TI DSP, MCU 및 Xilinx Zynq FPGA 프로그램 전문가 과정

Innova Lee(이상훈) 강사
gcccompil3r@gmail.com

은태영 학생
zero_bird@naver.com

Cortex - m4

```
; 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       .
```

Reset handler 동작을 확인할 수 있다.

사용하고자 하는 함수를 IMPORT 를 통해 불러온다.

우선 FPU settings 를 진행한다.

LDR R0, = 0xE000ED88

: R0 에 주소 0xE000ED88 을 저장한다.

: 0xE000ED88 을 살펴보면 CPACR 의 주소인 것을 알 수 있다.

CPACR 은 Coprocessor Access Control Register 이다.

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.

Cortex - m4

```
; 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       .
```

LDR R1, [R0]

: R0 에 들어있는 CPACR의 값을 R1에 저장한다.

ORR R1, R1, #(0xF << 20)

: 0000 0000 .... 1111 의 값을 왼쪽으로 20번 shift 한다.

: 0000 0000 1111 0000 .... 0000 으로 셋팅 된다.(23bit ~ 20bit)

: ORR 을 통하여 해당 bit 에 무조건 1111이 들어가도록 한다.

STR R1, [R0]

: 그 결과 값을 R0 의 메모리 주소에 저장한다.

LDR R0, = SystemInit

: R0 에 SystemInit 함수의 주소를 저장한다.

BLX R0

: 복귀할 주소를 저장하고 저장된 SystemInit 함수로 이동한다.

Cortex - m4

4.6.1 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:

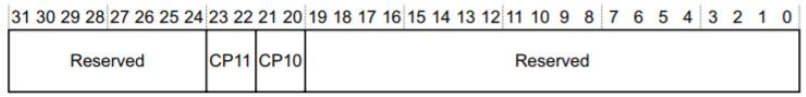

Table 4-50 CPACR register bit assignments

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 레지스터에 대한 설명을 볼 수 있다.

: Cortex M4 의 부동 소수점에 대한 설정임을 확인할 수 있다.

앞서 셋팅 된 값이 CP11 과 CP10 임을 알 수 있다.

CP11 과 CP10 에 각각 셋팅 된 값 11 의 정보를 확인할 수 있다.

0b11 = Full access.

즉, 유저와 커널 모두 사용하도록 값이 셋팅 된다.

Cortex - m4

```
/**
  * @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
}
```

SystemInit 내부를 살펴보도록 한다.

RCC 내부의 값들을 셋팅 하는 것을 볼 수 있다.

RCC->CR != (uint33_t)0x00000001;

RCC->CFGR = 0x00000000;

RCC->CR &= (unit_t)0xFFFBFFFF;

RCC->CIR = 0x00000000;

그 후, SetSysClock() 을 진행한다.

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

Cortex - m4

```
/**
  * @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;
  }
```

Cortex - m4

```
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. */
 }
}
```