Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

#48

강사 : Innova Lee(이 상훈)
학생 : 김 시윤

1.배운내용 복습.

부트로더

```
HL_sys_intvecs.asm
     .sect ".intvecs"
     .arm
; import reference for interrupt routines

     .ref _c_int00
     .ref phantomInterrupt
     .def resetEntry

; interrupt vectors

resetEntry
          b     _c_int00
undefEntry
          b     undefEntry
svcEntry
          b     svcEntry
prefetchEntry
          b     prefetchEntry
dataEntry
          b     dataEntry
          b     phantomInterrupt
          ldr  pc,[pc,#-0x1b0]
          ldr  pc,[pc,#-0x1b0]
;-------------------------------------------------------------------
```

b   _c_int00
c_int00으로 jump 한다.

```
/* Include Files */

#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_sys_vim.h"
#include "HL_sys_core.h"
#include "HL_esm.h"
#include "HL_sys_mpu.h"

extern void exit(int _status);

/* Startup Routine */
void _c_int00(void);

#pragma CODE_STATE(_c_int00, 32)
#pragma INTERRUPT(_c_int00, RESET)
#pragma WEAK(_c_int00)

void _c_int00(void)
{
    /* Initialize Core Registers to avoid CCM Error */
    _coreInitRegisters_();

    /* Initialize Stack Pointers */
    _coreInitStackPointer_();
```

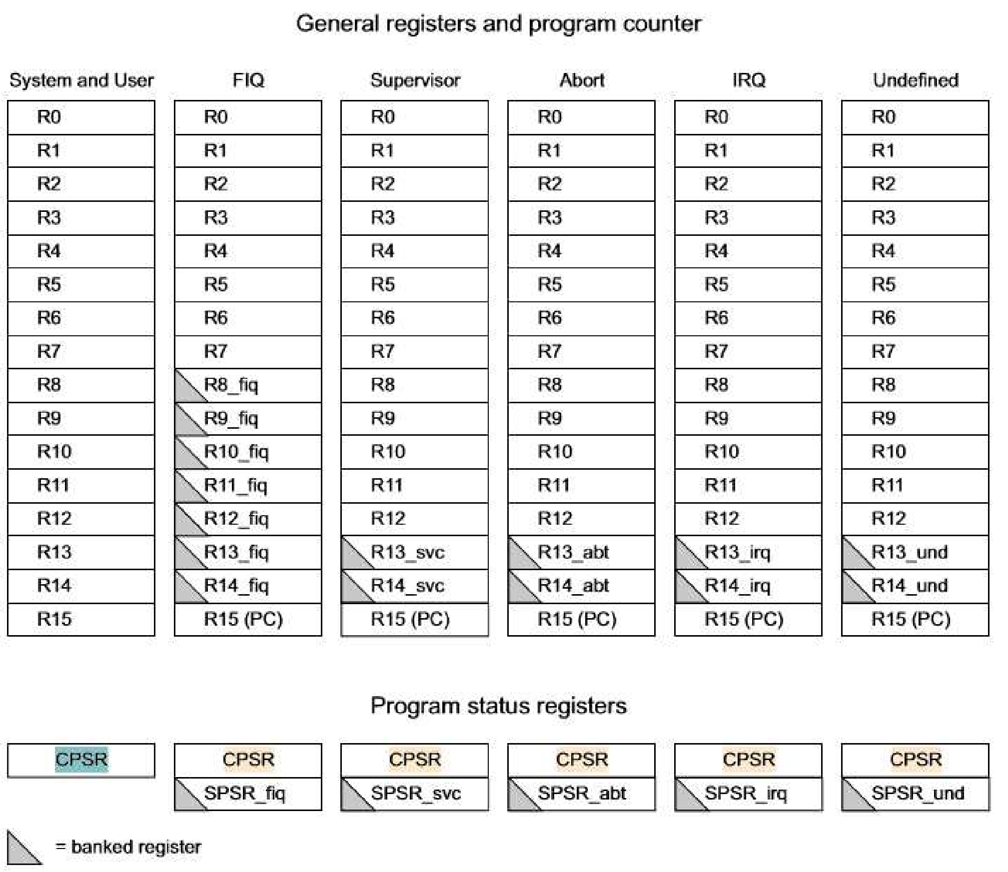
```
switch(getResetSource())
{
    case POWERON_RESET:
    case DEBUG_RESET:
    case EXT_RESET:

    _memInit_();

    _coreEnableEventBusExport_();

    if ((esmREG->SR1[2]) != 0U)
    {
        esmGroup3Notification(esmREG,esmREG->SR1[2]);

    }

    systemInit();

    _coreEnableIrqVicOffset_();

        vimInit();

    esmInit();

    break;

    case OSC_FAILURE_RESET:
    break;
        case WATCHDOG_RESET:
        case WATCHDOG2_RESET:
        break;

        case CPU0_RESET:

        _coreEnableEventBusExport_();

        break;

        case SW_RESET:

        break;

        default:

        break;
    }

    _mpuInit_();

    _cacheEnable_();

    __TI_auto_init();

    exit(0);
}
```

coreInitRegisters();
레지스터 초기화 하는 부분으로 들어간다.
"HL_sys_core.asm"에 어셈으로 정의되어있는걸 확인해본다.

```
_coreInitRegisters_

    ; After reset, the CPU is in the Supervisor mode (M = 10011)
        mov r0, lr
        mov r1, #0x0000
        mov r2, #0x0000
        mov r3, #0x0000
        mov r4, #0x0000
        mov r5, #0x0000
        mov r6, #0x0000
        mov r7, #0x0000
        mov r8, #0x0000
        mov r9, #0x0000
        mov r10, #0x0000
        mov r11, #0x0000
        mov r12, #0x0000
        mov r13, #0x0000
        mrs r1, cpsr
        msr spsr_cxsf, r1
        ; Switch to FIQ mode (M = 10001)
        cps #17
        mov lr, r0
        mov r8, #0x0000
        mov r9, #0x0000
        mov r10, #0x0000
        mov r11, #0x0000
        mov r12, #0x0000
        mrs r1, cpsr
        msr spsr_cxsf, r1
        ; Switch to IRQ mode (M = 10010)
        cps #18
        mov lr, r0
        mrs r1,cpsr
        msr spsr_cxsf, r1
        ; Switch to Abort mode (M = 10111)
        cps #23
        mov lr, r0
        mrs r1,cpsr
        msr spsr_cxsf, r1
        ; Switch to Undefined Instruction Mode (M = 11011)
        cps #27
        mov lr, r0
        mrs r1,cpsr
        msr spsr_cxsf, r1
        ; Switch to System Mode ( Shares User Mode registers ) (M = 11111)
        cps #31
        mov lr, r0
        mrs r1,cpsr
        msr spsr_cxsf, r1


        mrc   p15,     #0x00,      r2,       c1, c0, #0x02
        orr   r2,      r2,         #0xF00000
        mcr   p15,     #0x00,      r2,       c1, c0, #0x02
        mov   r2,      #0x40000000
        fmxr  fpexc,   r2

        fmdrr d0,         r1,     r1
        fmdrr d1,         r1,     r1
        fmdrr d2,         r1,     r1
        fmdrr d3,         r1,     r1
```

```
        fmdrr d4,          r1,      r1
        fmdrr d5,          r1,      r1
        fmdrr d6,          r1,      r1
        fmdrr d7,          r1,      r1
        fmdrr d8,          r1,      r1
        fmdrr d9,          r1,      r1
        fmdrr d10,         r1,      r1
        fmdrr d11,         r1,      r1
        fmdrr d12,         r1,      r1
        fmdrr d13,         r1,      r1
        fmdrr d14,         r1,      r1
        fmdrr d15,         r1,      r1
        bl     next1
next1
        bl     next2
next2
        bl     next3
next3
        bl     next4
next4
        bx     r0

    .endasmfunc
```

확인해보면 모드별로 spsr을 초기화 시킨다.



위처럼 각 모드마다의 레지스터를 초기화 시킨다.

```
_coreInitStackPointer_();

_coreInitStackPointer_

        cps    #17
        ldr    sp,        fiqSp
        cps    #18
        ldr    sp,        irqSp
        cps    #19
```

```
        ldr    sp,        svcSp
        cps    #23
        ldr    sp,        abortSp
        cps    #27
        ldr    sp,        undefSp
        cps    #31
        ldr    sp,        userSp
        bx     lr

userSp  .word 0x08000000+0x00001000
svcSp   .word 0x08000000+0x00001000+0x00000100
fiqSp   .word 0x08000000+0x00001000+0x00000100+0x00000100
irqSp   .word 0x08000000+0x00001000+0x00000100+0x00000100+0x00000100
abortSp .word 0x08000000+0x00001000+0x00000100+0x00000100+0x00000100+0x00000100
undefSp .word 0x08000000+0x00001000+0x00000100+0x00000100+0x00000100+0x00000100+0x00000100

    .endasmfunc
```

Stack Pointer를 초기화하는 함수이다. 아직 실제로 스택이 할당된 것은 아니다. 스택 공간을 할당해 주었는데 그 공간을 초과해버리면 오류가난다.

나중에 우리가 각 모드별 스택공간을 조정할 수 있다.

스택을 초기화 한 후 리턴하고 나면

getResetSource() 가 호출된다.

```
resetSource_t getResetSource(void)
{
    register resetSource_t rst_source;

    if ((SYS_EXCEPTION & (uint32)POWERON_RESET) != 0U)
    {
        /* power-on reset condition */
        rst_source = POWERON_RESET;

        /* Clear all exception status Flag and proceed since it's power up */
        SYS_EXCEPTION = 0x0000FFFFU;
    }
    else if ((SYS_EXCEPTION & (uint32)EXT_RESET) != 0U)
    {
        /* Reset caused due to External reset. */
        rst_source = EXT_RESET;
        SYS_EXCEPTION = (uint32)EXT_RESET;
    }
    else if ((SYS_EXCEPTION & (uint32)DEBUG_RESET) !=0U)
    {
        /* Reset caused due Debug reset request */
        rst_source = DEBUG_RESET;
        SYS_EXCEPTION = (uint32)DEBUG_RESET;
    }
    else if ((SYS_EXCEPTION & (uint32)OSC_FAILURE_RESET) != 0U)
    {
     /* Reset caused due to oscillator failure.
        Add user code here to handle oscillator failure */
        rst_source = OSC_FAILURE_RESET;
```

```
        SYS_EXCEPTION = (uint32)OSC_FAILURE_RESET;
    }
    else if ((SYS_EXCEPTION & (uint32)WATCHDOG_RESET) !=0U)
    {
        /* Reset caused due watchdog violation */
        rst_source = WATCHDOG_RESET;
        SYS_EXCEPTION = (uint32)WATCHDOG_RESET;
    }
    else if ((SYS_EXCEPTION & (uint32)WATCHDOG2_RESET) !=0U)
    {
        /* Reset caused due watchdog violation */
        rst_source = WATCHDOG2_RESET;
        SYS_EXCEPTION = (uint32)WATCHDOG2_RESET;
    }
    else if ((SYS_EXCEPTION & (uint32)CPU0_RESET) !=0U)
    {
        /* Reset caused due to CPU0 reset.
        CPU reset can be caused by CPU self-test completion, or
        by toggling the "CPU RESET" bit of the CPU Reset Control
Register. */
        rst_source = CPU0_RESET;
        SYS_EXCEPTION = (uint32)CPU0_RESET;
    }
    else if ((SYS_EXCEPTION & (uint32)SW_RESET) != 0U)
    {
        /* Reset caused due to software reset. */
        rst_source = SW_RESET;
        SYS_EXCEPTION = (uint32)SW_RESET;
    }
    else
    {
        /* No_reset occured. */
        rst_source = NO_RESET;
    }

    return rst_source;
}
```

SYS_EXCEPTION은 0xFFFFFFE4로 정의되어 있고, E4번 offset이 어떤 System Control Register인지 찾아보면,SYSESR 레지스터임을 알 수 있다. Power,Ocillator,Watchdog,Debug 등이 reset이 될 때, 해당 비트가 set 되는 register임을 알 수 있다.

2.5.1.46 System Exception Status Register (SYSESR)

The SYSESR register, shown in Figure 2-53 and described in Table 2-65, shows the source for different resets encountered. Previous reset source status bits are not automatically cleared if new resets occur. After reading this register, the software should clear any flags that are set so that the source of future resets can be determined. Any bit in this register can be cleared by writing a 1 to the bit.

Figure 2-53. System Exception Status Register (SYSESR) (offset = E4h)

31-16
Reserved
R-0

15	14	13	12	11	10-8
PORST	OSCRST	WDRST	Reserved	DBGRST	Reserved
R/WC-X	R/WC-X*	R/WC-X*	R-0	R/WC-X*	R-0

7	6	5	4	3	2-0
ICSTRST	Reserved	CPURST	SWRST	EXTRST	Reserved
R/WC-X*	R/WC-X*	R/WC-X*	R/WC-X*	R/WC-X*	R-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; X = value is unchanged after reset; X* = 0 after PORST but unchanged after other resets; -*n* = value after reset

Table 2-65. System Exception Status Register (SYSESR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15	PORST		Power-on reset. This bit is set when a power-on reset occurs, either internally asserted by the VMON or externally asserted by the nPORRST pin.
		0	No power-on reset has occurred since this bit was last cleared.
		1	A reset was caused by a power-on reset. (This bit should be cleared after being read so that subsequent resets can be properly identified as not being power-on resets.)
14	OSCRST		Reset caused by an oscillator failure or PLL cycle slip. This bit is set when a reset is caused by an oscillator failure or PLL slip. Write 1 will clear this bit. Write 0 has no effect. **Note: The action taken when an oscillator failure or PLL slip is detected must configured in the PLLCTL1 register.**
		0	No reset has occurred due to an oscillator failure or a PLL cycle slip.
		1	A reset was caused by an oscillator failure or a PLL cycle slip.
13	WDRST		Watchdog reset flag. This bit is set when the last reset was caused by the digital watchdog (DWD). Write 1 will clear this bit. Write 0 has no effect.
		0	No reset has occurred because of the DWD.
		1	A reset was caused by the DWD.
12	Reserved	0	Reads return 0. Writes have no effect.
11	DBGRST		Debug reset flag. This bit is set when the last reset was caused by the debugger reset request. Write 1 will clear this bit. Write 0 has no effect.
		0	No reset has occurred because of the debugger.
		1	A reset was caused by the debugger.
10-8	Reserved	0	Reads return 0. Writes have no effect.
7	ICSTRST		Interconnect reset flag. This bit is set when the last CPU reset was caused by the entering and exiting of interconnect self-test check. While the interconnect is under self-test check, the CPU is also held in reset until the interconnect self-test is complete.
		0	No CPUx reset has occurred because of an interconnect self-test check.
		1	A reset has occurred to the CPUx because of the interconnect self-test check.
6	Reserved	0	Reads return 0. Writes have no effect.

poweronreset 15번 비트 1

```
_memInit_
        ldr   r12, MINITGCR      ;Load MINITGCR register address
        mov   r4, #0xA
        str   r4, [r12]          ;Enable global memory hardware initialization

        ldr   r11, MSIENA        ;Load MSIENA register address
        mov   r4, #0x1           ;Bit position 0 of MSIENA corresponds to SRAM
        str   r4, [r11]          ;Enable auto hardware initalisation for SRAM
mloop                            ;Loop till memory hardware initialization comletes
        ldr   r5, MSTCGSTAT
        ldr   r4, [r5]
        tst   r4, #0x100
        beq   mloop

        mov   r4, #5
        str   r4, [r12]          ;Disable global memory hardware initialization
        bx lr
        .endasmfunc
```

Table 2-40. Memory Hardware Initialization Global Control Register (MINITGCR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	MINITGENA		Memory hardware initialization global enable key.
		Ah	Global memory hardware initialization is enabled.
		Others	Global memory hardware initialization is disabled. **Note: It is recommended that a value of 5h be used to disable memory hardware initialization. This value will give maximum protection from an event that would inadvertently enable the controller.**

0xA를 MINITGCR 에 셋해주었기 때문에 메모리 하드웨어 초기화 된다.

Table 2-41. MBIST Controller/Memory Initialization Enable Register (MSINENA) Field Descriptions

Bit	Field	Value	Description
31-0	MSIENA		PBIST controller and memory initialization enable register. In memory self-test mode, all the corresponding bits of the memories to be tested should be set before enabling the global memory self-test controller key (MSTGENA) in the MSTGCR register (offset 58h). The reason for this is that MSTGENA, in addition to being the global enable for all individual PBIST controllers, is the source for the reset generation to all the PBIST controller state machines. Disabling the MSTGENA or MINITGENA key (by writing from an Ah to any other value) will reset all the MSIENA[31-0] bits to their default values.
		0	*In memory self-test mode (MSTGENA = Ah):* PBIST controller [31-0] is disabled. *In memory initialization mode (MINITGENA = Ah):* Memory module [31-0] auto hardware initialization is disabled.
		1	*In memory self-test mode (MSTGENA = Ah):* PBIST controller [31-0] is enabled. *In memory initialization mode (MINITGENA = Ah):* Memory module [31-0] auto hardware initialization is enabled. **Note: Software should ensure that both the memory self-test global enable key (MSTGENA) and the memory hardware initialization global key (MINITGENA) are not enabled at the same time.**

MSIENA에 0x1이 셋되었고 위에서 MINITGCR 에 0xA를 셋해주었기 때문에
initialization mode memory module[0] auto hardware initialization enable 되었다.

```
mloop                                  ;Loop till memory hardware initialization comletes
          ldr    r5, MSTCGSTAT
          ldr    r4, [r5]
          tst    r4, #0x100
          beq    mloop
```

MSTCGSTAT 의 8번째 비트가 1이 될 때까지 무한루프를 돈다,
MSTCGSTAT를 확인해본다.

Table 2-42. MSTC Global Status Register (MSTCGSTAT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	MINIDONE		Memory hardware initialization complete status. **Note: Disabling the MINITGENA key (By writing from a Ah to any other value) will clear the MINIDONE status bit to 0.** **Note: Individual memory initialization status is shown in the MINISTAT register.**
		0	*Read:* Memory hardware initialization is not complete for all memory. *Write:* A write of 0 has no effect.
		1	*Read:* Hardware initialization of all memory is completed. *Write:* The bit is cleared to 0.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	MSTDONE		Memory self-test run complete status. **Note: Disabling the MSTGENA key (by writing from a Ah to any other value) will clear the MSTDONE status bit to 0.**
		0	*Read:* Memory self-test is not completed. *Write:* A write of 0 has no effect.
		1	*Read:* Memory self-test is completed. *Write:* The bit is cleared to 0.

확인해보면 8번째 비트가 1일 경우 하드웨어 초기화가 완료 되었다는 뜻이다.
즉 이 루프는 하드웨어 초기화가 완료될때까지 대기하는 루프인걸 확인하였다.

MINITGCR 에 5를 넣어준다.
0번째 비트와 3번째 비트가 1인걸 인지하고 MINITGCR레지스터를 확인한다.
0101 은 0xA가 아니라서 hardware 메모리 초기화를 disable 하고 리턴한다.