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

<펌웨어 프로그래밍>

2018.05.04 - 48일차

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

학생 – 안상재
sangjae2015@naver.com

<CORTEX-R5F 부트코드 분석>

1. 부트 코드 시작 부분

```
.sect ".intvecs"
    .arm

    .ref _c_int00
    .ref phantomInterrupt
    .def resetEntry

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]
```

2. _c_int00 함수

```
void _c_int00(void)
{
    _coreInitRegisters_();      // 레지스터 설정
    _coreInitStackPointer_();    // 스택 포인터 설정

    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();

main();

exit(0);

}
```

3. _c_int00 => _coreInitRegisters_()

```
_coreInitRegisters_
    ; After reset, the CPU is in the Supervisor mode (M = 10011)
        mov r0, lr
        mov r1, #0x0000    // Supervisor mode의 모든 레지스터를 0으로 초기화
        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       // special register의 값을 register로 가져옴
        msr spsr_cxsf, r1  /* register의 값을 special register로 가져옴
                              (cpsr값을 spsr_cxsf 값으로 가져옴) */
        ; Switch to FIQ mode (M = 10001)
        cps #17            // cpsr 의 mode bit에 10001를 셋팅해서 FIQ mode로 바꿈
        mov lr, r0         // 복귀 주소를 lr에 셋팅
        mov r8, #0x0000    // FIQ mode의 레지스터들을 0으로 초기화
        mov r9, #0x0000
        mov r10, #0x0000
        mov r11, #0x0000
        mov r12, #0x0000
        mrs r1, cpsr
        msr spsr_cxsf, r1    // 현재 cpsr 값을 spsr_cxsf 로 갱신
        ; Switch to IRQ mode (M = 10010)
        cps #18          // IRQ mode로 변환
        mov lr, r0      // 복귀 주소를 lr에 셋팅
        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    // 배정도 소수형 레지스터 초기화
        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                // 모든 레지스터 셋팅이 끝날 때 까지 대기함

    .endasmfunc
```

4. _c_int00 => _coreInitStackPointer_()

```
.def     _coreInitStackPointer_
    .asmfunc

_coreInitStackPointer_

        cps   #17
        ldr   sp,       fiqSp   // fiq mode 스택 포인터를 sp로 가져옴
        cps   #18
        ldr   sp,       irqSpq
        cps   #19
```

```
        ldr   sp,       svcSp
        cps   #23
        ldr   sp,       abortSp
        cps   #27
        ldr   sp,       undefSp
        cps   #31
        ldr   sp,       userSp
        bx    lr
// 각 mode에 해당하는 스택 포인터의 경계를 설정함
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
```

5. _c_int00 => getResetSource

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

    if ((SYS_EXCEPTION & (uint32)POWERON_RESET) != 0U)  // POWERON_REST = 0x8000U
    {
        /* 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)  // EXT_RESET = 0x0008U
    {
        /* Reset caused due to External reset. */
        rst_source = EXT_RESET;
        SYS_EXCEPTION = (uint32)EXT_RESET;
    }
    else if ((SYS_EXCEPTION & (uint32)DEBUG_RESET) !=0U)  // DEBUG_RESET = 0x0800U
    {
        /* Reset caused due Debug reset request */
        rst_source = DEBUG_RESET;
        SYS_EXCEPTION = (uint32)DEBUG_RESET;
    }
    else if ((SYS_EXCEPTION & (uint32)OSC_FAILURE_RESET) != 0U)// OSC_FAILURE_RESET = 0x4000U
    {
     /* 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)  // WATCHDOG_RESET = 0x2000U
    {
        /* Reset caused due watchdog violation */
        rst_source = WATCHDOG_RESET;
        SYS_EXCEPTION = (uint32)WATCHDOG_RESET;
    }
    else if ((SYS_EXCEPTION & (uint32)WATCHDOG2_RESET) !=0U)  // WATCHDOG2_RESET = 0x1000U
    {
        /* Reset caused due watchdog violation */
        rst_source = WATCHDOG2_RESET;
        SYS_EXCEPTION = (uint32)WATCHDOG2_RESET;
    }
    else if ((SYS_EXCEPTION & (uint32)CPU0_RESET) !=0U)  // CPU0_RESET = 0x0020U
    {
```

```
        /* 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)   // SW_RESET = 0x0010U
    {
        /* Reset caused due to software reset. */
        rst_source = SW_RESET;
        SYS_EXCEPTION = (uint32)SW_RESET;
    }
    else
    {
        /* No_reset occured. */
        rst_source = NO_RESET;      // NO_RESET = 0x0000U
    }

    return rst_source;
}
```

- SYSESR 레지스터(SYS_EXCEPTION) : 데이터 시트를 통해 각각의 bit 에 대한 정보를 확인할 수 있음.

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.

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

Bit	Field	Value	Description
5	CPURST		CPU reset flag. This bit is set when the CPU is reset. Write 1 will clear this bit. Write 0 has no effect.
			Note: A CPU reset can be initiated by the CPU self-test controller (LBIST) or by toggling the CPU RESET bit field in CPURSTCR register.
		0	No CPU reset has occurred.
		1	A CPU reset occurred.
4	SWRST		Software reset flag. This bit is set when a software system reset has occurred. Write 1 will clear this bit. Write 0 has no effect.
			Note: A software system reset can be initiated by writing to the RESET bits in the SYSECR register.
		0	No software reset has occurred.
		1	A software reset occurred.
3	EXTRST		External reset flag. This bit is set when a reset is caused by the external reset pin nRST or by any reset that also asserts the nRST pin (PORST, OSCRST, WDRST, WD2RST, and SWRST).
		0	The external reset pin has not asserted a reset.
		1	A reset has been caused by the external reset pin.
2-0	Reserved	0	Reads return 0. Writes have no effect.

6. _c_int00 => _memInit_()

```
.def _memInit_
       .asmfunc

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

        ldr   r11,  MSINENA      ;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  // Hardware initialization of all memory 가 끝날 때까지 대기
         beq   mloop

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

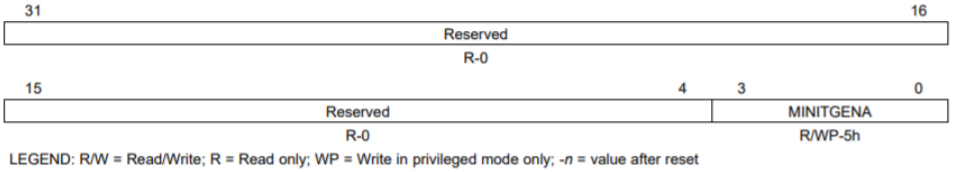
- MINITGCR 레지스터

MINITGCR 레지스터에 0xA 를 셋팅함으로써 Global memory hardware 초기화를 enable 시킴.

2.5.1.21 Memory Hardware Initialization Global Control Register (MINITGCR)

The MINITGCR register, shown in Figure 2-28 and described in Table 2-40, enables automatic hardware memory initialization.

Figure 2-28. Memory Hardware Initialization Global Control Register (MINITGCR) (offset = 5Ch)



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; *-n* = value after reset

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.

- MSINENA 레지스터

MSINENA 레지스터에 1을 셋팅함으로써 PBIST controllers for memory self test and the memory modules initialized during automatic hardware memory initialization 를 가능하게 함.

2.5.1.22 MBIST Controller/ Memory Initialization Enable Register (MSINENA)

The MSINENA register, shown in Figure 2-29 and described in Table 2-41, enables PBIST controllers for memory self test and the memory modules initialized during automatic hardware memory initialization.

Figure 2-29. MBIST Controller/Memory Initialization Enable Register (MSINENA) (offset = 60h)

31	16
MSIENA	
R/WP-0	

15	0
MSIENA	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

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.**

- MSTCGSTAT 레지스터

MSTCGSTAT 레지스터의 8번째 bit 를 계속 읽어서 1이 될 때까지 무한루프를 돔. MSTCGSTAT 레지스터의 8번째 bit 가 1이 되는 것은 Hardware initialization of all memory 이 완료되는 것이다.

2.5.1.23 MSTC Global Status Register (MSTCGSTAT)

The MSTCGSTAT register, shown in Figure 2-30 and described in Table 2-42, shows the status of the memory hardware initialization and the memory self-test.

Figure 2-30. MSTC Global Status Register (MSTCGSTAT) (offset = 68h)

31	16
Reserved	
R-0	

15	9	8	7	1	0
Reserved		MINIDONE	Reserved		MSTDONE
R-0		R/WPC-0	R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; WP = Write in privileged mode only; -*n* = value after reset

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.

7. _c_int00 => _coreEnableEventBusExport_()

```
_coreEnableEventBusExport_

        mrc   p15, #0x00, r0,         c9, c12, #0x00
        orr   r0,  r0,    #0x10
        mcr   p15, #0x00, r0,         c9, c12, #0x00
        bx    lr

    .endasmfunc
```

8. _c_int00 함수 안

```
if ((esmREG->SR1[2]) != 0U)   // error 가 발생했다면
        {
           esmGroup3Notification(esmREG,esmREG->SR1[2]); /* 에러가 없어질 때까지 무한루프를 돌며
                                                            대기 */
        }
```

```
void esmGroup3Notification(esmBASE_t *esm, uint32 channel)
{
    for(;;)
    {
    }
}
```

16.4.9 ESM Status Register 3 (ESMSR3)

This register is dedicated for Group3.

Figure 16-19. ESM Status Register 3 (ESMSR3) [offset = 20h]

31	16
ESF3[31:0]	
R/W1CP-X/0	

15	0
ESF3[15:0]	
R/W1CP-X/0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = value is unchanged

Table 16-11. ESM Status Register 3 (ESMSR3) Field Descriptions

Bit	Field	Value	Description
31-0	ESF3		Error Status Flag. Provides status information on a pending error. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: No error occurred. Write: Leaves the bit unchanged.
		1	Read: Error occurred. Write: Clears the bit.

9. _c_int00 => setupPLL(void)

```
void setupPLL(void)
{
    systemREG1->CSDISSET = 0x00000002U | 0x00000040U; // PLL1, PLL2 를 disabled 시킴

    while((systemREG1->CSDIS & 0x42U) != 0x42U) // PLL1, PLL2 가 disabled 될 때까지 대기
    {
    }

    /* Clear Global Status Register */
    systemREG1->GBLSTAT = 0x301U;
    // No PLL over cycle slip has been detected.
    // No PLL under cycle slip has been detected.
    // No oscillator failure has been detected.
```

```
    /** - Configure PLL control registers */
    /** @b Initialize @b Pll1: */

    /**   - Setup pll control register 1:
    *     - Setup reset on oscillator slip
    *     - Setup bypass on pll slip
    *     - setup Pll output clock divider to max before Lock
    *     - Setup reset on oscillator fail
    *     - Setup reference clock divider
    *     - Setup Pll multiplier
    */
    systemREG1->PLLCTL1 =  (uint32)0x00000000U
                         |  (uint32)0x20000000U  // Bypass on PLL Slip is disabled
                         |  (uint32)((uint32)0x1FU << 24U)   // f PLL CLK= f post-ODCLK / 32
                         |  (uint32)0x00000000U
                         |  (uint32)((uint32)(8U - 1U)<< 16U)    // f INT CLK = f OSCIN / 8
                         |  (uint32)(0x9500U);   // f VCO CLK = f INT CLK * 150

    /**   - Setup pll control register 2
    *     - Setup spreading rate
    *     - Setup bandwidth adjustment
    *     - Setup internal Pll output divider
    *     - Setup spreading amount
    */
    systemREG1->PLLCTL2 =  (uint32)((uint32)255U << 22U)    // f mod= f s= f INT CLK / (2 ×
(255+1))
                         |  (uint32)((uint32)7U << 12U)      // MUL_when_MOD = 7/256
                         |  (uint32)((uint32)(1U - 1U) << 9U)    // f post-ODCLK= f VCO CLK /
1
                         |  (uint32)61U; // SPR_AMOUNT : NV = 62/2048

    /** @b Initialize @b Pll2: */

    /**   - Setup pll2 control register :
    *     - setup Pll output clock divider to max before Lock
    *     - Setup reference clock divider
    *     - Setup internal Pll output divider
    *     - Setup Pll multiplier
    */
    systemREG2->PLLCTL3 = (uint32)((uint32)(1U - 1U) << 29U)    // fpost_ODCLK2 = fout-
put_CLK2 / 1
                        | (uint32)((uint32)0x1FU << 24U)        // fPLL2 CLK = fpost_ODCLK2 /
32
                        | (uint32)((uint32)(8U - 1U)<< 16U)     // fINTCLK2 = fOSCIN / 8
                        | (uint32)(0x9500U);            // fVCOCLK2 = fINTCLK2 x (149+1)

    /** - Enable PLL(s) to start up or Lock */
    systemREG1->CSDIS = 0x00000000U
                      | 0x00000000U
                      | 0x00000008U     // EXTCLKIN disable
                      | 0x00000080U     // EXTCLKIN2 disable
                      | 0x00000000U
                      | 0x00000000U
                      | 0x00000000U
                      | 0x00000004U;    // reserved
}
```

10. _c_int00 => periphInit(void)

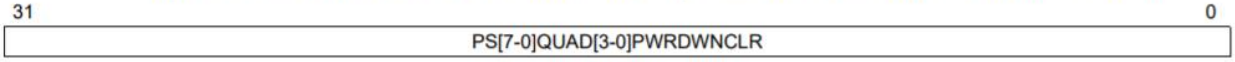
```
void periphInit(void)
{
    /** - Disable Peripherals before peripheral powerup*/
    systemREG1->CLKCNTL &= 0xFFFFFEFFU; // peripheral/peripheral memory 리셋중

    /** - Release peripherals from reset and enable clocks to all peripherals */
    /** - Power-up all peripherals */
    pcrREG1->PSPWRDWNCLR0 = 0xFFFFFFFFU;
    pcrREG1->PSPWRDWNCLR1 = 0xFFFFFFFFU;
    pcrREG1->PSPWRDWNCLR2 = 0xFFFFFFFFU;
    pcrREG1->PSPWRDWNCLR3 = 0xFFFFFFFFU;

    pcrREG2->PSPWRDWNCLR0 = 0xFFFFFFFFU;
    pcrREG2->PSPWRDWNCLR1 = 0xFFFFFFFFU;
    pcrREG2->PSPWRDWNCLR2 = 0xFFFFFFFFU;
    pcrREG2->PSPWRDWNCLR3 = 0xFFFFFFFFU;

    pcrREG3->PSPWRDWNCLR0 = 0xFFFFFFFFU;
    pcrREG3->PSPWRDWNCLR1 = 0xFFFFFFFFU;
    pcrREG3->PSPWRDWNCLR2 = 0xFFFFFFFFU;
    pcrREG3->PSPWRDWNCLR3 = 0xFFFFFFFFU;

    /** - Enable Peripherals */
    systemREG1->CLKCNTL |= 0x00000100U;    // 리셋 종료 (초기화 완료)
}
```

2.5.3.21 Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0)

There is one bit for each quadrant for PS0 to PS7. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-92 and described in Table 2-106.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to unimplemented bits have no effect and reads are 0.

Figure 2-92. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) (offset = A0h)



LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 2-106. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0] PWRDWNCLR	0	Peripheral select quadrant clock power-down clear. *Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is cleared to 0.