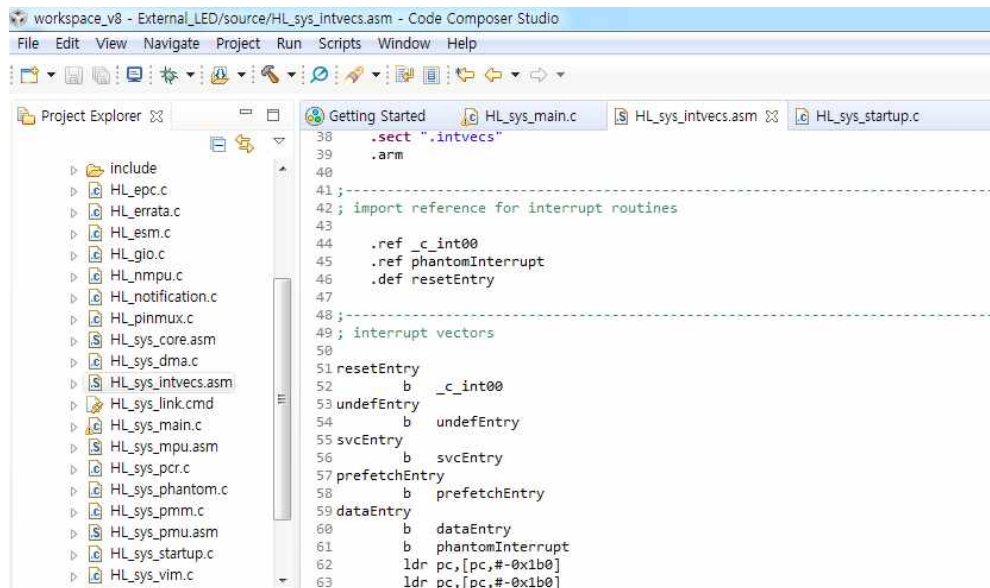


# TI DSP, MCU, Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee (이상훈)  
[gcccompil3r@gmail.com](mailto:gcccompil3r@gmail.com)  
학생 – 김형주  
[mihaelkel@naver.com](mailto:mihaelkel@naver.com)

## Cortex-R5F 부트로더 분석



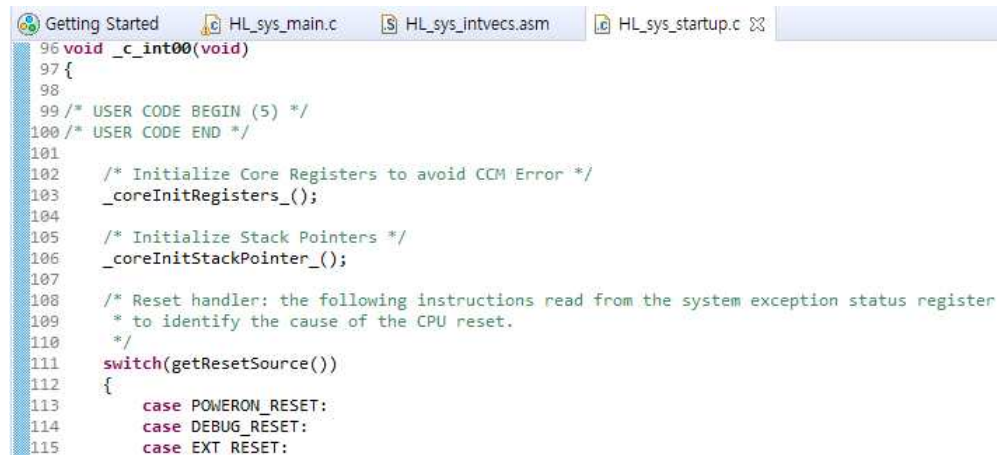
The screenshot shows the Code Composer Studio interface. The Project Explorer on the left lists various files under the 'include' directory, including HL\_epc.c, HL\_errata.c, HL\_esm.c, HL\_glo.c, HL\_nmpu.c, HL\_notification.c, HL\_pinmux.c, HL\_sys\_core.asm, HL\_sys\_dma.c, HL\_sys\_intvecs.asm (selected), HL\_sys\_link.cmd, HL\_sys\_main.c, HL\_sys\_mpu.asm, HL\_sys\_pcr.c, HL\_sys\_phantom.c, HL\_sys\_pmm.c, HL\_sys\_pmu.asm, HL\_sys\_startup.c, and HL\_sys\_vim.c. The main editor displays the contents of HL\_sys\_intvecs.asm, which defines interrupt vectors and routines. The code includes comments about import reference for interrupt routines and defines resetEntry, undefEntry, svcEntry, prefetchEntry, and dataEntry.

부트로더는 "HL\_sys\_intvecs.asm" 파일 안에 작성되어 있다.

코드를 보면, b(branch)명령어를 통해 \_c\_int00로 점프를 하고 있다.

F3을 누르면 해당 함수의 정의부로 갈 수 있고, F2를 누르면 정의를 볼 수 있다.

F3을 눌러 \_c\_int00가 뭔지 알아보자.

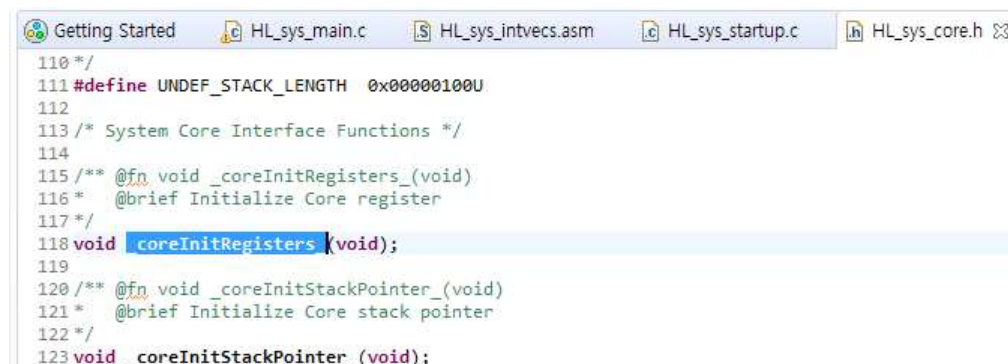


The screenshot shows the Code Composer Studio interface with the HL\_sys\_intvecs.asm file open. The code defines the \_c\_int00 function, which is a reset handler. It includes comments about initializing core registers and stack pointers, and a switch statement that handles different reset sources: POWERON\_RESET, DEBUG\_RESET, and EXT\_RESET. The function calls \_coreInitRegisters\_() and \_coreInitStackPointer\_() to initialize the system.

함수 안에 함수가 또 있다.

하나씩 알아보자. 먼저 \_coreInitRegisters\_()가 어떻게 정의되어 있는지,

F3을 눌러서 다시 들어간다.



The screenshot shows the Code Composer Studio interface with the HL\_sys\_core.h file open. The code defines the \_coreInitRegisters\_ function, which is a system core interface function. It includes comments about initializing core registers and stack pointers, and defines the function signature: void \_coreInitRegisters\_(void). The function is also defined in the same file, calling \_coreInitStackPointer\_() to initialize the stack pointer.

헤더파일, 즉 선언부밖에 보이지 않고, 정의는 안보인다.

정의는 다른 파일에 있다고 추측할 수 있는데, 함수 이름을 잘 기억해두고

include된 모든 파일들을 살펴보자

"HL\_sys\_core.asm" 라는 어셈블리 파일이 눈에 띈다.

해당 파일을 열어 coreInitRegisters를 검색하면 아래와 같이 정의를 볼 수 있다.

```
Getting Started | HL_sys_core.asm | HL_sys_main.c | HL_sys_intvecs.asm
51 _coreInitRegisters_
52
53 ; After reset, the CPU is in the Supervisor mode (M = 10011)
54 mov r0, lr
55 mov r1, #0x0000
56 mov r2, #0x0000
57 mov r3, #0x0000
58 mov r4, #0x0000
59 mov r5, #0x0000
60 mov r6, #0x0000
61 mov r7, #0x0000
62 mov r8, #0x0000
63 mov r9, #0x0000
64 mov r10, #0x0000
65 mov r11, #0x0000
66 mov r12, #0x0000
67 mov r13, #0x0000
68 mrs r1, cpsr
69 msr spsr_cxsf, r1
```

mov 명령어가 여러 개 보인다.

mov r0, lr :

lr에 복귀주소(return address)가 저장되어 있으므로,

복귀주소를 r0 레지스터에 저장하는 코드라는 것을 알 수 있다.

mov r(x), #0x0000(x 1:13) :

각 범용 레지스터(general purpose registers)들을 0으로 초기화하는 코드이다.

mrs r1, cpsr :

cpsr 레지스터에 있는 정보를 r1으로 읽어온다.

msr spsr\_cxsf, r1 :

r1에 저장되어 있는 정보를 spsr\_cxsf 레지스터에 저장한다.

※psr(cpsr, spsr)

### 3.7 Program status registers

The processor contains one CPSR and five SPSRs for exception handlers to use. The program status registers:

- hold information about the most recently performed ALU operation
- control the enabling and disabling of interrupts
- set the processor operating mode.

Figure 3-4 shows the bit arrangement in the status registers.

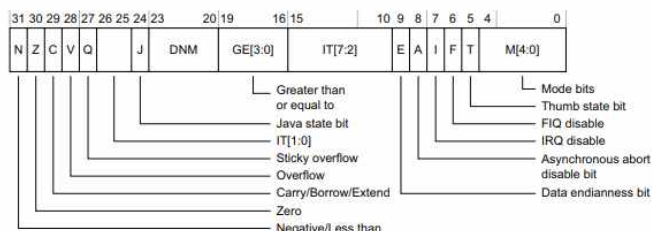


Figure 3-4 Program status register

데이터시트를 찾아보면, cpsr은 Current Program Status Register를 의미한다.

프로그램의 상태를 저장하는 레지스터임을 알 수 있다. spsr은 각 모드마다 1개씩 존재한다.

spsr이 각 모드마다 존재하므로, 모드가 바뀔시 이전에 저장되었던 context를 따로 불러올 필요가 없어진다.

x86의 경우 context switching시 현재 context의 정보를 별도의 stack에 보관하고, return시 stack에 있는 context를 다시 불러왔기 때문에 불필요한 연산, 즉, overhead가 발생했다. ARM에서는 mode별 context를 고유의 spsr에 저장하기 때문에, context swtiching을 overhead 없이 빠르게 처리할 수 있게 한다.

General registers and program counter

System and User	FIQ	Supervisor	Abort	IRQ	Undefined
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)

Program status registers

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und	

▲ = banked register

데이터시트를 더 찾아보면, 각 mode별로 범용 레지스터 또한 banked register로 가지고 있는 것을 알 수 있다.

```

Getting Started | HL_sys_core.asm | HL_sys_main.c | HL_sys_int
70 ; Switch to FIQ mode (M = 10001)
71 cps #17
72 mov lr, r0
73 mov r8, #0x0000
74 mov r9, #0x0000
75 mov r10, #0x0000
76 mov r11, #0x0000
77 mov r12, #0x0000
78 mrs r1, cpsr
79 msr spsr_cxsf, r1
80 ; Switch to IRQ mode (M = 10010)
81 cps #18
82 mov lr, r0
83 mrs r1, cpsr
84 msr spsr_cxsf, r1
85 ; Switch to Abort mode (M = 10111)
86 cps #23
87 mov lr, r0
88 mrs r1, cpsr
89 msr spsr_cxsf, r1
90 ; Switch to Undefined Instruction Mode (M = 11011)
91 cps #27
92 mov lr, r0
93 mrs r1, cpsr
94 msr spsr_cxsf, r1

```

다음 코드를 보면 code change를 하고 각 모드별 context를 초기화 하는 것을 할 수 있다.

위의 레지스터 정보를 보면, r0는 banked register가 아니기 때문에 Supervisor mode에서 연산했던 r0의 정보를 그대로 가지고 있을 것이다. 즉, r0에는 return address가 저장되어 있을 것이다.

cps #17 :

mode를 17(0b10001)로 바꾼다. mode bit는 cpsr에 있으므로, cpsr의 mode bit를 변경하는 명령어가 된다.

Table 3-3 PSR ode bit values

M[4:0]	Mode
b10000	User
b10001	FIQ
b10010	IRQ

Table 3-3 PSR ode bit values (continued)

M[4:0]	Mode
b10011	Supervisor
b10111	Abort
b11011	Undefined
b11111	System

datasheet를 보면, 0b10001은 FIQ mode라는 것을 알 수 있다.

이 과정을 반복하여, System mode의 context까지 모두 초기화해준다.

```

102 mrc p15, #0x00, r2, c1, c0, #0x02
103 orr r2, r2, #0xF00000
104 mcr p15, #0x00, r2, c1, c0, #0x02
105 mov r2, #0x40000000
106 fmxr fpexc, r2

```

다음 코드를 보면, mrc 명령어가 보인다.

To access the CPACR, read or write CP15 with:

MRC p15, 0, <Rd>, c1, c0, 2 ; Read CPACR

mrc p15, #0x00, r2, c1, c0, #0x02 :

datasheet를 검색해보면, 이 코드는 CPACR값을 읽어서 r2에 저장한다는 것을 알 수 있다.

orr r2, r2, #0xF00000 :

읽어온 CPACR값에 0x00F0 0000값을 or하므로, 23:20번 bit를 1로 set한다는 것을 알 수 있다.

mcr p15, #0x00, r2, c1, c0, #0x02 :

r2 레지스터에 저장된 값을 CPACR 레지스터에 저장한다.

즉, 3줄의 코드는 CPACR 레지스터의 23:20번 bit를 1로 set하는 코드라 할 수 있다.

CPACR을 검색해보자.

[23:22]	cp11	Defines access permissions for the FPU. If the FPU is not included for this processor, these bits are RAZ/WI.
[21:22]	cp10	If the FPU is included, both cp10 and cp11 must be programmed to the same value: b00 = Access denied. Attempts to access generates an Undefined Instruction exception. This is the reset value. b01 = Privileged mode access only b10 = Reserved b11 = Privileged and User mode access.

23:22 번 bit는 FPU에 대한 접근 허가를 정의한다고 써있다.

21:22 번 bit라고 써있는데, 오타인 듯 하다. 21:20으로 정정하자.

21:20 bit는 값에 따라 권한 설정을 한다는 것을 알 수 있다.

우리는 21:20 = 11 로 set했기 때문에, b11을 보자.

User mode와 Privileged mode 둘 다 접근을 허가한다고 써있다.

즉, 위의 3줄의 코드는 최종적으로 FPU를 정의하고 FPU에 대한 접근 권한을 설정하는 코드이다.

mov r2, #0x40000000

fmxr fpexc, r2 :

위 2줄은 결국 fpexc에 0x40000000을 저장하는 코드가 된다.

즉, fpexc 레지스터의 30번 bit를 1로 set해준다.

Bits	Name	Function
[31]	-	RAZ.
[30]	EN	VFP enable bit. Setting EN enables VFP functionality. Reset clears EN.
[29]	DEX	Set when an Undefined Instruction exception is taken because of a vector instruction that would have been executed if the processor supported vectors. This field is cleared when an Undefined Instruction exception is taken for any other reason. Resets to zero. In single-precision only configurations, this bit is not set for any double-precision operations, whether they are vector operations or not.
[28:0]	-	RAZ.

30번 bit가 set되면, VFP를 사용할 수 있게 하고 다시 0으로 clear된다고 써있다.

#### 11.2.1 FPU views of the register bank

In the FPU, you can view the register bank as:

- Sixteen 64-bit doubleword registers, D0-D15.
- Thirty-two 32-bit single-word registers, S0-S31.
- A combination of registers from these views.

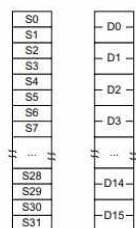


Figure 11-1 FPU register bank

The mapping between the registers is as follows:

- $S\langle 2n \rangle$  maps to the least significant half of  $D\langle n \rangle$ .
- $S\langle 2n+1 \rangle$  maps to the most significant half of  $D\langle n \rangle$ .

For example, you can access the least significant half of the value in D6 by accessing S12, and the most significant half of the elements by accessing S13.

FPU에 대한 설명을 읽어보면, D0~ D15까지 64bit의 레지스터 16개를 가지고 있음을 알 수 있다.

64bit이므로, 초기화를 하기 위해서는 32bit가 2개 필요하므로, 초기화는 아래와 같이 한다.

```

108      fmdrr d0,      r1,  r1
109      fmdrr d1,      r1,  r1
110      fmdrr d2,      r1,  r1
111      fmdrr d3,      r1,  r1
112      fmdrr d4,      r1,  r1
113      fmdrr d5,      r1,  r1
114      fmdrr d6,      r1,  r1
115      fmdrr d7,      r1,  r1
116      fmdrr d8,      r1,  r1
117      fmdrr d9,      r1,  r1
118      fmdrr d10,     r1,  r1
119      fmdrr d11,     r1,  r1
120      fmdrr d12,     r1,  r1
121      fmdrr d13,     r1,  r1
122      fmdrr d14,     r1,  r1
123      fmdrr d15,     r1,  r1

```

이 뒤에 의미없어 보이는 코드들이 있는데, d0-d15를 초기화 하는 도중 b(branch)를 하게 되면 pipeline이 깨지게 되므로 위 명령어들이 완료되는 동안 clock을 지연시키는 코드로 봐도 무방하다.



```

146 _coreInitStackPointer_
147
148     cps    #17
149     ldr    sp,    fiqSp
150     cps    #18
151     ldr    sp,    irqSp
152     cps    #19
153     ldr    sp,    svcSp
154     cps    #23
155     ldr    sp,    abortSp
156     cps    #27
157     ldr    sp,    undefSp
158     cps    #31
159     ldr    sp,    userSp
160     bx     lr
161
162 userSp .word 0x08000000+0x00001000
163 svcSp  .word 0x08000000+0x00001000+0x00000100
164 fiqSp  .word 0x08000000+0x00001000+0x00000100+0x00000100
165 irqSp  .word 0x08000000+0x00001000+0x00000100+0x00000100+0x00000100
166 abortSp .word 0x08000000+0x00001000+0x00000100+0x00000100+0x00000100+0x00000100
167 undefSp .word 0x08000000+0x00001000+0x00000100+0x00000100+0x00000100+0x00000100+0x00000100
168
169 .endasmfunc

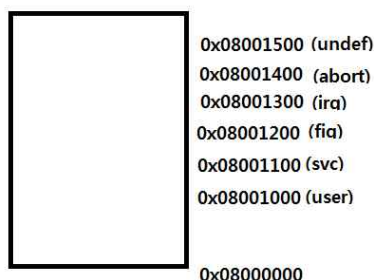
```

레지스터들을 초기화 한 후, Stack Pointer를 초기화하는 함수이다.

cps는 mode를 변경하는 명령어다.

sp 레지스터는 r14레지스터를 말하고,

r14레지스터는 Banked Register이므로 mode마다 각각 고유의 register를 가진다.



위와 같이 sp를 설정하는 함수가 된다. 실제 stack이 할당 되는 것은 아직 아니다.

예를 들면, user mode에서 stack을 사용시, 0x08001000을 기준으로 아래로 stack이 자라게 되고,

abort mode에서 stack을 사용시, 0x08001400을 기준으로 아래로 stack이 자란다.

```

111 switch(getResetSource())
112 {
113     case POWERON_RESET:
114     case DEBUG_RESET:
115     case EXT_RESET:

```

return하고나면, getResetSource()를 호출하는 것을 볼 수 있다.

```

447 resetSource_t getResetSource(void)
448 {
449     register resetSource_t rst_source;
450
451     if ((SYS_EXCEPTION & (uint32)POWERON_RESET) != 0U)
452     {
453         /* power-on reset condition */
454         rst_source = POWERON_RESET;
455
456         /* Clear all exception status Flag and proceed since it's power up */
457         SYS_EXCEPTION = 0x0000FFFFU;
458     }
459     else if ((SYS_EXCEPTION & (uint32)EXT_RESET) != 0U)
460     {
461         /* Reset caused due to External reset. */
462         rst_source = EXT_RESET;
463         SYS_EXCEPTION = (uint32)EXT_RESET;
464     }
465     else if ((SYS_EXCEPTION & (uint32)DEBUG_RESET) != 0U)
466     {
467         /* Reset caused due Debug reset request */
468         rst_source = DEBUG_RESET;
469         SYS_EXCEPTION = (uint32)DEBUG_RESET;
470     }

```

getResetSource()안에는 if문과 else if로 이루어진 switch ~ case: 문이 있고, rst\_source에 값을 넣어 리턴하는 형태이다.

```

124 #define SYS_EXCEPTION (*(volatile uint32 *)0xFFFFFE4U)

```

SYS\_EXCEPTION은 0xFFFFFFE4로 정의되어 있고, E4번 offset이 어떤 System Control Register인지 찾아보면,

E4h SYSESR System Exception Status Register Section 2.5.1.46

SYSESR 레지스터임을 알 수 있다.

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	0 1	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	0 1	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. <b>Note: The action taken when an oscillator failure or PLL slip is detected must be configured in the PLLCTL1 register.</b> 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	0 1	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	DBG_RST	0 1	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	0 1	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.

SYSESR register의 description은 위와 같다. 필드 이름을 보니, Power,Ocillator,Watchdog,Debug 등이 reset이 될 때, 해당 비트가 set되는 register임을 알 수 있다.

분석하고 있는 것은 부트코드 이므로, PORST(15번 bit)가 1로 set되어있을 것이다. 코드로 돌아와서

```
POWERON_RESET = 0x8000U, /**< Alias for Power On Reset */
```

0x80000이므로, POWERON\_RESET은 15번bit를 의미하는 것을 알 수 있다.

```
451 if ((SYS_EXCEPTION & (uint32)POWERON_RESET) != 0U)
452 {
453     /* power-on reset condition */
454     rst_source = POWERON_RESET;
455
456     /* Clear all exception status Flag and proceed since it's power up */
457     SYS_EXCEPTION = 0x0000FFFFU;
458 }
```

if문들 중, 위의 code가 실행될 것이고,

rst\_source = 0x80000, SYS\_EXCEPTION=0x0000FFFF;

가 저장되고, rst\_source가 return될 것이다.

```
111 switch(getResetSource())
112 {
113     case POWERON_RESET:
114     case DEBUG_RESET:
115     case EXT_RESET:
116
117 /* USER CODE BEGIN (6) */
118 /* USER CODE END */
119
120     /* Initialize L2RAM to avoid ECC errors right after power on */
121     _memInit();
```

getResetSource()의 return값이 POWERON\_RESET이므로, 제일 위쪽 코드부터 실행될 것이다.

제일 처음 보이는 함수는 \_memInit();이므로 해당 함수를 분석해보자

```
Getting Started | HL_sys_core.asm | HL_sys_main.c | HL_sys_intvecs.asm | HL_
596 _memInit
597     ldr r12, MINITGCR      ;Load MINITGCR register address
598     mov r4, #0xA         ;Enable global memory hardware initialization
599     str r4, [r12]
600
601     ldr r11, MSIEA        ;Load MSIEA register address
602     mov r4, #0x1         ;Bit position 0 of MSIEA corresponds to SRAM
603     str r4, [r11]        ;Enable auto hardware initialisation for SRAM
```

위 3줄은 MINITGCR 레지스터의 값을 r12로 가져오고,

1,3번 비트를 1로 set한 후 다시 MINITGCR 레지스터에 값을 넣는 것을 알 수 있다.

MINITGCR이 뭔지 검색해보자.

```
91 uint32 MINITGCR; /* 0x005C */
```

offset은 5c이고,

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	Ah Others	Memory hardware initialization global enable key. Global memory hardware initialization is enabled. Global memory hardware initialization is disabled. <b>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.</b>

위 코드에서 MINITGCR에 0xA를 set해주었으므로, 메모리 하드웨어 초기화가 Enable 될 것이다.

```

601      ldr    r11, MSIENA      ;Load MSIENA register address
602      mov    r4, #0x1        ;Bit position 0 of MSIENA corresponds to SRAM
603      str    r4, [r11]       ;Enable auto hardware initialisation for SRAM

```

다음 코드도 보니, MSIENA 레지스터의 0번 bit를 1로 set해준다는 것을 알 수 있다.

MSIENA를 찾아보자.

```

92      uint32 MSINENA;        /* 0x0060 */

```

MSINENA의 오타인 듯 하다. MSIENA는 코드 내에서 정의를 찾을 수가 없다. 왜 정상동작하는지는 모르겠다.

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

Bit	Field	Value	Description
31-0	MSIENA		<p>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 a Ah to any other value) will reset all the MSIENA[31-0] bits to their default values.</p> <p><b>0</b></p> <p><i>In memory self-test mode (MSTGENA = Ah):</i> PBIST controller [31-0] is disabled.</p> <p><i>In memory initialization mode (MINITGENA = Ah):</i> Memory module [31-0] auto hardware initialization is disabled.</p> <p><b>1</b></p> <p><i>In memory self-test mode (MSTGENA = Ah):</i> PBIST controller [31-0] is enabled.</p> <p><i>In memory initialization mode (MINITGENA = Ah):</i> Memory module [31-0] auto hardware initialization is enabled.</p> <p><b>Note:</b> 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.</p>

0번 bit가 1로 Set되었고, 이 전에 MINITGENA 레지스터에 A값을 Set해줬으므로, Memory module[0]의 auto hardware initialization이 enable된다. 주의사항도 한 번 읽어보면, 위의 두 옵션(self-test, initialization)이 동시에 일어나지 않도록 프로그램하라고 써있다.

```

604 mloop
605     ldr    r5, MSTCGSTAT
606     ldr    r4, [r5]
607     tst    r4, #0x100
608     beq    mloop

```

1 instruction만큼 덜 최적화된 코드이긴 하지만, MSTCGSTAT 레지스터 값을 읽어와서, 8번째 bit가 1이 될 때까지, Loop를 도는 형태로 되어있다. MSTCGSTAT의 8번째 bit가 뭔지 알아보자

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

8번의 read값이 1이라는 것은, 하드웨어의 모든 메모리 초기화가 완료되었다는 뜻이다.

즉, 위의 loop문은 하드웨어가 초기화될 때까지 대기하는 코드가 된다.

```

610      mov    r4, #5
611      str    r4, [r12]        ;Disable global memory hardware initialization
612      bx     lr

```

다음 코드는 r12, 즉 MINITGCR 레지스터에 5값(0b0101)을 set 하고 복귀주소로 return하는 코드이다.

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		<p>Memory hardware initialization global enable key.</p> <p>Ah</p> <p>Global memory hardware initialization is enabled.</p> <p>Others</p> <p>Global memory hardware initialization is disabled.</p> <p><b>Note:</b> 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.</p>

0b0101은 0xA가 아니므로, 하드웨어 초기화를 disable해주는 코드가 된다. 한 번 초기화를 했으니, 다시 초기화할 필요가 없기 때문에 초기화 옵션을 꺼준다.



```

234 .def _coreEnableEventBusExport_
235 .asmfunc
236
237 _coreEnableEventBusExport_
238
239 mrc p15, #0x00, r0, c9, c12, #0x00
240 orr r0, r0, #0x10
241 mcr p15, #0x00, r0, c9, c12, #0x00
242 bx lr
243
244 .endasmfunc

```

다음 함수는 \_coreEnableEventBusExport\_(); 다.

명령어 4개짜리 함수인데, c9, c12, 0에 해당하는 CoProcessor의 4번 bit에 1을 set해주는 코드가 된다. c9, c12, 0을 검색해보자

The PMCR Register is always accessible in Privileged mode. To access the register, read or write CP15 with:

```

MRC p15, 0, <Rd>, c9, c12, 0 ; Read PMCR Register
MCR p15, 0, <Rd>, c9, c12, 0 ; Write PMCR Register

```

PMCR 레지스터의 4번 bit가 뭔지 찾아보자.

Table 21-36. Page Mode Control Register (PMCR) Field Descriptions

Bit	Field	Value	Description
31-26	CS5_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS5. CS5 is not available on this device.
25	CS5_PG_SIZE		Page Size for NOR Flash connected on CS5. CS5 is not available on this device.
24	CS5_PG_MD_EN		Page Mode enable for NOR Flash connected on CS5. CS5 is not available on this device.
23-18	CS4_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS4. Number of EMIF_CLK cycles required for the page read data to be valid, minus one cycle. This value must not be cleared to 0.
17	CS4_PG_SIZE		Page Size for NOR Flash connected on CS4.
		0	Page size is 4 words
		1	Page size is 8 words
16	CS4_PG_MD_EN		Page Mode enable for NOR Flash connected on CS4.
		0	Page mode disabled for this chip select
		1	Page mode enabled for this chip select
15-10	CS3_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS3. Number of EMIF_CLK cycles required for the page read data to be valid, minus one cycle. This value must not be cleared to 0.
9	CS3_PG_SIZE		Page Size for NOR Flash connected on CS3.
		0	Page size is 4 words
		1	Page size is 8 words
8	CS3_PG_MD_EN		Page Mode enable for NOR Flash connected on CS3.
		0	Page mode disabled for this chip select
		1	Page mode enabled for this chip select
7-2	CS2_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS2. Number of EMIF_CLK cycles required for the page read data to be valid, minus one cycle. This value must not be cleared to 0.
1	CS2_PG_SIZE		Page Size for NOR Flash connected on CS2.
		0	Page size is 4 words
		1	Page size is 8 words
0	CS2_PG_MD_EN		Page Mode enable for NOR Flash connected on CS2.
		0	Page mode disabled for this chip select
		1	Page mode enabled for this chip select

7:2 bit는 Page에 접근하는 데 필요한 delay를 설정해주는 bit인 것 같다. 0이 되면 안된다고 써있다.

```

Getting Started  HL_sys_core.asm  HL_sys_main.c  HL_
146 if ((esmREG->SR1[2]) != 0U)
147 {
148     esmGroup3Notification(esmREG,esmREG->SR1[2]);
149 }
150

```

다음 코드인데, esmReg->SR1[2]가 뭔지 알아보자.

```

78 uint32 SR1[3U]; /* 0x0018, 0x001C, 0x0020 */

```

SR1[2] 이므로, offset은 0x0020이 된다.

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. <b>Read in User and Privileged mode. Write in Privileged mode only.</b>
		0	Read: No error occurred. Write: Leaves the bit unchanged.
		1	Read: Error occurred. Write: Clears the bit.

위 코드에서 ESMSR3 레지스터가 0이 아닐 경우, if문 안에 있는 함수를 호출한다.

ESMSR3 레지스터는 에러가 발생할 경우만 1이므로, read값은 0이 될 것이다.

따라서 if문은 건너뛰고 다음 코드를 살펴보자.

```

152 systemInit();

```

```

339 void systemInit(void)
340 {
341     /* USER CODE BEGIN (15) */
342     /* USER CODE END */
343
344     /* Configure PLL control registers and enable PLLs.
345      * The PLL takes (127 + 1024 * NR) oscillator cycles to acquire lock.
346      * This initialization sequence performs all the tasks that are not
347      * required to be done at full application speed while the PLL locks.
348      */
349     setupPLL();

```

```

73 void setupPLL(void)
74 {
75
76 /* USER CODE BEGIN (3) */
77 /* USER CODE END */
78
79 /* Disable PLL1 and PLL2 */
80 systemREG1->CSDISSET = 0x00000042U | 0x00000040U;
81 /*SAFETYMCUSW 28 D MR:NA <APPROVED> "Hardware status bit read check" */
82 while((systemREG1->CSDIS & 0x42U) != 0x42U)
83 {
84 /* Wait */
85 }

```

CSDISSET 레지스터에 0x00000042값을 set해준다. 즉, 1번,6번 bit를 1로 set한다.

Table 2-30. Clock Source Disable Set Register (CSDISSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-3	SETCLKSR[7-3]OFF	0	Set clock source[7-3] to the disabled state. Read: Clock source[7-3] is enabled. Write: Clock source[7-3] is unchanged.
		1	Read: Clock source[7-3] is disabled. Write: Clock source[7-3] is set to the disabled state. <b>Note:</b> After a new clock source disable bit is set via the CSDISSET register, the new status of the bit will be reflected in the CSDIS register (offset 30h), the CSDISSET register (offset 34h), and the CSDISCLR register (offset 38h).
2	Reserved	1	Reads return 1. Writes have no effect.
1-0	SETCLKSR[1-0]OFF	0	Set clock source[1-0] to the disabled state. Read: Clock source[1-0] is enabled. Write: Clock source[1-0] is unchanged.
		1	Read: Clock source[1-0] is disabled. Write: Clock source[1-0] is set to the disabled state. <b>Note:</b> After a new clock source disable bit is set via the CSDISSET register, the new status of the bit will be reflected in the CSDIS register (offset 30h), the CSDISSET register (offset 34h), and the CSDISCLR register (offset 38h).

clock source[1], clock source[6]을 disabled state로 설정한다.

Table 2-28. Clock Source Disable Register (CSDIS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-3	CLKSR[7-3]OFF	0	Clock source[7-3] off.
		1	Clock source[7-3] is enabled. Clock source[7-3] is disabled. <b>Note:</b> On wakeup, only clock sources 0, 4, and 5 are enabled.
2	Reserved	1	Reads return 1. Writes have no effect.
1-0	CLKSR[1-0]OFF	0	Clock source[1-0] off.
		1	Clock source[1-0] is enabled. Clock source[1-0] is disabled. <b>Note:</b> On wakeup, only clock sources 0, 4, and 5 are enabled.

while문은 clock source[1],clock source[6]이 정상적으로 disable state로 설정될 때까지 기다리는 코드다.

```

87 /* Clear Global Status Register */
88 systemREG1->GBLSTAT = 0x301U;

```

GBLSTAT 레지스터에 0x301를 set한다. 0,8,9번 bit를 1로 set한다.

Table 2-67. Global Status Register (GBLSTAT) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9	FBSLIP	0	PLL over cycle slip detection. (cleared by nPORRST, maintains its previous value for all other resets). Read: No PLL over cycle slip has been detected. Write: The bit is unchanged.
		1	Read: A PLL over cycle slip has been detected. Write: The bit is cleared to 0.
8	RFSLIP	0	PLL under cycle slip detection. (cleared by nPORRST, maintains its previous value for all other resets). Read: No PLL under cycle slip has been detected. Write: The bit is unchanged.
		1	Read: A PLL under cycle slip has been detected. Write: The bit is cleared to 0.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	OSCFAIL	0	Oscillator fail flag bit. (cleared by nPORRST, maintains its previous value for all other resets). Read: No oscillator failure has been detected. Write: The bit is unchanged.
		1	Read: An oscillator failure has been detected. Write: The bit is cleared to 0.

0,8,9번을 1로 write하면, 0으로 바뀌면서 아무 일도 일어나지 않는다.

0,8,9번이 set되어 있다는 것은, 각각의 bit에 해당하는 에러가 발생했다는 뜻이므로, 에러 플래그를 초기화하는 코드가 된다.

```

101 systemREG1->PLLCTL1 = (uint32)0x00000000U
102 | (uint32)0x20000000U
103 | (uint32)((uint32)0x1FU << 24U)
104 | (uint32)0x00000000U
105 | (uint32)((uint32)(8U - 1U)<< 16U)
106 | (uint32)(0x9500U);

```

$$\begin{aligned}
 \text{PLLCTL1} &= 0x2000\ 0000 \mid 0x1F00\ 0000 \mid 0x0007\ 0000 \mid 0x0000\ 9500 \\
 &= 0x3F07\ 9500
 \end{aligned}$$

29,28 27,26,25,24 18,17,16 15,12 10,8번 bit 1로 set

			<b>Note: BPOS (Bits 30-29) must also be enabled for ROS to be enabled.</b>
30-29	BPOS	<div>2h</div> <div>Others</div>	<p>Bypass of PLL Slip.</p> <p>Bypass on PLL Slip is disabled. If a PLL Slip is detected no action is taken.</p> <p>Bypass on PLL Slip is enabled. If a PLL Slip is detected the device will automatically bypass the PLL and use the oscillator to provide the device clock.</p> <p><b>Note: If ROS (Bit 31) is set to 1, the device will be reset if a PLL Slip and the PLL will be bypassed after the reset occurs.</b></p>
28-24	PLLDIV	<div>0</div> <div>1h</div> <div>:</div> <div>1Fh</div>	<p>PLL Output Clock Divider</p> $R = PLLDIV + 1$ $f_{PLL\ CLK} = f_{post\_ODCLK} / R$ $f_{PLL\ CLK} = f_{post\_ODCLK} / 1$ $f_{PLL\ CLK} = f_{post\_ODCLK} / 2$ $f_{PLL\ CLK} = f_{post\_ODCLK} / 32$
23	ROF	<div>0</div> <div>1</div>	<p>Reset on Oscillator Fail.</p> <p>Do not reset system when oscillator is out of range.</p> <p>The ROF bit enables the OSC_FAIL condition to generate a system reset. If the ROF bit in the PLLCTL1 register is set when the oscillator fails, then a system reset occurs.</p>
22	Reserved	0	Value has no effect on PLL operation.
21-16	REFCLKDIV	<div>0</div> <div>1h</div> <div>:</div> <div>3Fh</div>	<p>Reference Clock Divider</p> $NR = REFCLKDIV + 1$ $f_{INT\ CLK} = f_{OSCIN} / NR$ $f_{INT\ CLK} = f_{OSCIN} / 1$ $f_{INT\ CLK} = f_{OSCIN} / 2$ $f_{INT\ CLK} = f_{OSCIN} / 64$
15-0	PLLMUL	<div>0h</div> <div>100h</div> <div>:</div> <div>5B00h</div> <div>5C00h</div> <div>:</div> <div>FF00h</div>	<p>PLL Multiplication Factor</p> $NF = (PLLMUL / 256) + 1, \text{ valid multiplication factors are from 1 to 256.}$ $f_{VCO\ CLK} = f_{INT\ CLK} \times NF$ $f_{VCO\ CLK} = f_{INT\ CLK} \times 1$ $f_{VCO\ CLK} = f_{INT\ CLK} \times 2$ $f_{VCO\ CLK} = f_{INT\ CLK} \times 92$ $f_{VCO\ CLK} = f_{INT\ CLK} \times 93$ $f_{VCO\ CLK} = f_{INT\ CLK} \times 256$

29번 bit : 2가 아니므로, PLL Slip에서 바이패스를 허용한다.

28:24번 bit : 1F이므로

$$LLCLK = f_{ostODCLK}/32$$

18:16번 bit : 7이므로

$$f_{CLK} = f_{OSCIN}/8$$

15:0번 bit : 0x9500==149이므로

$$f_{VCOCLK} = f_{CLK}/150$$

```

114 systemREG1->PLLCTL2 = (uint32)((uint32)255U << 22U)
115 | (uint32)((uint32)7U << 12U)
116 | (uint32)((uint32)(1U - 1U) << 9U)
117 | (uint32)61U;

```

PLLCTL2 = 0x3FC0 0000 | 0x0000 7000 | 0x0000 00(0011)(1101)

29,28 27,26,24,23 22,21 14,13,12 5,4,3,2,0 1로 set, 9번 bit clear.

Bit	Field	Value	Description
31	FMENA	0 1	Frequency Modulation Enable. Disable frequency modulation. Enable frequency modulation.
30-22	SPREADINGRATE	0 1h : 1FFh	NS = SPREADINGRATE + 1 $f_{mod} = f_s = f_{INT\_CLK} / (2 \times NS)$ $f_{mod} = f_s = f_{INT\_CLK} / (2 \times 1)$ $f_{mod} = f_s = f_{INT\_CLK} / (2 \times 2)$ : $f_{mod} = f_s = f_{INT\_CLK} / (2 \times 512)$
21	Reserved	0	Value has no effect on PLL operation.
20-12	MULMOD	0 8h 9h : 1FFh	Multiplier Correction when Frequency Modulation is enabled. When FMENA = 0, MUL_when_MOD = 0; when FMENA = 1, MUL_when_MOD = (MULMOD / 256) No adder to NF. MUL_when_MOD = 8/256 MUL_when_MOD = 9/256 : MUL_when_MOD = 511/256
11-9	ODPLL	0 1h : 7h	Internal PLL Output Divider OD = ODPLL + 1 $f_{post-ODCLK} = f_{VCO\_CLK} / OD$ <b>Note: PLL output clock is gated off, if ODPLL is changed while the PLL is active.</b> $f_{post-ODCLK} = f_{VCO\_CLK} / 1$ $f_{post-ODCLK} = f_{VCO\_CLK} / 2$ : $f_{post-ODCLK} = f_{VCO\_CLK} / 8$
8-0	SPR_AMOUNT	0 1h : 1FFh	Spreading Amount $NV = (SPR\_AMOUNT + 1) / 2048$ NV ranges from 1/2048 to 512/2048 Note that the PLL output clock is disabled for 1 modulation period, if the SPR_AMOUNT field is changed while the frequency modulation is enabled. If frequency modulation is disabled and SPR_AMOUNT is changed, there is no effect on the PLL output clock. NV = 1/2048 NV = 2/2048 : NV = 512/2048

30-22 : 0xff 이므로  $f_{mod} = f_s = f_{CLK} / 512$

20-12 : 7 이므로 no function

11-9 : 0 이므로  $f_{postODCLK} = f_{VCOCLK} / 1$

8-0 : 61 이므로 NV = 62/2048

```

127 systemREG2->PLLCTL3 = ((uint32)((uint32)(1U - 1U) << 29U)
128 | (uint32)((uint32)0x1FU << 24U)
129 | (uint32)((uint32)(8U - 1U)<< 16U)
130 | (uint32)(0x9500U));

```

29번 bit clear, 28,27,26,25,24번 bit set, 0x0007 0000 set, 0x0000 9500 set

Bit	Field	Value	Description
31-29	ODPLL2	0 $f_{POST\_ODCLK2} = f_{INPUT\_CLK2} / 1$ 1h $f_{POST\_ODCLK2} = f_{INPUT\_CLK2} / 2$ : 7h $f_{POST\_ODCLK2} = f_{INPUT\_CLK2} / 8$	Internal PLL Output Divider $OD2 = ODPLL2 + 1$ , ranges from 1 to 8. $f_{POST\_ODCLK2} = f_{INPUT\_CLK2} / OD2$ <b>Note: PLL output clock is gated off if ODPLL2 is changed while the PLL#2 is active.</b>
28-24	PLLDIV2	0 $f_{PLL2\_CLK} = f_{POST\_ODCLK2} / R2$ 1h $f_{PLL2\_CLK} = f_{POST\_ODCLK2} / 2$ : 1Fh $f_{PLL2\_CLK} = f_{POST\_ODCLK2} / 32$	PLL2 Output Clock Divider $R2 = PLLDIV2 + 1$ , ranges from 1 to 32. $f_{PLL2\_CLK} = f_{POST\_ODCLK2} / R2$
23-22	Reserved	0	Value has no effect on PLL operation.
21-16	REFCLKDIV2	0 $f_{INTCLK2} = f_{OSCIN} / 1$ 1h $f_{INTCLK2} = f_{OSCIN} / 2$ : 3Fh $f_{INTCLK2} = f_{OSCIN} / 64$	Reference Clock Divider $NR2 = REFCLKDIV2 + 1$ , ranges from 1 to 64. $f_{INTCLK2} = f_{OSCIN} / NR2$ <b>Note: This value should not be changed while the PLL2 is active.</b>
15-0	PLLMUL2	100h $f_{VCOCLK2} = f_{INTCLK2} \times 1$ : 5B00h $f_{VCOCLK2} = f_{INTCLK2} \times 92$ 5C00h $f_{VCOCLK2} = f_{INTCLK2} \times 93$ : FF00h $f_{VCOCLK2} = f_{INTCLK2} \times 256$	PLL2 Multiplication Factor $NF2 = (PLLMUL2 / 256) + 1$ , valid multiplication factors are from 1 to 256. $f_{VCOCLK2} = f_{INTCLK2} \times NF2$ User and privileged mode (read): Privileged mode (write):

31-29 :  $f_{ost\_DCLK2} = f_{OUTPUTCLK2} / 1$

28-24 :  $f_{PLL2CLK} = f_{POSTODCLK2} / 32$

21-16 :  $f_{CLK2} = f_{OSCIN} / 8$

15-0 :  $f_{VCOCLK2} = f_{CLK2} * 150$

로 PLL1, PLL2, PLL3의 clock 등을 설정해준다.

```

systemREG1->CSDIS = 0x00000000U
| 0x00000000U
| 0x00000008U
| 0x00000080U
| 0x00000000U
| 0x00000000U
| 0x00000000U
| 0x00000000U
| 0x0000004U;

```

CSDIS = 0x0000008C 이므로 2,3,7번 bit를 1로 set해준다.

Table 2-28. Clock Source Disable Register (CSDIS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-3	CLKSR[7-3]OFF	0 Clock source[7-3] is enabled. 1 Clock source[7-3] is disabled. <b>Note: On wakeup, only clock sources 0, 4, and 5 are enabled.</b>	
2	Reserved	1	Reads return 1. Writes have no effect.
1-0	CLKSR[1-0]OFF	0 Clock source[1-0] is enabled. 1 Clock source[1-0] is disabled. <b>Note: On wakeup, only clock sources 0, 4, and 5 are enabled.</b>	

Clock Source 3번과 7번을 disabled 해준다.

Table 2-29. Clock Sources Table

Clock Source #	Clock Source Name
Clock Source 0	Oscillator
Clock Source 1	PLL1
Clock Source 2	Not Implemented
Clock Source 3	EXTCLKIN
Clock Source 4	Low Frequency LPO (Low Power Oscillator) clock
Clock Source 5	High frequency LPO (Low Power Oscillator) clock
Clock Source 6	PLL2
Clock Source 7	EXTCLKIN2

EXTCLKIN과 EXTCLKIN2가 disabled 되는 것인데, 뭔지 검색이 안되긴 해도 외부 clock 입력을 막는 것이라는 것으로 추측할 수는 있다



```

Getting Started  HL_sys_core.asm  HL_sys_main.c  HL_sys_intvecs.asm
355  /* Enable clocks to peripherals and release peripheral reset */
356  periphInit();
그 다음 줄에 periphInit()함수가 있으니 들어가보자

```

```

Getting Started  HL_sys_core.asm  HL_sys_main.c
218  /** - Disable Peripherals before peripheral p
219  systemREG1->CLKCNTL &= 0xFFFFFEFF;
220
221  /** - Release peripherals from reset and enab
222  /** - Power-up all peripherals */
223  pcrREG1->PSPWRDNCLR0 = 0xFFFFFFFFFU;
224  pcrREG1->PSPWRDNCLR1 = 0xFFFFFFFFFU;
225  pcrREG1->PSPWRDNCLR2 = 0xFFFFFFFFFU;
226  pcrREG1->PSPWRDNCLR3 = 0xFFFFFFFFFU;
227
228  pcrREG2->PSPWRDNCLR0 = 0xFFFFFFFFFU;
229  pcrREG2->PSPWRDNCLR1 = 0xFFFFFFFFFU;
230  pcrREG2->PSPWRDNCLR2 = 0xFFFFFFFFFU;
231  pcrREG2->PSPWRDNCLR3 = 0xFFFFFFFFFU;
232
233  pcrREG3->PSPWRDNCLR0 = 0xFFFFFFFFFU;
234  pcrREG3->PSPWRDNCLR1 = 0xFFFFFFFFFU;
235  pcrREG3->PSPWRDNCLR2 = 0xFFFFFFFFFU;
236  pcrREG3->PSPWRDNCLR3 = 0xFFFFFFFFFU;
237
238  /** - Enable Peripherals */
239  systemREG1->CLKCNTL |= 0x0000100U;

```

뭔가 많아보이지만, 이름이 똑같고 0,1,2,3으로 다른 애들이 있으니 금방 할 수 있을 것 같다. CLKCNTL &= 0xFFFFFEFF; 이므로, CLKCNTL의 8번 비트를 0으로 clear해주는 코드이다. CLKCNTL의 정의부로 들어가서 offset을 본 후, 8번 비트가 어떤 역할을 하는지 알아보자. 이름을 보니, clcok을 control하는 녀석인 듯 하다.

```

120  uint32 CLKCNTL; /* 0x00D0 */

```

Table 2-61. Clock Control Register (CLKCNTL) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	VCLK2R	0 : Fh	VBUS clock2 ratio.  Note: The VCLK2 frequency must always be greater than or equal to the VCLK frequency. The VCLK2 frequency must be an integer multiple of the VCLK frequency. In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously. The VCLK2 speed is HCLK divided by 1. : The VCLK2 speed is HCLK divided by 16.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	VCLKR	0 : Fh	VBUS clock ratio.  Note: The VCLK2 frequency must always be greater than or equal to the VCLK frequency. The VCLK2 frequency must be an integer multiple of the VCLK frequency. In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously. The VCLK speed is HCLK divided by 1. : The VCLK speed is HCLK divided by 16.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	PENA	0 1	Peripheral enable bit. The application must set this bit before accessing any peripheral. The global peripheral/peripheral memory frames are in reset. All peripheral/peripheral memory frames are out of reset.
7-0	Reserved	0	Reads return 0. Writes have no effect.

8번 bit가 0으로 clear되면, 모든 peripheral과 peripheral memory frames의 reset을 가능하게 한다.

즉, 이 뒤로 peripheral을 reset하기 위한 코드가 있을 것이다 라고 추측은 할 수 있다.

다음 코드는 PSPWRDNCLR0=0xFFFFFFFF;인데, 일단 검색이 안된다.

PCR 레지스터 목록을 찾은 뒤, offset을 통해 찾아보자.

```

96  uint32 PSPWRDNCLR0; /* 0x00A0 */

```

Table 2-106. Peripheral Power-Down Clear Register 0 (PSPWRDNCLR0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7:0]QUAD[3:0] PWRDNCLR	0 1	Peripheral select quadrant clock power-down clear. Read: The clock to the peripheral select quadrant is active. Write: The bit is unchanged. Read: The clock to the peripheral select quadrant is inactive. Write: The corresponding bit in PSPWRDNSET0 and PSPWRDNCLR0 registers is cleared to 0.

이름이 다른 것도 아닌데, 검색이 안된 이유는 모르겠다. 검색이 되지 않아도 다른 방법을 통해 찾아보도록 하자.

이 레지스터를 1로 set하면, PSPWRDNSET 레지스터와 이 레지스터의 해당되는 bit가 0으로 clear된다.

PSPWRDNSET을 찾아보자. 나머지 PCR2, PCR3도 같은 코드들이다.

Table 2-102. Peripheral Power-Down Set Register 0 (PSPWRDNSET0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7:0]QUAD[3:0] PWRDNSET	0 1	Peripheral select quadrant clock power-down set. Read: The clock to the peripheral select quadrant is active. Write: The bit is unchanged. Read: The clock to the peripheral select quadrant is inactive. Write: The corresponding bit in PSPWRDNSET0 and PSPWRDNCLR0 registers is set to 1.

마지막 코드는 CLKCNTL |= 0x00000100;이니 clear했던 8번 bit를 1로 set해준다.

Table 2-61. Clock Control Register (CLKCNTL) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	VCLK2R	0 : Fh	VBUS clock2 ratio. <b>Note:</b> The VCLK2 frequency must always be greater than or equal to the VCLK frequency. The VCLK2 frequency must be an integer multiple of the VCLK frequency. In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously. The VCLK2 speed is HCLK divided by 1. The VCLK2 speed is HCLK divided by 16.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	VCLKR	0 : Fh	VBUS clock ratio. <b>Note:</b> The VCLK2 frequency must always be greater than or equal to the VCLK frequency. The VCLK2 frequency must be an integer multiple of the VCLK frequency. In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously. The VCLK speed is HCLK divided by 1. The VCLK speed is HCLK divided by 16.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	PENA	0 1	Peripheral enable bit. The application must set this bit before accessing any peripheral. The global peripheral/peripheral memory frames are in reset. All peripheral/peripheral memory frames are out of reset.
7-0	Reserved	0	Reads return 0. Writes have no effect.

peripheral이 reset되는 것을 방지한다고 써있다. 즉, 일련의 코드들은 peripgeral의 reset을 허용하게 한 뒤, pcr1부터 pcr3까지 peripheral들을 reset한 뒤, reset을 막는 코드가 된다.

```
362     muxInit();
```

다음 코드인 muxInit()으로 가보자

```
230     pinMuxReg->KICKER0 = 0x83E70B13U;  
231     pinMuxReg->KICKER1 = 0x95A4F1E0U;
```

KICKER가 보인다. KICKER로 데이터 시트 또는 레퍼런스 매뉴얼을 검색해보자.

Table 6-15. Kicker Register 0 Field Descriptions

Bit	Field	Description
31-0	KICK0	Kicker 0 Register. The value 83E7 0B13h must be written to KICK0 as part of the process to unlock the CPU write access to the PINMMRnn registers.

Table 6-16. Kicker Register 1 Field Descriptions

Bit	Field	Description
31-0	KICK1	Kicker 1 Register. The value 95A4 F1E0h must be written to the KICK1 as part of the process to unlock the CPU write access to the PINMMRnn registers.

읽어보니, KICK0와 KICK1 둘 다 특정 값을 넣어야 CPU가 PINMMRnn 레지스터들에 쓰기 권한을 얻을 수 있는 것 같다. 다음 코드를 보자.

```
236     pinMuxReg->PINMUX[0] = PINMUX_BALL_N19_AD1EVT | PINMUX_BALL_D4_EMIF_ADDR_00 | PINMUX_BALL_D5_EMIF_ADDR_01 | PINMUX_BALL_C4_EMIF_ADDR_06;  
237  
238     pinMuxReg->PINMUX[1] = PINMUX_BALL_C5_EMIF_ADDR_07 | PINMUX_BALL_C6_EMIF_ADDR_08 | PINMUX_BALL_C7_EMIF_ADDR_09 | PINMUX_BALL_C8_EMIF_ADDR_10;  
239  
240     pinMuxReg->PINMUX[2] = PINMUX_BALL_C9_EMIF_ADDR_11 | PINMUX_BALL_C10_EMIF_ADDR_12 | PINMUX_BALL_C11_EMIF_ADDR_13 | PINMUX_BALL_C12_EMIF_ADDR_14;  
241  
242     pinMuxReg->PINMUX[3] = PINMUX_BALL_C13_EMIF_ADDR_15 | PINMUX_BALL_D14_EMIF_ADDR_16 | PINMUX_BALL_C14_EMIF_ADDR_17 | PINMUX_BALL_D15_EMIF_ADDR_18;  
243  
244     pinMuxReg->PINMUX[4] = PINMUX_BALL_C15_EMIF_ADDR_19 | PINMUX_BALL_C16_EMIF_ADDR_20 | PINMUX_BALL_C17_EMIF_ADDR_21;  
245  
246     pinMuxReg->PINMUX[5] = 0U;
```

PINMUX 배열들에 어떤 값들을 넣어주고 있다.

```
86     uint32 PINMUX[180];          /*< 0x110 - 1A4 : Output Pin Multiplexing Control Registers (38 registers); 0x250 - 0x29C
```

PINMUX는 180개짜리 배열로 선언되어 있다.

mux로 검색해보니, 결과가 나오지 않는다. 이럴 땐 abbreviated name이 아닌 full name으로도 검색해보자 (multiplex)

#### Control of Multiplexed Inputs

www.ti.com

Table 6-1. Multiplexing for Outputs on 337ZWT Package

Address Offset	337ZWT BALL	Default Function	Selection Bit	Alternate Function 1	Selection Bit	Alternate Function 2	Selection Bit	Alternate Function 3	Selection Bit	Alternate Function 4	Selection Bit	Alternate Function 5	Selection Bit
110h	N19	AD1EVT	0[0]			MII_RX_ER	0[2]	RMII_RX_ER	0[3]			nTZ1_1	0[5]
	D4	EMIF_ADDR[00]	0[8]			N2HET2[01]	0[10]						
	D5	EMIF_ADDR[01]	0[16]			N2HET2[03]	0[18]						
	C4	EMIF_ADDR[06]	0[24]	RTP_DATA[13]	0[25]	N2HET2[11]	0[26]						
114h	C5	EMIF_ADDR[07]	1[0]	RTP_DATA[12]	1[1]	N2HET2[13]	1[2]						
	C6	EMIF_ADDR[08]	1[8]	RTP_DATA[11]	1[9]	N2HET2[15]	1[10]						
	C7	EMIF_ADDR[09]	1[16]	RTP_DATA[10]	1[17]								
	C8	EMIF_ADDR[10]	1[24]	RTP_DATA[09]	1[25]								
118h	C9	EMIF_ADDR[11]	2[0]	RTP_DATA[08]	2[1]								
	C10	EMIF_ADDR[12]	2[8]	RTP_DATA[06]	2[9]								
	C11	EMIF_ADDR[13]	2[16]	RTP_DATA[05]	2[17]								
	C12	EMIF_ADDR[14]	2[24]	RTP_DATA[04]	2[25]								

이런 테이블이 나온다. 코드가 너무 많으므로, 첫 줄만 자세히 분석해보자.

```
236     pinMuxReg->PINMUX[0] = PINMUX_BALL_N19_AD1EVT | PINMUX_BALL_D4_EMIF_ADDR_00 | PINMUX_BALL_D5_EMIF_ADDR_01 | PINMUX_BALL_C4_EMIF_ADDR_06;
```

테이블을 보니, 110에 해당하는 레지스터 1개가 N19, D4, D5, C4 4개의 pin의 output을 결정해주는 것 같다.

코드에 or 연산이 3개가 있는데, 각 4개의 define은 1개당 1개의 pin의 output을 정해준다고 추측할 수 있다.

첫 번째 define을 보니, PINMUX\_BALL\_N19\_AD1EVT라고 써있다. N19를 AD1EVT로 사용할 수 있게 해주는 놈이다. N19를 AD1EVT로 사용하려면, Selection bit를 0[0]으로 설정해줘야 된다고 써있다. 이게 정확히 무슨 뜻인지는 define을 보면서 추측해보자.

```
447 #define PINMUX BALL_N19_AD1EVT ((uint32)((uint32)0x1U << PINMUX BALL_N19_SHIFT))
55 #define PINMUX BALL_N19_SHIFT 0U
```

0x1 << 0 이 되므로, 0번 bit를 1로 set한다는 뜻이 된다.

즉, 괄호 [] 안에 있는 숫자 번째의 bit를 1로 set하면, 해당 output으로 설정할 수 있다는 뜻이다.

두 번째 define은 PINMUX\_BALL\_D4\_EMIF\_ADDR\_00 이므로, D4를 EMIF\_ADDR[00]으로 설정하겠다는 의미가 된다. 테이블을 보니, Selection Bit가 0[8]이므로, 8번째 bit를 1로 set한다는 의미가 된다.

즉, #define PINMUX\_BALL\_D4\_EMIF\_ADDR\_00 0x0000 0100로 정의되어 있을 것이다. 확인해보자.

```
452 #define PINMUX BALL_D4_EMIF_ADDR_00 ((uint32)((uint32)0x1U << PINMUX BALL_D4_SHIFT))
56 #define PINMUX BALL_D4_SHIFT 8U
```

1<<8 이므로 0x0000 0100이 맞다.

같은 방식으로 PINMUX\_BALL\_D5\_EMIF\_ADDR\_01 과 PINMUX\_BALL\_C4\_EMIF\_ADDR\_06 각각 0x0001 0000, 0x0100 0000로 define 되어 있을 것이다.

```
455 #define PINMUX BALL_D5_EMIF_ADDR_01 ((uint32)((uint32)0x1U << PINMUX BALL_D5_SHIFT))
```

```
57 #define PINMUX BALL_D5_SHIFT 16U
```

```
458 #define PINMUX BALL_C4_EMIF_ADDR_06 ((uint32)((uint32)0x1U << PINMUX BALL_C4_SHIFT))
```

```
58 #define PINMUX BALL_C4_SHIFT 24U
```

확인되었다.

그럼 테이블에서 Selection bit의 0[24]에서 24의 의미는 알았지만, 0의 의미는 정확히 알 수 없다.

하지만, 생긴 형태를 보니 0번째 레지스터를 의미하는 것 같다. 즉 0번째 레지스터의 24번 bit를 set하면, C4 핀의 출력을 EMIF\_ADDR[06]으로서 사용하게 한다 라는 뜻이다.