아래는 csd_asm.S 코드입니다.

```
#include "csd_zynq_peripherals.h"
#include "uart_init.s"
#include "uart_timer.s"

#define TIMER_INITIAL  0x500000

.global main
main:

        // -------------------------------------
        //  To distribute the learning burden.
        // -- stack is NOT used in private timer code
        // -- stack will be used in the interrupt code
        // -------------------------------------

        // -----------------
   // Stack setup begin
        // -----------------

        mrs     r0, cpsr                /* get the current PSR */
        mvn     r1, #0x1f               /* To mask out the MODE bits (M[4:0]) in
CPSR */
        and     r2, r1, r0
        orr     r2, r2, #0x12   /* IRQ mode */
        msr     cpsr, r2
        ldr     r13,=irq_stack_top      /* IRQ stack pointer */

        mrs     r0, cpsr                /* get the current PSR */
        mvn     r1, #0x1f               /* To mask out the MODE bits (M[4:0]) in
CPSR */
        and     r2, r1, r0
        orr     r2, r2, #0x13   /* Supervisor mode */
        msr     cpsr, r2
        ldr     r13,=svc_stack_top      /* Supervisor stack pointer */

        mrs     r0, cpsr                /* get the current PSR */
        mvn     r1, #0x1f               /* To mask out the MODE bits (M[4:0]) in
CPSR */
        and     r2, r1, r0
        orr     r2, r2, #0x11   /* FIQ mode */
        msr     cpsr, r2
        ldr     r13,=fiq_stack_top      /* FIQ stack pointer */

        mrs     r0, cpsr                /* get the current PSR */
        mvn     r1, #0x1f               /* To mask out the MODE bits (M[4:0]) in
CPSR */
        and     r2, r1, r0
        orr     r2, r2, #0x1F   /* SYS mode */
        msr     cpsr, r2

        // ---------------
   // Stack setup end
        // ---------------

        // Private Timer Load Register
        ldr r0, =PRIVATE_LOAD
        ldr r1, =TIMER_INITIAL
        str r1, [r0]

        // Private Timer Control Register
        ldr r0, =PRIVATE_CONTROL
        mov r1, #78 << 8    // Prescalar
```

```
        orr r1, r1, #7      // IRQ Enable, Auto-Reload, Timer Enable
        str r1, [r0]

        ldr r2, =PRIVATE_COUNTER
        ldr r3, =PRIVATE_STATUS

        uart_init

        mov r6, #1
        mov r7, #0 // second
        mov r8, #0
        mov r9, #0 // minute
        mov r10, #0
        mov r11, #0 // hour

        add r4, pc, #4
        mov lr, r4
        timer_initialize

forever:

        add r4, pc, #4
        mov lr, r4
        uart_timer

        cmp  r12, #1
        streq  r12, [r3] // clear sticky bit in the status register

        b forever


.data
.align 4

irq_stack:      .space 1024
irq_stack_top:
fiq_stack:      .space 1024
fiq_stack_top:
svc_stack:      .space 1024
svc_stack_top:
```

아래는 csd_zynq_peripherals.h 코드입니다.

```
/*
 * csd_zynq_peripherals.h
 *
 *  Created on: 2014. 8. 23.
 *      Author: Taeweon
 */

#ifndef CSD_ZYNQ_PERIPHERALS_H_
#define CSD_ZYNQ_PERIPHERALS_H_

#define SLCR                    0xF8000000
#define SLCR_LOCK               SLCR + 0x4
#define SLCR_UNLOCK             SLCR + 0x8
#define SLCR_LOCKSTA SLCR + 0xC
#define TZ_DDR_RAM              SLCR + 0x430


#define GICD_BASE          0xF8F01000         // Distributor Base Address
#define GICD_CTLR          GICD_BASE + 0x0    // Distributor Control Register
#define GICD_TYPER             GICD_BASE + 0x4    // Interrupt Controller Type
Register
#define GICD_IGROUP0  GICD_BASE + 0x80   // Interrupt Group Register
#define GICD_ISENABLER0        GICD_BASE + 0x100  // Interrupt Set-Enable
Register 0
#define GICD_ISPENDR0 GICD_BASE + 0x200  // Interrupt Set-Pending Register 0
#define GICD_ICACTIVE0         GICD_BASE + 0x380  // Interrupt Clear-Active 0
#define GICD_ICACTIVE1         GICD_BASE + 0x384  // Interrupt Clear-Active 1
#define GICD_ICACTIVE2         GICD_BASE + 0x388  // Interrupt Clear-Active 2
#define GICD_ICACTIVE3         GICD_BASE + 0x38c  // Interrupt Clear-Active 3
#define GICD_ICACTIVE4         GICD_BASE + 0x390  // Interrupt Clear-Active 4
#define GICD_ICACTIVE5         GICD_BASE + 0x394  // Interrupt Clear-Active 5
#define GICD_ICACTIVE6         GICD_BASE + 0x398  // Interrupt Clear-Active 6
#define GICD_ICACTIVE7         GICD_BASE + 0x39c  // Interrupt Clear-Active 7
#define GICD_PRIOR0            GICD_BASE + 0x400  // Interrupt Priority Register 0
#define GICD_PRIOR7            GICD_PRIOR0 + 4*7   // Interrupt Priority Register
7
#define GICD_SGIR              GICD_BASE + 0xF00  // Software Generated
Interrupt Register
#define GICD_SPENDSGIR0        GICD_BASE + 0xF20  // SGI Set-Pending Register 0
#define GICD_ICPIDR1   GICD_BASE + 0xFE4  // Peripheral ID1 (ICPIDR1)
#define GICD_ICPIDR2   GICD_BASE + 0xFE8  // Peripheral ID2 (ICPIDR2)

#define GICC_CTLR      0xF8F00100        // CPU Interface Control Register
#define GICC_PMR           GICC_CTLR + 0x4   // CPU Interface Priority Mask
Register
#define GICC_BPR           GICC_CTLR + 0x8   // CPU Binary Pointer Register
#define GICC_IAR           GICC_CTLR + 0xC   // CPU Interface Ack Register
#define GICC_EOIR          GICC_CTLR + 0x10  // CPU Interface End of Interrupt
Register
#define GICC_IIDR          GICC_CTLR + 0xFC  // CPU Interface Identification
Register


#define PRIVATE_TIMER          0xF8F00600            // Private Timer Base
Address
#define PRIVATE_LOAD      PRIVATE_TIMER + 0x0   // Private Timer Load Register
#define PRIVATE_COUNTER       PRIVATE_TIMER + 0x4   // Private Timer Counter
Register
#define PRIVATE_CONTROL       PRIVATE_TIMER + 0x8   // Private Timer Control
Register
#define PRIVATE_STATUS        PRIVATE_TIMER + 0xC   // Private Timer Interrupt
Status Register
```

```
#endif /* CSD_ZYNQ_PERIPHERALS_H_ */
```

아래는 uart_init.s 코드입니다.

```
#include "uart_regs.h"

.macro uart_init

/*
# 1.Reset controller
        ldr             r0, =slcr_UART_RST_CTRL
        ldr             r1, [r0, #0]                    @ read
slcr.UART_RST_CTRL
        orr             r1, r1, #0x0000000A             @ set both
slcr.UART_RST_CTRL[UART1_REF_RST, UART1_CPU1X_RST] (bit[3][1])
        str             r1, [r0, #0]                    @ update

# 2.Configure I/O signal routing
# 3.Configure UART_Ref_Clk
        ldr             r0, =slcr_UART_CLK_CTRL
        ldr             r1, =0x00001402                 @ write 0x00001402 to
slcr.UART_CLK_CTRL @ mov    r1, #0x00001402(*** ERROR ***)
        str             r1, [r0, #0]                    @ update
*/

# 4.Configure controller functions
        ldr             r0, =UART1_BASE

#       4-1. Configure UART character frame
        mov             r1, #0x00000020
        str             r1, [r0, #UART_MODE_REG0_OFFSET]

#       4-2. Configure the Baud Rate
        # a-b. Disable Rx Path and Tx Path
        ldr             r1, [r0, #UART_CONTROL_REG0_OFFSET]       @ read
uart.Control_reg0
        bic             r1, r1, #0x0000003C                                 @
clear TXDIS, TXEN, RXDIS, RXEN (bit[5][4][3][2])
        orr             r1, r1, #0x00000028                                 @
TXDIS = 1 TXEN = 0 RXDIS = 1 RXEN = 0 (bit[5][4][3][2])
        str             r1, [r0, #UART_CONTROL_REG0_OFFSET]       @ update
        # c-d. Write the calculated CD value and BDIV
        mov             r1, #0x0000003E
                        @ CD = 62 (Baud rate 115200)
        str             r1, [r0, #UART_BAUD_RATE_GEN_REG0_OFFSET]       @
update uart.Baud_rate_gen_reg0
        mov             r1, #0x00000006
                @ BDIV = 6 (Baud rate 115200)
        str             r1, [r0, #UART_BAUD_RATE_DIV_REG0_OFFSET]       @
update uart.Baud_rate_divider_reg0 @ strb      r1, [r0,
#UART_BAUD_RATE_DIV_REG0_OFFSET]
        # e. Reset Tx and Px Path
        ldr             r1, [r0, #UART_CONTROL_REG0_OFFSET]       @ read
uart.Control_reg0
        orr             r1, r1, #0x00000003                                 @
set TXRST, RXRST (bit[1][0]:self-clearing) - this resets Tx and Rx paths
        str             r1, [r0, #UART_CONTROL_REG0_OFFSET]       @ update
        # f-g. Enable Rx Path and Tx Path
        ldr             r1, [r0, #UART_CONTROL_REG0_OFFSET]       @ read
uart.Control_reg0
        bic             r1, r1, #0x0000003C                                 @
clear TXDIS, TXEN, RXDIS, RXEN (bit[5][4][3][2])
        orr             r1, r1, #0x00000014                                 @
TXDIS = 0 TXEN = 1 RXDIS = 0 RXEN = 1 (bit[5][4][3][2])
        str             r1, [r0, #UART_CONTROL_REG0_OFFSET]       @ update
```

```
#       4-5. Enable the Controller
        ldr             r1, =0x00000117                                 @
write 0x00000117 to uart.Control_reg0
        str             r1, [r0, #UART_CONTROL_REG0_OFFSET]       @ update

.endm
```

아래는 uart_regs.h 코드입니다.

```
#define slcr_UART_RST_CTRL                          0xF8000228
#define slcr_UART_CLK_CTRL                          0xF8000154

#define UART1_BASE                                          0xE0001000
#define UART_CONTROL_REG0_OFFSET            0x0
#define UART_MODE_REG0_OFFSET                       0x4
#define UART_INTRPT_EN_REG0_OFFSET                  0x8
#define UART_INTRPT_DIS_REG0_OFFSET                 0xC
#define UART_INTRPT_MASK_REG0_OFFSET        0x10
#define UART_BAUD_RATE_GEN_REG0_OFFSET      0x18
#define UART_RCVR_TIMEOUT_REG0_OFFSET       0x1C
#define UART_RCVR_FIFO_TRG_LV0_OFFSET       0x20
#define UART_CHANNEL_STS_REG0_OFFSET        0x2C
#define UART_TX_RX_FIFO0_OFFSET                     0x30
#define UART_BAUD_RATE_DIV_REG0_OFFSET      0x34
```

아래는 uart_timer.s 코드입니다.

```
.data

string_initialize:

        .ascii "00:00:00"
        .byte 0x0D
        .byte 0x0A
        .byte 0x00

.text

.macro timer_initialize

        b timer_print

.endm

.macro uart_timer

        b uart_print

.endm

#include "uart_regs.h"

uart_print:

////////////////////////////////////////////
//        아래에 오류 발생 시 clean project          //
////////////////////////////////////////////

        ldr r4, [r2]
        ldr r5, [r3] // update digital watch if count is equal to 0
        cmp r5, #1 // if count is equal to 0 then event flag will be 1
        moveq r12, #1

        movne           pc, lr                          @     return to the caller

        ldr      r0, =0xE0001000

TRANSMIT_loop:

        // ---------  Check to see if the Tx FIFO is empty
-----------------------------
        ldr      r4, [r0, #0x2C] @ get Channel Status Register
        and      r4, r4, #0x8            @ get Transmit Buffer Empty bit(bit[3:3])
        cmp      r4, #0x8                                @ check if TxFIFO is empty
and ready to receive new data
        bne      TRANSMIT_loop          @ if TxFIFO is NOT empty, keep checking
until it is empty

//-------------------------------------------------------------------------------
---

        add r5, r11, #48
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48
        add r5, r10, #48
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48 // print hour
        ldrb r5, =#58
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48
        add r5, r9, #48
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48
```

```
        add r5, r8, #48
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48 // print minute
        ldrb r5, =#58
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48
        add r5, r7, #48
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48
        add r5, r6, #48
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48 // print second
        ldrb r5, =#13
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48
        ldrb r5, =#10
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48

        add r6, #1 // count the next clock

        cmp r6, #10
        addeq r7, #1
        moveq r6, #0 // update the number of second

        cmp r7, #6
        addeq r8, #1
        moveq r7, #0 // update the number of second

        cmp r8, #10
        addeq r9, #1
        moveq r8, #0 // update the number of minute

        cmp r9, #6
        addeq r10, #1
        moveq r9, #0 // update the number of minute

        cmp r10, #10
        addeq r11, #1
        moveq r10, #0 // update the number of hour

        cmp r11, #10
        moveq r11, #0 // initialize the timer

        mov             pc, lr                          @     return to the caller

timer_print:

        ldr     r0, =0xE0001000
        ldr     r1, =string_initialize

TRANSMIT_loop_1:

        // ---------  Check to see if the Tx FIFO is empty
------------------------------
        ldr     r4, [r0, #0x2C] @ get Channel Status Register
        and     r4, r4, #0x8            @ get Transmit Buffer Empty bit(bit[3:3])
        cmp     r4, #0x8                                @ check if TxFIFO is empty
and ready to receive new data
        bne     TRANSMIT_loop_1             @ if TxFIFO is NOT empty, keep
checking until it is empty

//-------------------------------------------------------------------------
---

        ldrb r5, [r1], #1
        strb            r5, [r0, #0x30]  @ fill the TxFIFO with 0x48
        cmp r5, #0x00
        bne TRANSMIT_loop_1

        mov             pc, lr                          @     return to the caller
```