Zeynepnur ŞAHİNEL
2305399
Wed, 12:40 16.01.2021

EE447 EXP5 PRELIMINARY WORK

Part 1)

In this question initially, ADC and PORT E3 is initialized. Since we use ADC0, PORT E3 is configured. Both initializations are made in the same subroutine called ATD_PORT_INIT which is shown in the Fig. 1. In the Program_Directive.s, main code is written. In the main code, initially sampling is started by enabling Sequencer 3 in ADC0_PSSI. After setting proper bit, sample completenes is checked through polling. Until the flag bit in the ADC0_RIS is set to 1, polling continues. Having 1 in the flag bit means that sequence is finished. After branch fails, data in the ADC0_SSFIFO3 is read to R0. FIFO register stores result address. Then, flag is cleared through setting ADC0_ISC. Moreover, it is important to mention that potentiometer output pin is connected to the PE3 pin. Through potentiometer and PE3, input data is introduced. Changing potentiometer yields different R0 values from 0x000 to 0xFFF, which can be seen from Fig. 3. Eventhough, it is not asked to use Termit, for the sake of completeness OutStr and Convrt codes are impelemented and the resulted is printed. (Fig. 4)

```
; ADC Registers
RCGCADC             EQU 0x400FE638 ; ADC clock register
; ADC0 base address  EQU 0x40038000
ADC0_ACTSS          EQU 0x40038000 ; Sample sequencer (ADC0 base address)
ADC0_RIS            EQU 0x40038004 ; Interrupt status
ADC0_IM             EQU 0x40038008 ; Interrupt select
ADC0_EMUX           EQU 0x40038014 ; Trigger select
ADC0_PSSI           EQU 0x40038028 ; Initiate sample
ADC0_SSMUX3         EQU 0x400380A0 ; Input channel select
ADC0_SSCTL3         EQU 0x400380A4 ; Sample sequence control
ADC0_SSFIFO3        EQU 0x400380A8 ; Channel 3 results
ADC0_PC             EQU 0x40038FC4 ; Sample rate
ADC0_ISC            EQU 0x4003800C ; Interrupt Status and Clear Register

; GPIO Registers
RCGCGPIO            EQU 0x400FE608 ; GPIO clock register
;PORT E base address EQU 0x40024000
PORTE_DIR           EQU 0x40024400 ; Direction Register
PORTE_DEN           EQU 0x4002451C ; Digital Enable
PORTE_PCTL          EQU 0x4002452C ; Alternate function select
PORTE_AFSEL         EQU 0x40024420 ; Enable Alt functions
PORTE_AMSEL         EQU 0x40024528 ; Enable analog


                                AREA atd_port_init, CODE, READONLY
```

```
                    THUMB
                    EXPORT ATD_PORT_INIT

ATD_PORT_INIT PROC

    ; Start clocks for features to be used
    LDR R1, =RCGCADC ; Turn on ADC clock
    LDR R0, [R1]
    ORR R0, R0, #0x01 ; set bit 0 to enable ADC0 clock
    STR R0, [R1]
    NOP
    NOP
    NOP ; Let clock stabilize

    LDR R1, =RCGCGPIO ; Turn on GPIO clock
    LDR R0, [R1]
    ORR R0, R0, #0x10 ; set bit 4 to enable port E clock
    STR R0, [R1]
    NOP
    NOP
    NOP ; Let clock stabilize

    ; Setup GPIO to make PE3 input for ADC0
    ; Enable alternate functions
    LDR R1, =PORTE_AFSEL
    LDR R0, [R1]
    ORR R0, R0, #0x08 ; set bit 3 to enable alt functions on PE3
    STR R0, [R1]

    ; PCTL does not have to be configured
    ; since ADC0 is automatically selected when
    ; port pin is set to analog.
    ; Disable digital on PE3
    LDR R1, =PORTE_DEN
    LDR R0, [R1]
    BIC R0, R0, #0x08 ; clear bit 3 to disable digital on PE3
    STR R0, [R1]

    ;Set direction of DE3
    LDR R1, =PORTE_DIR
    LDR R0, [R1]
    BIC R0, R0, #0x08 ; clear bit 3 input
    STR R0, [R1]

    ; Enable analog on PE3
    LDR R1, =PORTE_AMSEL
    LDR R0, [R1]
    ORR R0, R0, #0x08 ; set bit 3 to enable analog on PE3
    STR R0, [R1]
```

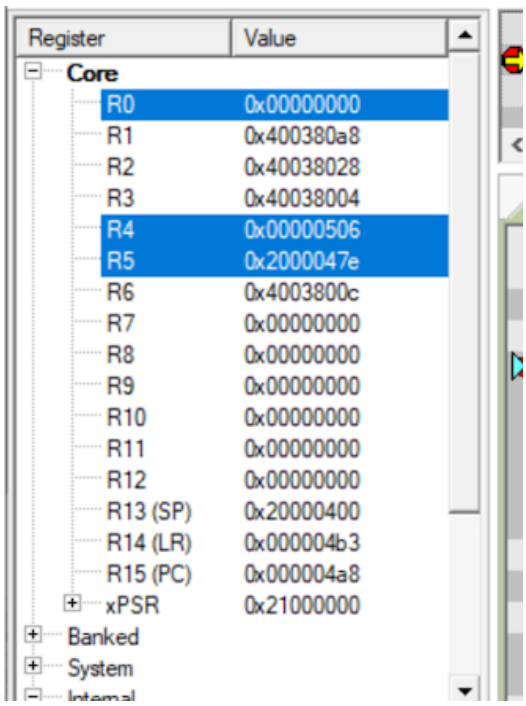
```
        ; Disable sequencer while ADC setup
        LDR R1, =ADC0_ACTSS
        LDR R0, [R1]
        BIC R0, R0, #0x08 ; clear bit 3 to disable seq 3
        STR R0, [R1]

        ; Select trigger source
        LDR R1, =ADC0_EMUX
        LDR R0, [R1]
        BIC R0, R0, #0xF000 ; clear bits 15:12 to select SOFTWARE
        STR R0, [R1] ; trigger

        ; Select input channel
        LDR R1, =ADC0_SSMUX3
        LDR R0, [R1]
        BIC R0, R0, #0x000F ; clear bits 3:0 to select AIN0
        STR R0, [R1]

        ; Config sample sequence
        LDR R1, =ADC0_SSCTL3
        LDR R0, [R1]
        ORR R0, R0, #0x06 ; set bits 2:1 (IE0, END0) IE0 is set since we want RIS to be
set
        STR R0, [R1]

        ; Set sample rate
        LDR R1, =ADC0_PC
        LDR R0, [R1]
        ORR R0, R0, #0x01 ; set bits 3:0 to 1 for 125k sps
        STR R0, [R1]

        ; Done with setup, enable sequencer
        LDR R1, =ADC0_ACTSS
        LDR R0, [R1]
        ORR R0, R0, #0x08 ; set bit 3 to enable seq 3
        STR R0, [R1] ; sampling enabled but not initiated yet

        ;Disable Interrupt
        LDR R1, =ADC0_IM
        LDR R0, [R1]
        BIC R0, R0, #0x08 ; disable interrupt
        STR R0, [R1]

        BX LR
        ENDP
                ALIGN
                END
```

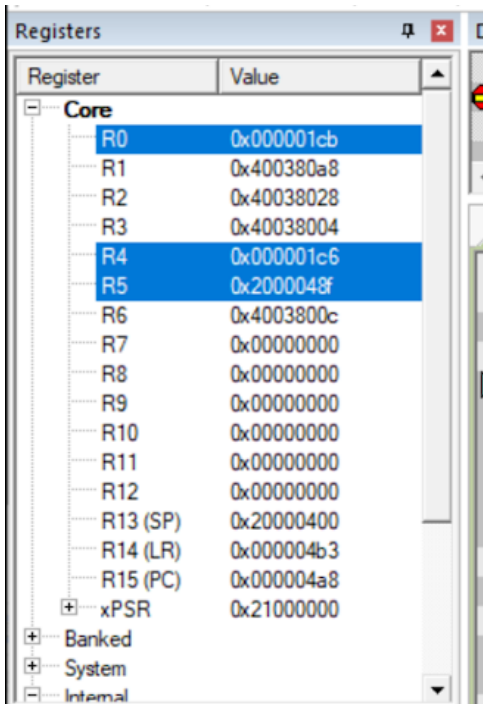
Figure 1. ADC and Port Init Code

```
;*********************
; Program_Directives.s

;*********************
ADC0_RIS EQU 0x40038004 ; Interrupt status
ADC0_PSSI EQU 0x40038028 ; Initiate sample
ADC0_SSFIFO3 EQU 0x400380A8 ; Channel 3 results
ADC0_ISC    EQU   0x4003800C

;*********************
; Program section
;*********************
;LABEL              DIRECTIVE  VALUE                COMMENT
                    AREA       main, READONLY, CODE
                    THUMB
                    EXTERN          ATD_PORT_INIT
                    EXTERN          OutStr
                    EXTERN          Convrt
                    EXPORT     __main; Make available

__main      PROC

                    BL ATD_PORT_INIT
start
                    ; start sampling routine
                    LDR R3, =ADC0_RIS ; interrupt address
                    LDR R1, =ADC0_SSFIFO3 ; result address
                    LDR R2, =ADC0_PSSI ; sample sequence initiate address
                    LDR R6,= ADC0_ISC ;interrupt status clear reg


                    ; initiate sampling by enabling sequencer 3 in ADC0_PSSI
Sample              LDR R0, [R2]
                    ORR R0, R0, #0x08 ; set bit 3 for SS3
                    STR R0, [R2]

                    ; check for sample complete (bit 3 of ADC0_RIS set)
Cont                LDR R0, [R3]
                    ANDS R0, R0, #8
                    BEQ Cont ;if flag is 0

                    ;branch fails if the flag is set so data can be read and flag is cleared
                    LDR R0,[R1]
                    ;STR R0,[R5],#4 ;store the data
                    MOV R4,R0
                    BL Convrt
                    BL OutStr

                    MOV R0, #8
```

```
                        STR R0, [R6] ; clear flag
                        B Sample

                        B exit

exit            NOP
                        B start

                        ENDP
;*********************
; End of the program  section
;*********************
;LABEL     DIRECTIVE     VALUE                    COMMENT
                        ALIGN
                        END
```

Figure 2. Main Code for Part 1





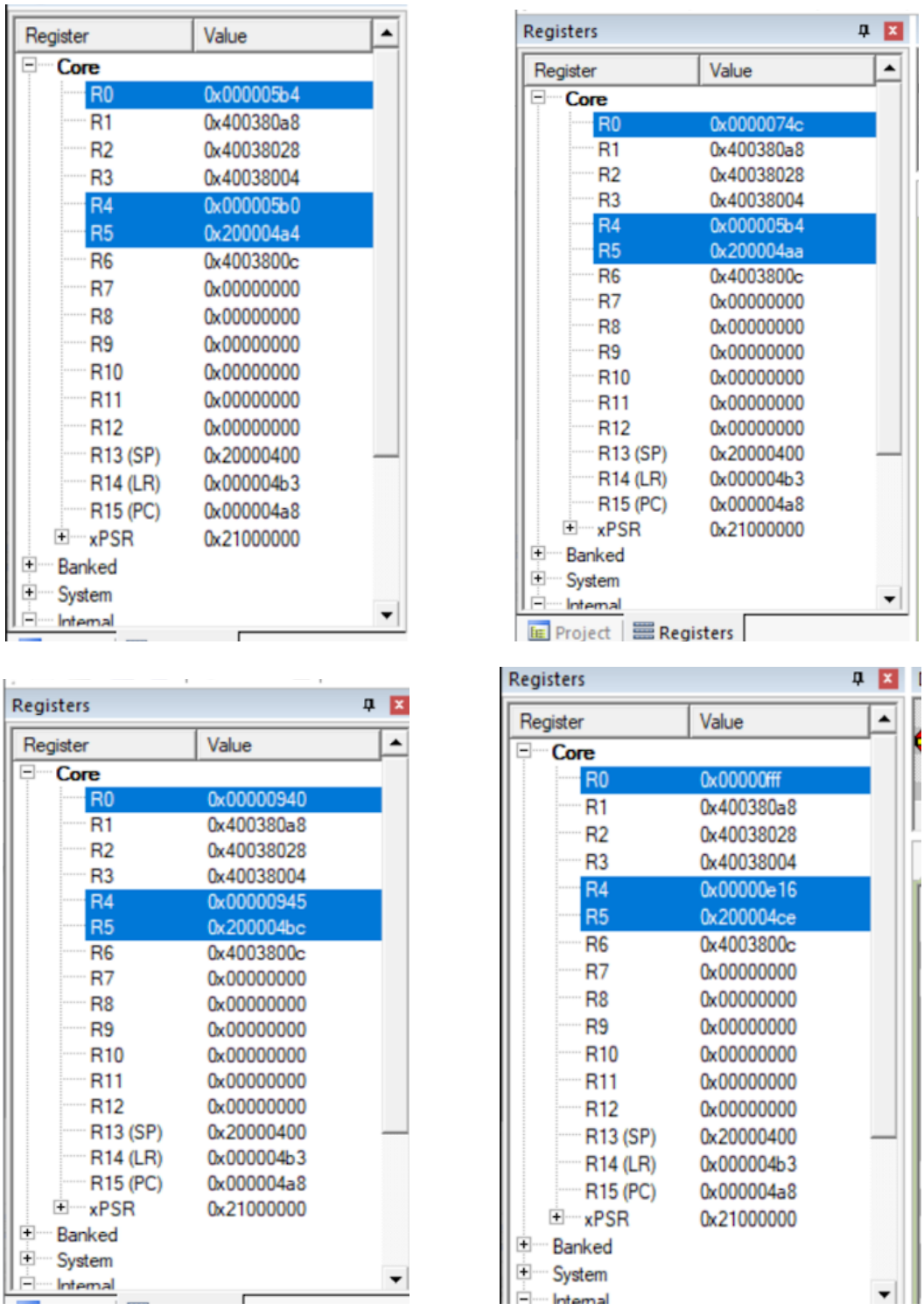


Figure 3. Register values, where R0 is changed from 0x000 to 0xFFF

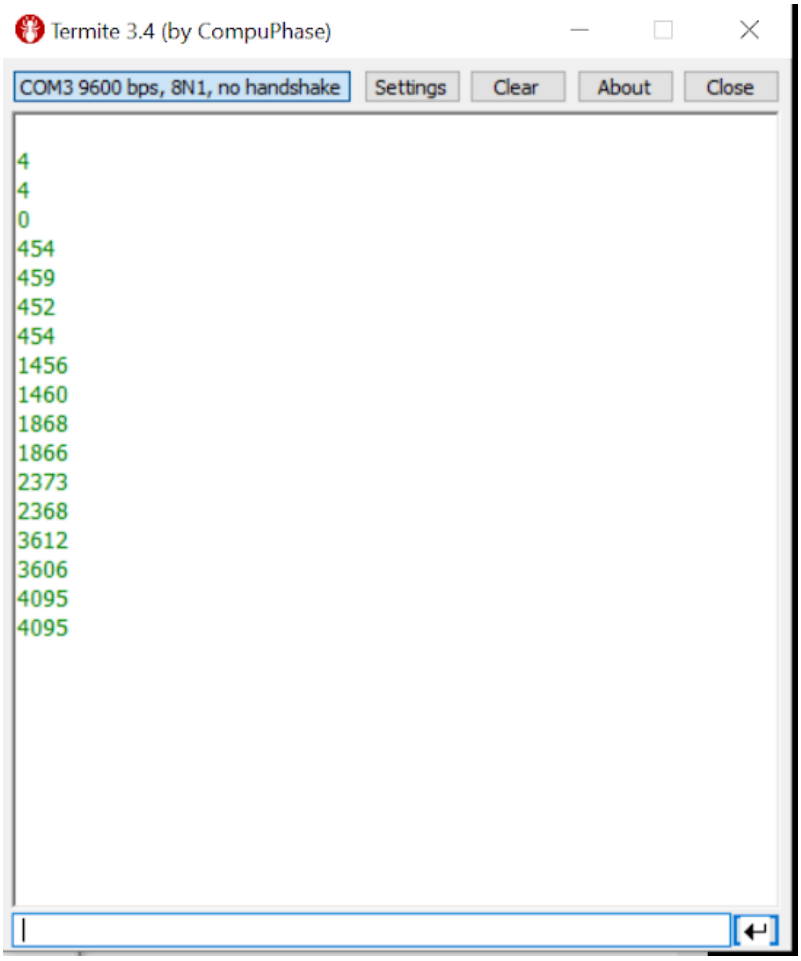


Figure 4. Termit Output

Part 2)

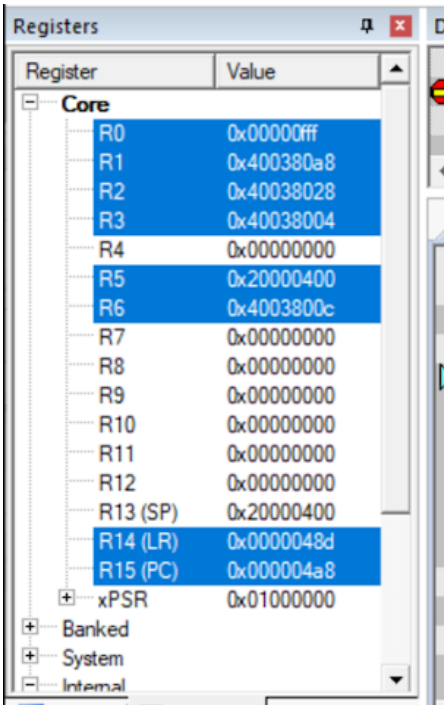
In this part, change is made just in the main code. Here after reading FIFO register, 2048 is substracted from this value. Making this substraction makes the max value 2047 which is equivalent to 1.65 value. However values smaller than 2047 yields negative values which will be handled in Part 3 later. Figure 5 shows the main code whereas Figure 6 shows the vaiation of R4 and Figure 7 shows the Termit Output.

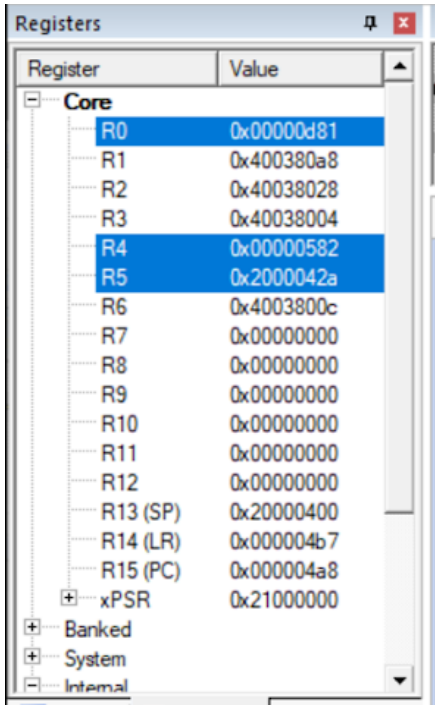
```
;*********************
; Program_Directives.s
;*********************
ADC0_RIS EQU 0x40038004 ; Interrupt status
ADC0_PSSI EQU 0x40038028 ; Initiate sample
ADC0_SSFIFO3 EQU 0x400380A8 ; Channel 3 results
ADC0_ISC    EQU   0x4003800C
MY_ADDR EQU 0x20000400
;*********************
```

```
; Program section
;**********************
;LABEL              DIRECTIVE  VALUE                COMMENT
                   AREA       main, READONLY, CODE
                   THUMB
                   EXTERN          ATD_PORT_INIT
                   EXTERN          OutStr
                   EXTERN          Convrt
                   EXPORT     __main; Make available

__main        PROC
                   BL ATD_PORT_INIT
start
                   ; start sampling routine
                   LDR R3, =ADC0_RIS ; interrupt address
                   LDR R1, =ADC0_SSFIFO3 ; result address
                   LDR R2, =ADC0_PSSI ; sample sequence initiate address
                   LDR R6,= ADC0_ISC ;interrupt status clear reg
                   LDR R5,=MY_ADDR

                   ; initiate sampling by enabling sequencer 3 in ADC0_PSSI
Sample             LDR R0, [R2]
                   ORR R0, R0, #0x08 ; set bit 3 for SS3
                   STR R0, [R2]

                   ; check for sample complete (bit 3 of ADC0_RIS set)
Cont               LDR R0, [R3]
                   ANDS R0, R0, #8
                   BEQ Cont ;if flag is 0

                   ;branch fails if the flag is set so data can be read and flag is cleared
                   LDR R0,[R1]
                   ;STR R0,[R5],#4 ;store the data

                   SUB R0, #0x800 ;substract 2048

                   MOV R4,R0
                   BL Convrt
                   BL OutStr
                   MOV R0, #8
                   STR R0, [R6] ; clear flag
                   B Sample

                   B exit

exit               NOP
                   B start

                   ENDP
```

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

; End of the program  section
;**********************
;LABEL     DIRECTIVE      VALUE                    COMMENT
                         ALIGN
                         END
```

Figure 5. Main Code of Part 2





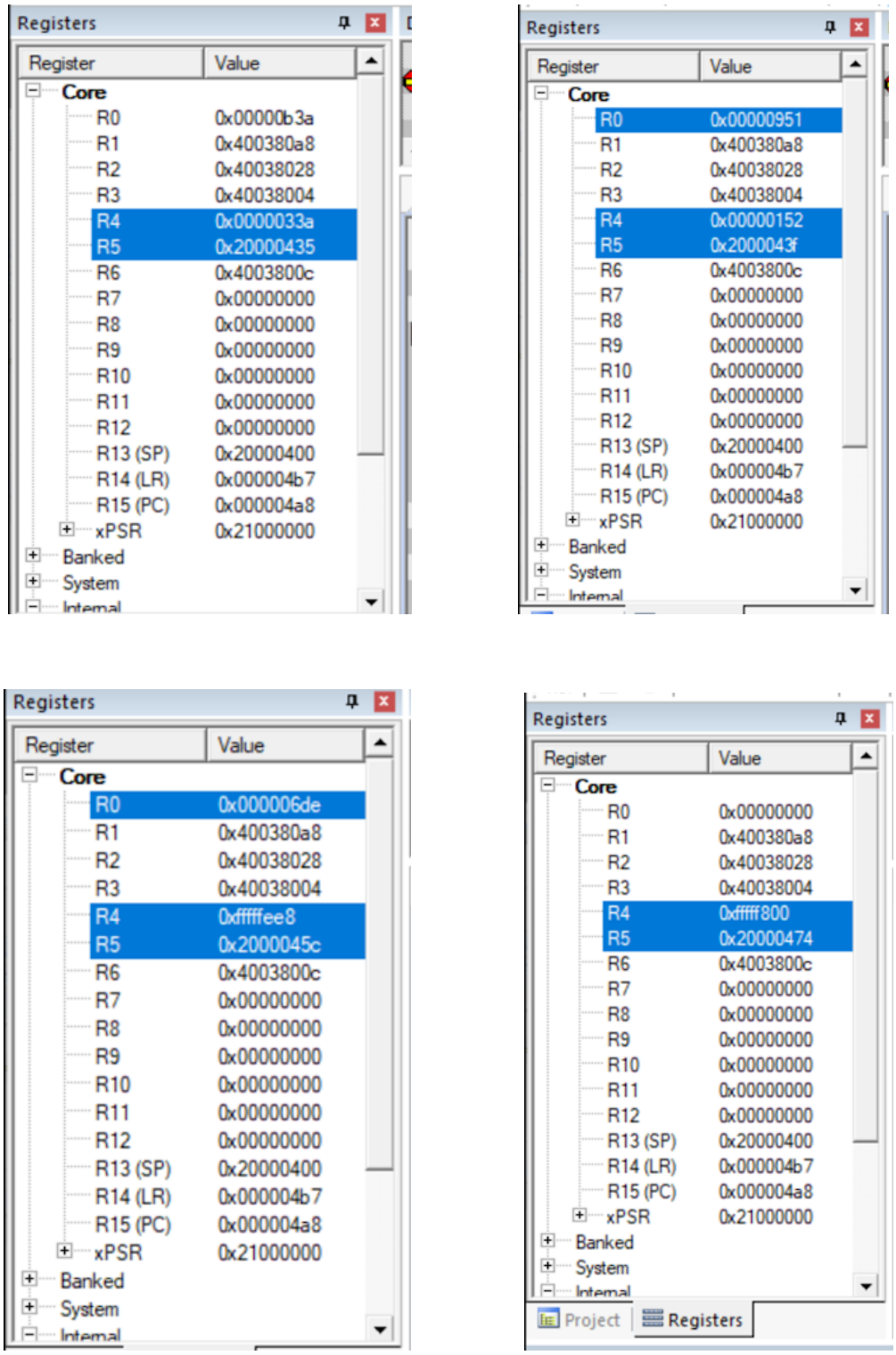


Figure 6. Register values, where R4 is changed from 0x000 to 0xfffff800

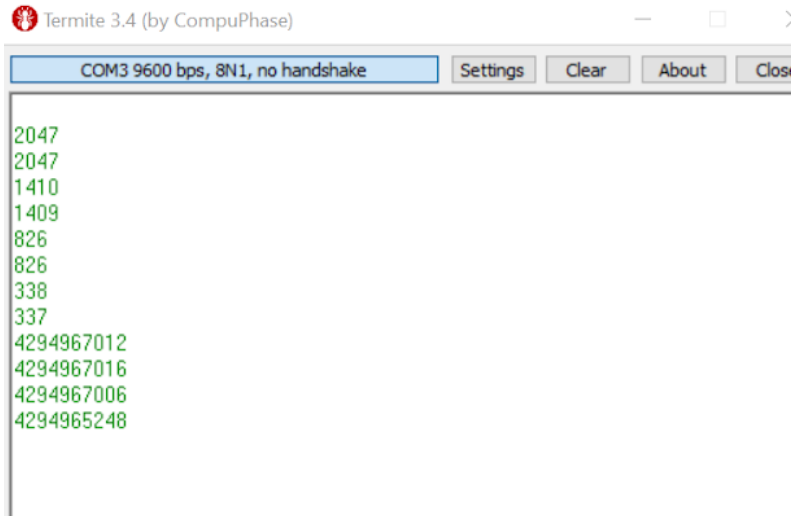


Figure 7. Termit Output of Part2

Part 3)

In this part, additional to the Part 2 value in the R0 (read from the FIFO reg) bitwise anded to find whether a value in R0 is negative positive or negative. If it is negative from ANDS, '1' comes and it jumps to negative branch. In this branch 2's complement of the R0 is taken. After taking complement, display branch is run. If the value of R0 is positve than it directly goes to the display_pos branch. Moreover in display_neg, - symbol is added through out char and Convrt.s is changed to add '.' after the first digit.The altered Convrt.s function is shown in Fig.9. Termit output is given in Fig. 10.

```
;*******
; Program_Directives.s
;*******
ADC0_RIS        EQU 0x40038004 ; Interrupt status
ADC0_PSSI       EQU 0x40038028 ; Initiate sample
ADC0_SSFIFO3    EQU 0x400380A8 ; Channel 3 results
ADC0_ISC        EQU 0x4003800C
MY_ADDR         EQU 0x20000400
;*******
; Program section
;*******
;LABEL              DIRECTIVE  VALUE                COMMENT
                    AREA       main, READONLY, CODE
                    THUMB
                    EXTERN          ATD_PORT_INIT
                    EXTERN          OutStr
                    EXTERN          OutChar
                    EXTERN          Convrt
                    EXPORT     __main; Make available
```

```
__main          PROC
                BL ATD_PORT_INIT
start
                ; start sampling routine
                LDR R3, =ADC0_RIS ; interrupt address
                LDR R1, =ADC0_SSFIFO3 ; result address
                LDR R2, =ADC0_PSSI ; sample sequence initiate address
                LDR R6,= ADC0_ISC ;interrupt status clear reg
                LDR R5,=MY_ADDR

                ; initiate sampling by enabling sequencer 3 in ADC0_PSSI
Sample          LDR R0, [R2]
                ORR R0, R0, #0x08 ; set bit 3 for SS3
                STR R0, [R2]

                ; check for sample complete (bit 3 of ADC0_RIS set)
Cont            LDR R0, [R3]
                ANDS R0, R0, #8
                BEQ Cont ;if flag is 0

                ;branch fails if the flag is set so data can be read and flag is cleared
                LDR R0, [R1]
                SUB R0,#0x800 ;substract 2048 == 1.65V

                ANDS R7, R0, #0x80000000
                BNE negative

display_pos
                MOV R8,#165
                MUL R0,R8
                MOV R8,#0x800
                UDIV R0,R8
                MOV R4,R0
                BL Convrt
                BL OutStr
                MOV R0, #8
                STR R0, [R6] ; clear flag
                B Sample

display_neg
                MOV R8,#165
                MUL R0,R8
                MOV R8,#0x800
                UDIV R0,R8
                ;print -
                PUSH {R5}
                MOV R5, "-"
                BL      OutChar
                POP {R5}
```

```
                    MOV R4,R0
                    BL Convrt
                    BL OutStr
                    MOV R0, #8
                    STR R0, [R6] ; clear flag
                    B Sample

negative            EOR R0,#0xFFFFFFFF
                    ADD R0,R0,#1
                    B display_neg

                    B start

                    ENDP
;*******
; End of the program  section
;*******
;LABEL     DIRECTIVE      VALUE                    COMMENT
                    ALIGN
                    END
```

Figure 8. Program Directive Code for Part 3

```
;*************************************************************
; Convrt.s
; Def: Converts max 32-bit hex number's decimal equivalent to ascii
; Writes the ascii result sstarting from [R5] address
;*************************************************************
; Constants

ASCII          EQU                  0x030

;*************************************************************
; Program section
;*************************************************************
;LABEL              DIRECTIVE   VALUE                COMMENT
                    AREA convert , READONLY, CODE
                    THUMB
                    EXPORT Convrt ; Reference external subroutine

Convrt              PROC

                    PUSH          {R0,R1,R2,R3,R4,R5,R6,R7,R8}

                    MOV                 R1,#0x0A
                    MOV           R6,#0
```

```
start       UDIV    R2, R4, R1 ;divide by ten, div->R2
            MUL     R3, R2, R1 ;multiply div*10 -> R3
            SUB     R0, R4, R3; Least significant digit -> R0
            ADD     R0,#ASCII ;convert digits to ascii
            PUSH    {R0}
            MOV         R4,R2

            ADD     R6,#0x01 ; counts how many decimal digits exists in
the number
            CMP     R2, #0
            BNE     start

            CMP     R6, #3
            BEQ     write_reg

            CMP     R6,#2       ;addonezero
            BEQ     addonezero

            CMP     R6,#1       ;add twozero
            BEQ     addtwozero

addonezero
            MOV  R7, #0x30
            STRB R7,[R5],#1

            MOV R7, #0x2E       ;add dot
            STRB R7,[R5],#1

            B write_reg

addtwozero  MOV  R7, #0x30
            STRB R7,[R5],#1

            MOV R7, #0x2E       ;add dot
            STRB R7,[R5],#1

            MOV  R7, #0x30
            STRB R7,[R5],#1

            B write_reg

            ;ascii equivalent of digits are pushed to stack, then write to location
of R5

write_reg
            SUB     R6,#0x01

            POP     {R7}
```

```
                    STRB  R7,[R5],#1

                    CMP   R6, #2
                    BNE   cont1
                    MOV   R7,  #0x2E            ;add dot
                    STRB  R7, [R5],#1



cont1               CMP   R6, #0x00
                    BNE   write_reg
                    MOV   R0, #0x0D
                    STRB  R0,[R5],#1
                    MOV   R0,#0x04
                    STRB  R0,[R5]


                    POP   {R0,R1,R2,R3,R4,R5,R6,R7,R8}
                    BX LR
                    ENDP

                    ALIGN
                    END
```

Figure 9. Altered Convrt Code

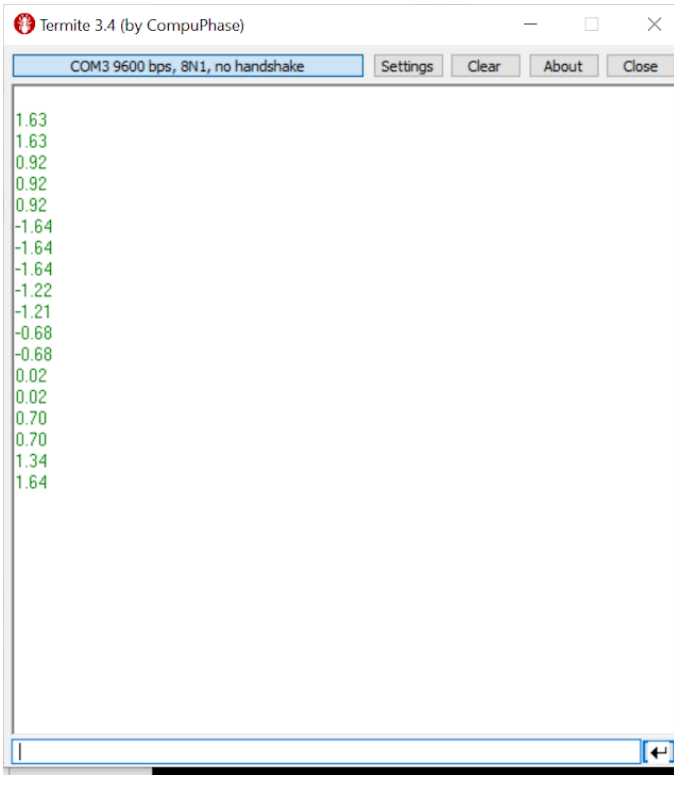


Figure 10. Termit Output of Part 3

Zeynepnur ŞAHİNEL
2305399
Wed 12:40 16.01.2021

Part 4)

In this part, additional to Part 3, DELAY subroutine is added (Fig.12). And this subroutine is implemented inside the display branches before going to Sample branch. The main code is shown in Fig.11 in which added part is indicated with bold lines.

```
;*******
; Program_Directives.s
; Copies the table from one location
; to another memory location.
; Directives and Addressing modes are
; explained with this program.
;*******
ADC0_RIS        EQU 0x40038004 ; Interrupt status
ADC0_PSSI       EQU 0x40038028 ; Initiate sample
ADC0_SSFIFO3    EQU 0x400380A8 ; Channel 3 results
ADC0_ISC        EQU   0x4003800C
MY_ADDR         EQU 0x20000400
ONESEC          EQU 5333330
;*******
; Program section
;*******
;LABEL          DIRECTIVE  VALUE              COMMENT
                AREA       main, READONLY, CODE
                THUMB
                EXTERN          ATD_PORT_INIT
                EXTERN          OutStr
                EXTERN          OutChar
                EXTERN          Convrt
                EXTERN          DELAY
                EXPORT     __main; Make available

__main      PROC
                BL ATD_PORT_INIT
start
                ; start sampling routine
                LDR R3, =ADC0_RIS ; interrupt address
                LDR R1, =ADC0_SSFIFO3 ; result address
                LDR R2, =ADC0_PSSI ; sample sequence initiate address
                LDR R6,= ADC0_ISC ;interrupt status clear reg
                LDR R5,=MY_ADDR

                ; initiate sampling by enabling sequencer 3 in ADC0_PSSI
Sample          LDR R0, [R2]
                ORR R0, R0, #0x08 ; set bit 3 for SS3
                STR R0, [R2]

                ; check for sample complete (bit 3 of ADC0_RIS set)
Cont        LDR R0, [R3]
```

```
                    ANDS R0, R0, #8
                    BEQ Cont ;if flag is 0

                    ;branch fails if the flag is set so data can be read and flag is cleared
                    LDR R0,[R1]
                    ;STR R0,[R5],#4 ;store the data
                    SUB R0,#0x800 ;substract 2048 == 1.65V

                    ANDS R7,R0,#0x80000000
                    BNE negative

display_pos
                    MOV R8,#165
                    MUL R0,R8
                    MOV R8,#0x800
                    UDIV R0,R8
                    MOV R4,R0
                    BL Convrt
                    BL OutStr
                    MOV R0, #8
                    STR R0, [R6] ; clear flag

                    LDR R0, =ONESEC
                    BL DELAY
                    B Sample

display_neg
                    MOV R8,#165
                    MUL R0,R8
                    MOV R8,#0x800
                    UDIV R0,R8

;print minus
                    PUSH {R5}
                    MOV R5, "-"
                    BL      OutChar
                    POP {R5}

                    MOV R4,R0
                    BL Convrt
                    BL OutStr
                    MOV R0, #8
                    STR R0, [R6] ; clear flag
                    LDR R0, =ONESEC
                    BL DELAY
                    B Sample

negative        EOR R0,#0xFFFFFFFF
                    ADD R0,R0,#1
```

```
                    B display_neg

                    B start
                    ENDP
;*******
; End of the program  section
;*******
;LABEL     DIRECTIVE      VALUE                    COMMENT
                    ALIGN
                    END
```

Figure 11. Program Directive Code of Part 4

```
                    ;************DELAY100**************
; Delay subroutine
; Input: R0  count
; Output: none
;When the delay subroutine is executed, the microp. does not execute other tasks.

                              AREA delayy, CODE, READONLY
                              THUMB
                              EXPORT DELAY

DELAY                         PROC
                              SUBS R0, R0, #1              ; R0 = R0 - 1
                              BNE DELAY                 ; Until R0 = 0, it returns
DELAY100 subroutine.
                              BX  LR                    ; returns to the main task
                              ENDP
```

Figure 12. Delay Subroutine

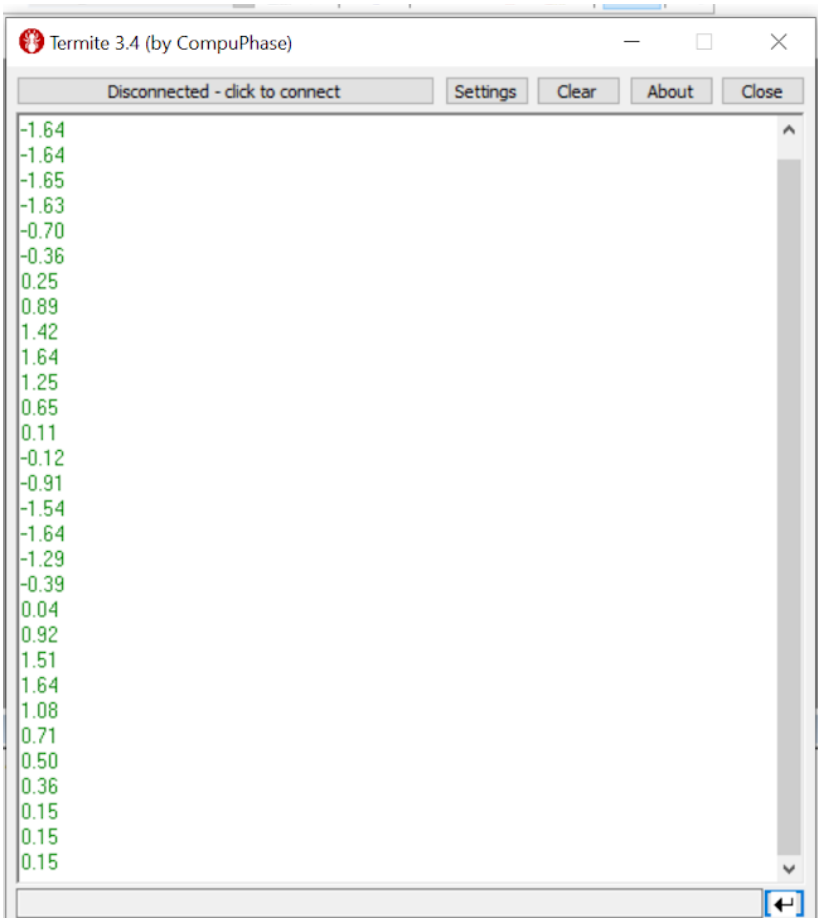


Figure 13. Termit Output of Part4