


Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정


날 짜 : 2018 . 5. 2

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

MCU 들어가기.

BOOTLOADER_CODE_ CORETEX-M4(32BIT)

BOOT CODE

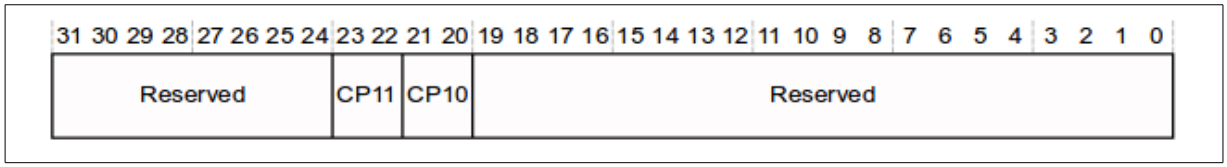
```
; Reset handler
Reset_Handler   PROC
                EXPORT  Reset_Handler             [WEAK]
        IMPORT  SystemInit
        IMPORT  __main

                ;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       .
```

1. Reset handler 를 초기화 해준다.

FPU(Floating point unit 부동소수점) SET (여러가지의 레지스터가 존재)

Address	Name	Type	Reset	Description
0xE000ED88	CPACR	RW	0x00000000	*Coprocessor Access Control Register*
0xE000EF34	FPCCR	RW	0xC0000000	*Floating-point Context Control Register* on page 4-49
0xE000EF38	FPCAR	RW	-	*Floating-point Context Address Register* on page 4-50
-	FPSCR	RW	-	*Floating-point Status Control Register* on page 4-50
0xE000EF3C	FPDSCR	RW	0x00000000	*Floating-point Default Status Control Register* on page 4-52

1. coretex 처음 시작시에 R0 에 **0x000ED88** 을 읽어온다. 위에 보면.. **CPACR** 을 가리킨다.
2. CPACR(Coprocessor Access Control Register)를 설정하게 된다.



3. 이게 cp11, cp10 에 대한 설명이다.

> 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.

4. **ORR R1, R1,#(0xF <<20)**
   이것을 해주면 0x00 F0 00 00 를 넣어준다. SET 함. F 가 오른 위치가 CP11 CP10 위치이다.
   (R1(0X000ED88) | 0x00 F0 00 00 을 원래 셋팅 값에 포함 시켜준다.)

5. **STR R1, [R0]**
   R1 을 R0 에 주소에 셋팅해준다.

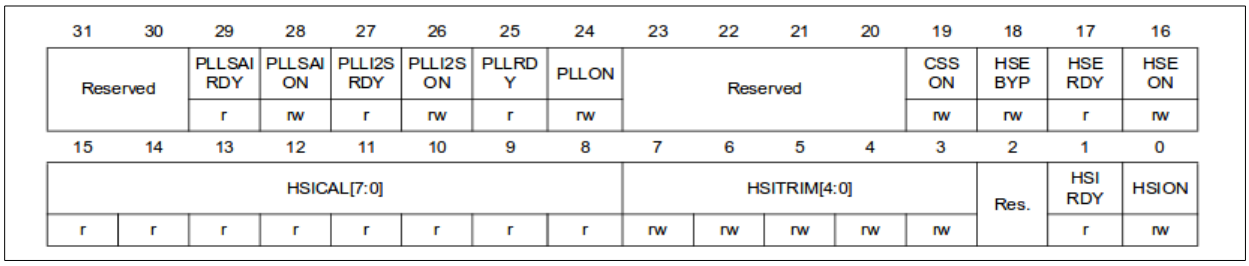
2. SystemInit

6. **LDR R0, = SystemInit**

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

7. **RCC→ CR | = (uint32_t)0x00 00 00 01;**
   데이터시트를 본다.(user manual)
   RCC, peripheral 은 mcu 에서 io 마다 어떤 mode 로 사용할지 주변회로를 관리하는 장치.
   RCC_CR 검색 Clock control register → 오실레이터를 켜줌.(HSION 를 SET)

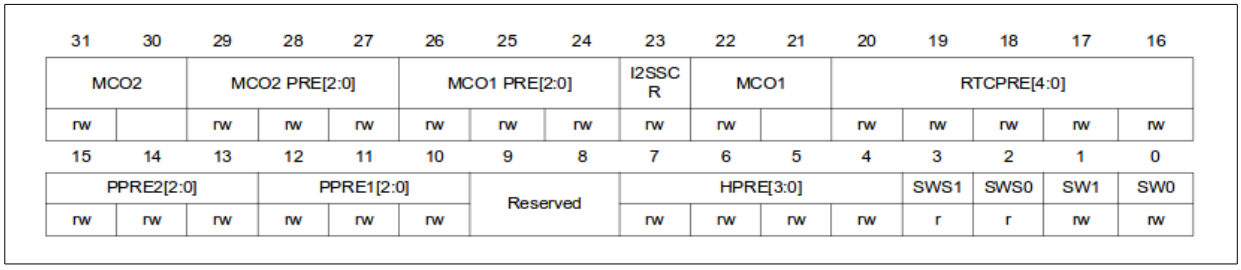
RCC→ CR | = (uint32_t)0x00 00 00 01; → 0 번 bit 를 활성화 시켜줌. (HISON)



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

8. **RCC→ CFGR = 0x00 00 00 00;**
   데이터 시트를 본다. → CFGR 레지스터의 값을 모두다 0 으로 초기화 해준다.
   (이유는 처음에 부팅되면서 플로팅 되거나 다른 값이 들어가 있을 수 있기에 해준다.)



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[4:0]				
rw		rw	rw	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
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

9. **RCC→CR & (uint32_t)0xFE F6 FF FF**
   위의 CR 레지스터를 보면 0xFE F6 FF FF 와 &를 시킨다.
   1111 1110 1111 0110 1111 1111 1111 1111 → 16, 19, 24 번 비트를 끔.
   (HSEON, CSSON, PLLON 을 0 으로 초기화 해준다.)

   HSEON = external clock (외부 클럭 오실레이터를 켜준다.)
   CSSON = clock security system 를 셋 해줌.

10. **RCC→ PLLCFGR = 0x24003010**
    PLLCFGR(phase lock loof) 위상 고정 루프의 상태값을 셋팅, 고주파이다.
    0010 0100 0000 0000 0011 0000 0001 0000 → 4, 12, 13, , 29

    PLLN3
    PLLN2
    PLLN1
    RESERVED

11. **RCC→ CR &= (uint32_t)= 0xFFFBFFFF;**
    18bit 를 0 으로 설정한다.
    HSEBYP: 내부 클럭의 바이패스를 0 으로 초기화 해준다.

12. **RCC→ CIR = 0x00 00 00 00;**
    Clock interrupt register 를 초기화 해준다. 왜냐고? 플로팅이나 똥값이 들어가서 set 되어 있을 수 있으니까!

3. SetSysClock

13. RCC→ CR | ((uint32_t)RCC_CR_HSEON);
HSE 외부 오실레이터를 사용. 한다.

14. Do{
HSEStatus = RCC→ CR & RCC_CR_HSERDY; // 클럭이 안정화 될때 까지 기다림
// 하드웨어에서 처리가 완료되면 RCC_CR_HSERDY bit 가 1 로 set 이 된다.
StartUpCounter++; // TIMEOUT 을 확인하기 위해서 count 를 해준다.
}while((HSEStatus ==0) && (StartUpCounter != HSE_STARTUP_TIMEOUT));
// 위의 조건이 클럭이 안정화 되고 일정 시간동안 클럭이 안정화가 안된다고 하면 루프를 나오게함.

15. if((RCC→ CR &RCC_CR_HSERDY)! = RESET) // RESET = 0
{
HSEStatus = (uint32_t)0x01; // 준비가 되면 다시 실행한다.
}

16. else
{
HSEStatus = (uint32_t)0x00; // 준비가 되지 않았다는 의미 이니 상태로 0 으로 만듬.
}

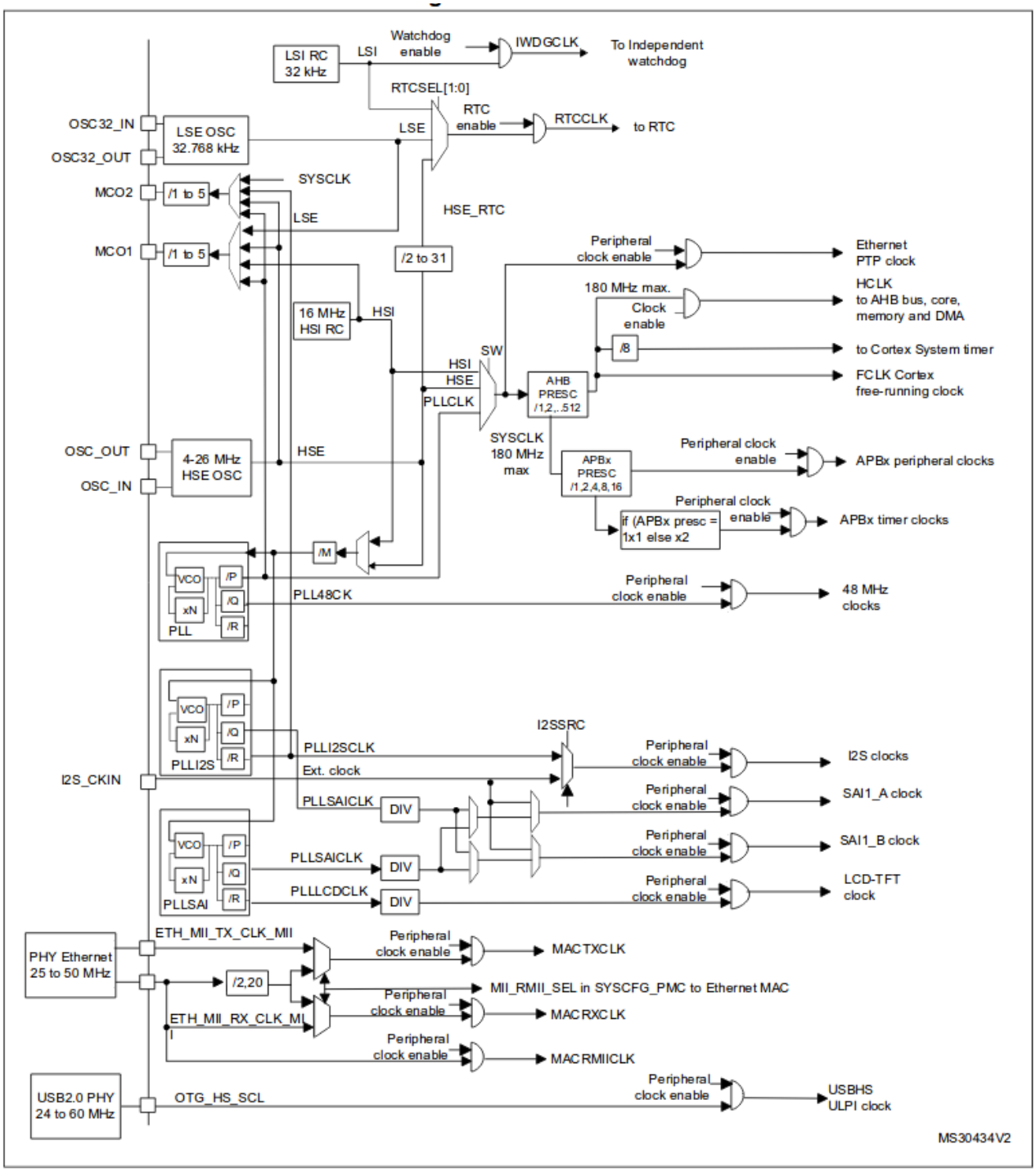
4. if(HSEStatus == (uint32_t)0x01) // 클럭이 안정되어 준비되면 실행되는 코드.

17. RCC→ APB1ENR |= RCC_APB1ENR_PWREN; // pheriheral 1 번 bus 가 활성화됨.
PWR→ CR |= PWR_CR_PMODE; // 파워를 pmode 로 설정 근데 안보임.
PWR_CR 은처음 전원 켜질 때 플로팅을 방지 하기 위한 녀석이다.

18. RCC→ CFGR |= RCC_CRGR_HPRE_DIV1;
0x00 00 00 00 | (00 00 00 00 ~ 00 00 00 07);

19. RCC→ CFGR |= RCC_CFGR_PPRE2_DIV2;
프리스케일, 분주비를 설정해준다.
여기서 , AHB (Advanced High-Performance Bus)
APB (Advanced Periheal Bus)
0x00 00 00 00 | 0x00 00 80 00

20. RCC→ CFGR |= RCC_CFGR_PPRE1_DIV4;
프리스케일, 분주비를 설정해준다. 분주비를 4 로 나누어줌.
0x00 00 00 00 | 0x00 00 14 00


LSI RC
32 kHz
LSI
Watchdog
enable
IWDGCLK
To Independent
watchdog
RTCSEL[1:0]
OSC32_IN
OSC32_OUT
LSE OSC
32.768 kHz
LSE
RTC
enable
RTCCLK
to RTC
MCO2
/1 to 5
SYSCLK
LSE
HSE_RTC
MCO1
/1 to 5
/2 to 31
Peripheral
clock enable
Ethernet
PTP clock
180 MHz max.
Clock
enable
HCLK
to AHB bus, core,
memory and DMA
16 MHz
HSI RC
HSI
/8
to Cortex System timer
SW
HSI
HSE
PLLCLK
AHB
PRESC
/1,2,..512
FCLK Cortex
free-running clock
SYSCLK
180 MHz
max
OSC_OUT
OSC_IN
4-26 MHz
HSE OSC
HSE
APBx
PRESC
/1,2,4,8,16
Peripheral clock
enable
APBx peripheral clocks
Peripheral clock
enable
if (APBx presc =
1x1 else x2
APBx timer clocks
/M
VCO
/P
/Q
xN
/R
PLL
PLL48CK
Peripheral
clock enable
48 MHz
clocks
PLLI2S
I2SSRC
PLLI2SCLK
Peripheral
clock enable
I2S clocks
I2S_CKIN
Ext. clock
PLLSAICLK
DIV
Peripheral
clock enable
SAI1_A clock
Peripheral
clock enable
SAI1_B clock
PLLSAI
PLLSAICLK
DIV
PLLLCDCLK
DIV
Peripheral
clock enable
LCD-TFT
clock
PHY Ethernet
25 to 50 MHz
ETH_MII_TX_CLK_MII
Peripheral
clock enable
MACTXCLK
/2,20
MII_RMII_SEL in SYSCFG_PMC to Ethernet MAC
Peripheral
clock enable
MACRXCLK
ETH_MII_RX_CLK_MI
I
Peripheral
clock enable
MACRMIICLK
USB2.0 PHY
24 to 60 MHz
OTG_HS_SCL
Peripheral
clock enable
USBHS
ULPI clock
MS30434V2

systemcall

```
#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 에서 sysfork 라는 명령어를 실행 하면 r0 에 는 그 결과값이(자식의 pid) 온다.

– memory 는 메모리를 보호한다. 인트스럭트 스케줄을 하지 말란 뜻인데 r7 의 값이 중도에 변할수 있어서 그렇다.