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

#46

강사 : Innova Lee(이 상훈)

학생 : 김 시윤

1.배운내용 복습.

ARM 부트 로더

```
; Reset handler
Reset_Handler   PROC
                EXPORT Reset_Handler            [WEAK]
        IMPORT SystemInit
        IMPORT __main

                ;FPU settings
                LDR    R0, =0xE000ED88          ; Enable CP10,CP11
                LDR    R1,[R0]
                ORR    R1,R1,#(0xF << 20)
                STR    R1,[R0]

                LDR    R0, =SystemInit
                BLX    R0
                LDR    R0, =__main
                BX     R0
                ENDP

; Dummy Exception Handlers (infinite loops which can be modified)

NMI_Handler     PROC
                EXPORT NMI_Handler              [WEAK]
                B      .
                ENDP
HardFault_Handler₩
                PROC
                EXPORT HardFault_Handler        [WEAK]
                B      .
```

Boot를 했을 때 Reset_Handler를 들어가게된다.
0xE000ED88를 레지스터에 로드 저장한다.
R0의 값을 R1에 저장한다.
R1 = 0xE000ED88 그다음 ORR로 어떤값을 셋팅한다.

Table 4-1 System control registers (continued)

Address	Name	Type	Reset	Description
0xE000ED64	ID_ISAR1	RO	0x02112000	Instruction Set Attributes Register 1
0xE000ED68	ID_ISAR2	RO	0x21232231	Instruction Set Attributes Register 2
0xE000ED6C	ID_ISAR3	RO	0x01111131	Instruction Set Attributes Register 3
0xE000ED70	ID_ISAR4	RO	0x01310102	Instruction Set Attributes Register 4
0xE000ED88	CPACR	RW	-	Coprocessor Access Control Register
0xE000EF00	STIR	WO	0x00000000	Software Triggered Interrupt Register

a. Bits [10:8] are reset to zero. The ENDIANNESS bit, bit [15], can reset to either state, depending on the implementation.
b. BFAR and MFAR are the same physical register. Because of this, the BFARVALID and MFAEVALID bits are mutually exclusive.

레퍼런스 데이터 시트를 보게되면
0xE000ED88 는 CPACR 레지스터 라는 것을 알 수 있다. 보조 프로세서 접근 제어 레지스터이다.

Coprocessor Access Control Register

The CPACR register specifies the access privileges for coprocessors. See the register summary in *Cortex-M4F floating-point system registers* for its attributes. The bit assignments are:

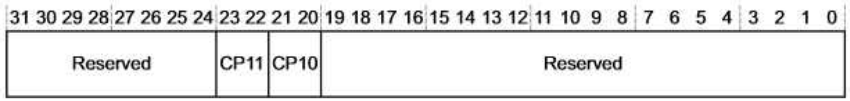


Table 4-50 CPACR register bit assignments

Bits	Name	Function
[31:24]	-	Reserved. Read as Zero, Write Ignore.
[2*n*+1:2*n*] for *n* values10 and 11	CP*n*	Access privileges for coprocessor *n*. The possible values of each field are: 0b00 = Access denied. Any attempted access generates a NOCP UsageFault. 0b01 = Privileged access only. An unprivileged access generates a NOCP fault. 0b10 = Reserved. The result of any access is Unpredictable. 0b11 = Full access.
[19:0]	-	Reserved. Read as Zero, Write Ignore.

CPACR 레지스터는 위와 같다. ORR로 CP10 과 CP11을 1로 셋팅해준다.
이 뜻은 부동 소수점을 유저와 커널 모두 허용한다는 뜻이다.

CPACR설정 완료후 다시 R0레지스터에 값을 넣고 SystemInit 함수로 진입한다.

```
/**
  * @brief  Micro Controller System을 설정한다.
  *         Embedded Flash Interface, PLL을 초기화하고 SystemFrequency 변수를 갱신한다.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
  /* RCC Clock 구성을 Default Reset State로 reset(재설정)한다. */
  /* Set HSION bit */
  RCC->CR |= (uint32_t)0x00000001;

  /* Reset CFGR register */
  RCC->CFGR = 0x00000000;

  /* Reset HSEON, CSSON and PLLON bits */
  RCC->CR &= (uint32_t)0xFEF6FFFF;

  /* Reset PLLCFGR register */
  RCC->PLLCFGR = 0x24003010;

  /* Reset HSEBYP bit */
  RCC->CR &= (uint32_t)0xFFFBFFFF;

  /* 모든 Interrupt를 비활성화한다. */
  RCC->CIR = 0x00000000;

#ifdef DATA_IN_ExtSRAM
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM */

  /* System Clock Source, PLL 곱셈기, 나눗셈기, AHB/APBx Prescalers와 Flash 설정을 구성한다. */
  SetSysClock();

  /* Offset Address를 더한 Vector Table 위치를 구성한다. */
#ifdef VECT_TAB_SRAM
  SCB->VTOR = SRAM_BASE | VECT_TAB_OFFSET; /* 내부 SRAM에 Vector Table 재배치 */
#else
  SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; /* 내부 FLASH(NAND)에 Vector Table 재배치 */
#endif
}
```

RCC 는 하드웨어 컨트롤 레지스터로 여러 하드웨어 레지스터들이 구조체 형태로 저장되어있다.
하드웨어 레지스터 CR을 살펴본다.

6.3.1 RCC clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		PLLSAIRDY	PLLSAION	PLLI2SRDY	PLLI2SON	PLLRDY	PLLON	Reserved				CSSON	HSEBYP	HSERDY	HSEON
		r	rw	r	rw	r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSIRDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

CR 레지스터는 클록 컨트롤 레지스터이다.

CR 과 0x1과 or 연산 즉 0번 비트를 1로 셋팅한다.
0번 비트를 살펴보면 HSION 이라 되어있다.

Bit 0 **HSION**: Internal high-speed clock enable
Set and cleared by software.
Set by hardware to force the HSI oscillator ON when leaving the Stop or Standby mode or in case of a failure of the HSE oscillator used directly or indirectly as the system clock. This bit cannot be cleared if the HSI is used directly or indirectly as the system clock.
0: HSI oscillator OFF
1: HSI oscillator ON

HSI는 오실레이터 이다. 오실레이터를 활성화 시킨다.
시스템 클록을 온 시킨다 되어있는데 .
default reset state를 만들기 위해 설정한 것이라는데 잘 모르겠다.

그다음 RCC -> CFGR을 살펴본다.
CFGR은 리셋벨류로 리셋 시켜놓았다 Reset value: 0x0000 0000
그다음 다시 CR = CR&0xFEF6FFFF
24 bit 16 bit 19 bit 를 0으로 셋팅한다. CR을 살펴본다.
즉, PLL, HSE , CSS를 리셋한다.

PLLCFGR 리셋벨류 로 리셋시킨다 0x2400 3010
다음 CR 의 HSEBYP를 리셋
그리고 CIR 인터럽트 레지스터를 초기화 시킨다
그다음 SetSysClock 함수로 넘어간다.

```
/**
  * @brief  System Clock Source, PLL 곱셈기 & 나눗셈기, AHB/APBx Prescalers와 Flash 설정을 구성한다.
  * @Note   이 함수는 RCC Clock 구성을 Default Reset State로 Reset하기 위해 단 한 번만 호출된다.
  * @param  None
  * @retval None
  */
static void SetSysClock(void)
{
/******************************************************************************/
/*            PLL (clocked by HSE)을 System Clock Source로 사용한다.          */
/******************************************************************************/
  __IO uint32_t StartUpCounter = 0, HSEStatus = 0;

  /* HSE를 활성화 */
  RCC->CR |= ((uint32_t)RCC_CR_HSEON);

  /* Time Out되서 종료되거나 HSE가 종료될때까지 대기한다. */
  do
  {
    HSEStatus = RCC->CR & RCC_CR_HSERDY;                    HSE_STARTUP_TIMEOUT의 값은 0x05000
    StartUpCounter++;
  } while((HSEStatus == 0) && (StartUpCounter != HSE_STARTUP_TIMEOUT));

  if ((RCC->CR & RCC_CR_HSERDY) != RESET)          RESET = 0
  {
    HSEStatus = (uint32_t)0x01;
  }
  else
  {
    HSEStatus = (uint32_t)0x00;
  }
```

CR 레지스터에서 HSE를 ON 시킨다 외부클럭을 받을 대기를 한다 HSERDY가 1되면 외부클럭을 받아들인다.
HSERDY 가 1이 되면 HSEStatus 가 1이 되어 다음 if문을 들어간다

```
  if (HSEStatus == (uint32_t)0x01)
  {
    /* High Performance Mode를 활성화하고, System Frequency를 168 MHz로 올린다. */
    RCC->APB1ENR |= RCC_APB1ENR_PWREN;
    PWR->CR |= PWR_CR_PMODE;          RCC->APB1ENR |= RCC_APB1ENR_PWREN;

    /* HCLK = SYSCLK / 1*/
    RCC->CFGR |= RCC_CFGR_HPRE_DIV1;

    /* PCLK2 = HCLK / 2*/
    RCC->CFGR |= RCC_CFGR_PPRE2_DIV2;

    /* PCLK1 = HCLK / 4*/
    RCC->CFGR |= RCC_CFGR_PPRE1_DIV4;

    /* main PLL을 구성한다. */
    RCC->PLLCFGR = PLL_M | (PLL_N << 6) | (((PLL_P >> 1) -1) << 16) |
                   (RCC_PLLCFGR_PLLSRC_HSE) | (PLL_Q << 24);

    /* main PLL을 활성화 */
    RCC->CR |= RCC_CR_PLLON;

    /* main PLL이 준비될때까지 대기한다. */
    while((RCC->CR & RCC_CR_PLLRDY) == 0)
    {
    }

    /* Flash Prefetch, Instruction Cache, Data Cache를 구성하고 대기 상태 */
    FLASH->ACR = FLASH_ACR_ICEN |FLASH_ACR_DCEN |FLASH_ACR_LATENCY_5WS;

    /* System Clock Source로 main PLL을 선택한다. */
    RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
    RCC->CFGR |= RCC_CFGR_SW_PLL;

    /* System Clock Source로 main PLL이 사용될때까지 대기한다. */
    while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS ) != RCC_CFGR_SWS_PLL);
    {
    }
  }
  else
  { /* HSE가 Start-Up에 실패하면 Application은 잘못된 Clock을 구성할 것이다.
       사용자(학생분들)가 이러한 오류를 다루기 위한 Code를 이곳에 추가하면 된다. */
  }
}
```

7.3.13 RCC APB1 peripheral clock enable register (RCC_APB1ENR)

Address offset: 0x40

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC EN	PWR EN	Reser-ved	CAN2 EN	CAN1 EN	Reser-ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	Reser-ved
		rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 28 **PWREN:** Power interface clock enable

Set and cleared by software.

0: Power interface clock disabled

1: Power interface clock enable

파워 인터페이스 클록을 셋한다.

VOS[1:0]: Regulator voltage scaling output selection

These bits control the main internal voltage regulator output voltage to achieve a trade-off between performance and power consumption when the device does not operate at the maximum frequency (refer to the STM32F42xx and STM32F43xx datasheets for more details).

These bits can be modified only when the PLL is OFF. The new value programmed is active only when the PLL is ON. When the PLL is OFF, the voltage scale 3 is automatically selected.

00: Reserved (Scale 3 mode selected)

01: Scale 3 mode

10: Scale 2 mode

11: Scale 1 mode (reset value)

- *PWR_CR_PMODE* changed to *PWR_CR_VOS*

PWR_CR_PMODE를 검색해보니 PWR_CR_VOS 로 바꾼다고 되어있다.

#define PWR_CR_PMODE PWR_CR_VOS

소스코드를 찾아보니 이렇게 정의 되어있는걸 확인하였다.

#define PWR_CR_VOS ((uint32_t)0x0000C000) /*!< VOS[1:0] bits (Regulator voltage scaling output selection) */

when the device does not operate at the maximum frequency

아무것도 정의가 되어있지 않았을 때 맥시멈으로 주파수로 설정한다 되어있으므로 168메가헤르즈이다.

HPRE: AHB prescaler

Set and cleared by software to control AHB clock division factor.

Caution: The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after HPRE write.

Caution: The AHB clock frequency must be at least 25 MHz when the Ethernet is used.

0xxx: system clock not divided

1000: system clock divided by 2

1001: system clock divided by 4

1010: system clock divided by 8

1011: system clock divided by 16

1100: system clock divided by 64

1101: system clock divided by 128

1110: system clock divided by 256

1111: system clock divided by 512

#define RCC_CFGR_HPRE_DIV1 ((uint32_t)0x00000000) /*!< SYSCLK not divided */

div1은 정의되어있지 않았기 때문에 나눠지지 않고 원래상태 그대로가 AHB로 들어간다.

PPRE1: APB Low speed prescaler (APB1)

Set and cleared by software to control APB low-speed clock division factor.

Caution: The software has to set these bits correctly not to exceed 45 MHz on this domain. The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after PPRE1 write.

0xx: AHB clock not divided

100: AHB clock divided by 2

101: AHB clock divided by 4

110: AHB clock divided by 8

111: AHB clock divided by 16

#define RCC_CFGR_PPRE1_DIV4 ((uint32_t)0x00001400) /*!< LCLK divided by 4 */

HCLK를 4로 나눠 APB1로 low-speed clock을 흘려보낸다

PPRE2: APB high-speed prescaler (APB2)

Set and cleared by software to control APB high-speed clock division factor.

Caution: The software has to set these bits correctly not to exceed 90 MHz on this domain. The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after PPRE2 write.

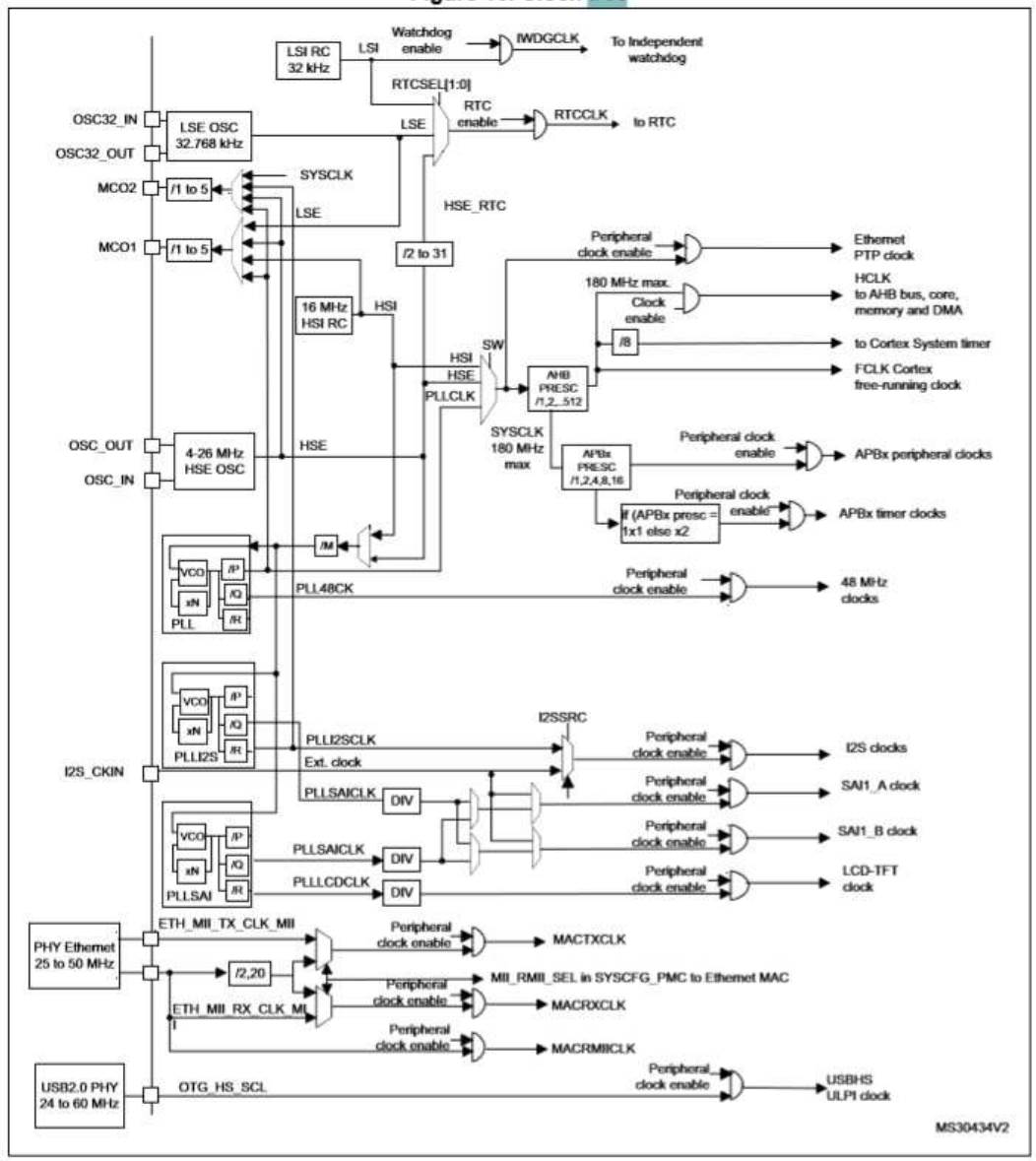
0xx: AHB clock not divided
100: AHB clock divided by 2
101: AHB clock divided by 4
110: AHB clock divided by 8
111: AHB clock divided by 16

#define RCC_CFGR_PPRE2_DIV2 ((uint32_t)0x00008000)
/*!< HCLK divided by 2 */

AHB clock divided by 2 이므로 APBx = HCLK를 2로 나눠 APB2로 high-speed clock 을 흘려보낸다

나머지도 이런식으로 따라가면 된다...
오르쪽 회로를 참고하여 prescaler를 확인한다.

Figure 16. Clock tree



systemcall.c

```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        int i;
        unsigned int test_arr[7]={0};

        register unsigned int *r0 asm("r0")=0;
        register unsigned int r1 asm("r1")=0;
        register unsigned int r2 asm("r2")=0;
        register unsigned int r3 asm("r3")=0;
        register unsigned int r4 asm("r4")=0;
        register unsigned int r5 asm("r5")=0;
        register unsigned int r6 asm("r6")=0;
        register int r7 asm("r7") = 0;

        r0 = test_arr;

        asm volatile("mov r1, #0x3\n"
                                "mov r2,r1,lsl#2\n"
                                "mov r4, #0x2\n"
                                "add r3,r1, r2,lsl r4\n"
                                "stmia r0!,{r1,r2,r3}\n"
                                "str r4, [r0]\n"
                                "mov r5, #128\n"
                                "stmia r0, {r4,r5,r6}\n"
                                "sub r0, r0, #12\n"
                                "ldmia r0, {r4,r5,r6}\n"
                                "swp r6, r3, [r0]\n");

        for(i=0; i<7;i++)
                printf("test_arr[%d]= %d\n", i,test_arr[i]);

        printf("r4 = %u, r5 = %u, r6=%u\n", r4, r5, r6);

        r7 = 2;

        asm volatile("swi #0" : "=r"(r0):"r"(r7):"memory");

        if(r0> 0)
                printf("r0= %p, Parent\n",r0);
        else if(r0==0)
                printf("r0=%p, Child\n", r0);
        else
        {
                perror("fork()");
                exit(-1);
        }
        return 0;
}
```

asm volatile("swi #0" : "=r"(r0):"r"(r7):"memory");
swi = software interrupt
명령어 : 출력 : 입력 : 특수동작

r7 = instruction schedule 방지
메모리를 보호한다. /*출력결과 부모 리턴값에 자식 피아이디 출력 */