [R3]

        ; End subroutine and go back to caller
        pop {LR}
        bx LR
        ENDP

```

```
55  checkcom    PROC    ; checks to see if a character is in the data receive register. writes
    0xFF to R0 if availabe, 0x00 otherwise
56          EXPORT checkcom
57          ; push LR to stack
58          push {LR}
59
60          ; check status register
61          ldr R3, =USART1_SR
62          ldr R1, [R3]
63          and R1, #32 ; mask out unneeded flags
64          cmp R1, #32 ; check if RXNE flag is set
65          beq ready
66
67          ; set R0 to 0x00 if not ready
68          mov R0, #0x00
69          b chEnd
70
71          ; set R0 to 0xFF if ready
72  ready   mov R0, #0xFF
73
74          ; End subroutine and go back to caller
75  chEnd   pop {LR}
76          bx LR
77          ENDP
78
79  getchar     PROC    ; fetches character from serial channel 1 and writes it as ASCII to R0
80          EXPORT getchar
81          ; push LR to stack
82          push {LR}
83
84          ldr R3, =USART1_DR
85          ldrb R0, [R3]
86
87          ; End subroutine and go back to caller
88          pop {LR}
89          bx LR
90          ENDP
91
92  showchar    PROC    ; checks that TXE is set, then outputs ASCII character in R0 to serial
    channel 1
93          EXPORT showchar
94          ; push LR to stack
95          push {LR}
96
97          ; check status register
98  shWait  ldr R3, =USART1_SR
99          ldr R1, [R3]
100         and R1, #128 ; mask out unneeded flags
101         cmp R1, #128 ; check if TXE flag is set
102         beq write
103
104         ; if DR is not ready yet
105         b shWait
106
107         ; if DR is ready
```

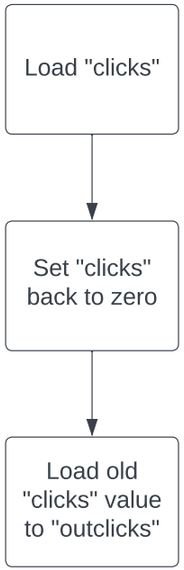
```
write   ldr R3, =USART1_DR
        strb R0, [R3]

        ; End subroutine and go back to caller
                pop {LR}
        bx LR
        ENDP

bindec      PROC ; converts a 16-bit signed binary number into five decimal characters (digits)
; preceded by either a space or a minus sign depending on whether the signed number is positive or negative
        EXPORT bindec
        ; push LR to stack
        push {LR}

        ; fill array with spaces
        mov R1, #1
        ldr R3, =array3
        mov R0, #0x20   ; ascii character for " "
clrloop strb R0, [R3], #1
        add R1, #1
        cmp R1, #7
        bne clrloop

        ; get input number from RAM, put into R0
        ldr R3, =num3
        ldrh R0, [R3]

        mov R1, R0
        lsr R1, #15 ; shift first digit down to LSB
        cmp R1, #1  ; is negative?
        beq neg1

        ; is positive
        mov R1, #0x20 ; ascii character for " "
        ldr R3, =array3
        strb R1, [R3], #5
        b binloop

        ; is negative
neg1    mov R1, #0x2D ; ascii character for "-"
        ldr R3, =array3
        strb R1, [R3], #5
        sxth R0
        sub R0, #1
        eor R0, #0xFFFFFFFF

        ; divide by 10
binloop mov R4, #10
                udiv R1, R0, R4  ; R1 holds quotient
        mul R2, R1, R4
        sub R2, R0, R2      ; R2 holds remainder

        ; convert to ASCII and store
```

```
        add R2, #0x30
        strb R2, [R3], #-1
        mov R0, R1
        cmp R0, #0
        bne binloop

        ; End subroutine and go back to caller
        pop {LR}
        bx LR
        ENDP

shownum     PROC ; takes binary half-word from R0 and outputs decimal over serial
        EXPORT shownum
        ; push LR to stack
        push {LR}

        ; Load R0 half-word to RAM for bindec to use it
        ldr R3, =num3
        strh R0, [R3]
        bl bindec

        ; Loop through decimal characters until all printed
        mov R2, #0
sloop   ldr R3, =array3
        add R3, R2
        ldrb R0, [R3]
        bl showchar
        add R2, #1
        cmp R2, #6
        bne sloop

                ; Write newline afterwards, forces buffer empty
                mov R0, #0x0A
                bl showchar
                mov R0, #0x0D
                bl showchar

        ; End subroutine and go back to caller
        pop {LR}
        bx LR
        ENDP
    END

```

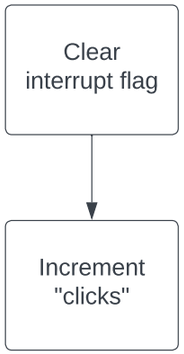
```
; Harsh Savla & TJ Wiegman
; ME 58600
; 2022-09-26
; lab4main.s

EXTI_PR         EQU 0x40010414

; allocate some RAM for clicks and outclicks
        AREA MyData, DATA, READWRITE
clicks      SPACE 2
outclicks   SPACE 2

; program code
        AREA ARMex, CODE, READONLY
        ENTRY
__main PROC
        EXPORT __main
        IMPORT initcom
        IMPORT initint
        IMPORT inittime
        IMPORT shownum
        IMPORT checkcom
        IMPORT getchar

        ; Initialize serial communications
        bl initcom

        ; Initialize clicks and outclicks to zero
        mov R0, #0
        ldr R3, =clicks
        strh R0, [R3]
        ldr R3, =outclicks
        strh R0, [R3]

        ; Set up external clock interrupt
        bl initint

        ; Set up 100ms timer
        bl inittime

        ; Main loop
chloop  ldr R3, =outclicks
        ldrh R1, [R3]
        cmp R1, #0x0000
        bne shclick

        ; Check if received serial comm
        bl checkcom
        cmp R0, #0xFF
        bne chloop

        ; Check if received character is ESC
        bl getchar
        cmp R0, #0x1B   ; ASCII value for ESCAPE
```

```
        beq done
        b chloop

shclick mov R0, R1 ; copy outclicks from R1 to R0
        mov R1, #0
        strh R1, [R3] ; reset outclicks to zero
        bl shownum ; R0 still holds old outclicks value
        b chloop

done    b done
        ENDP

EXTI1_IRQHandler    PROC
        EXPORT EXTI1_IRQHandler
        ; push LR to stack
        push {LR}

        ; Clear interrupt flag
        ldr R3, =EXTI_PR
        mov R1, #0x02
        str R1, [R3]

        ; Increase clicks value
        ldr R3, =clicks
        ldrh R1, [R3]
        add R1, #1
        strh R1, [R3]

        ; End subroutine and go back to caller
        pop {LR}
        bx LR
        ENDP

SysTick_Handler     PROC
        EXPORT SysTick_Handler
        ; push LR to stack
        push {LR}

        ; Get clicks, then reset
        ldr R3, =clicks
        ldrh R1, [R3]
        mov R2, #0
        strh R2, [R3]

        ; Move clicks to outclicks
        ldr R3, =outclicks
        strh R1, [R3]

        ; End subroutine and go back to caller
        pop {LR}
        bx LR
        ENDP
    END
```

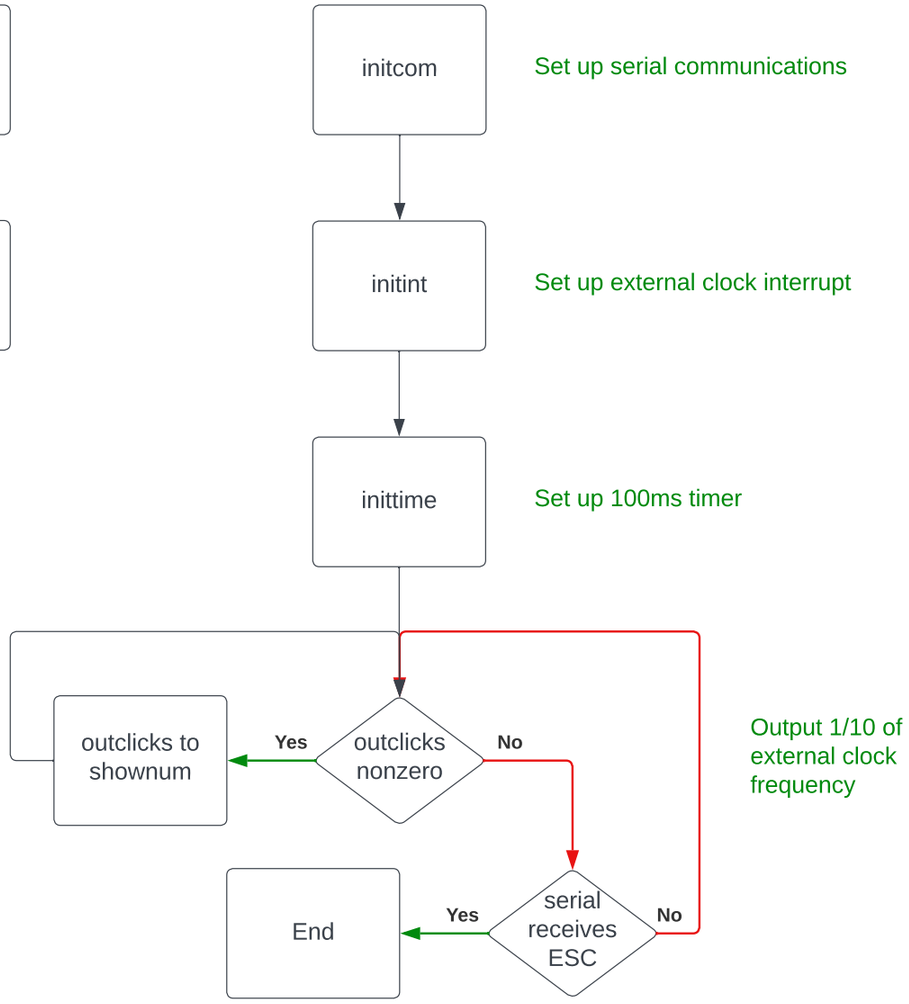
SysTick_Handler



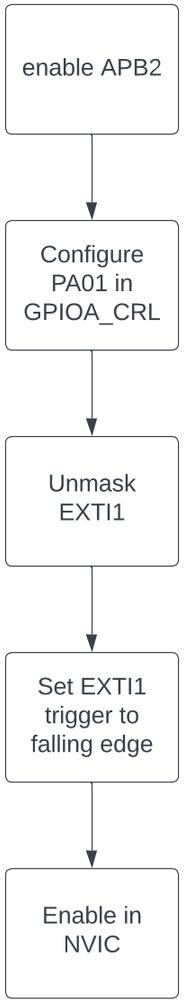
EXTI1_IRQHandler



lab4main.s



initint.s



initint.s