TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – GJ (박현우)
uc820@naver.com

목차

1. Cortex-R5F Hercules Safety MCU - boot code 분석 (_c_int00)

HL_sys_intvecs.asm | HL_sys_startup.c | HL_sys_core.asm | HL_sys_core.h | HL_system.c

```
;  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
;  LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
;  DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
;  THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
;  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
;  OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
;
;
;

    .sect ".intvecs"
    .arm

;---------------------------------------------------------------------------
; import reference for interrupt routines

    .ref _c_int00
    .ref phantomInterrupt
    .def resetEntry

;---------------------------------------------------------------------------
; interrupt vectors

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]

;---------------------------------------------------------------------------
```

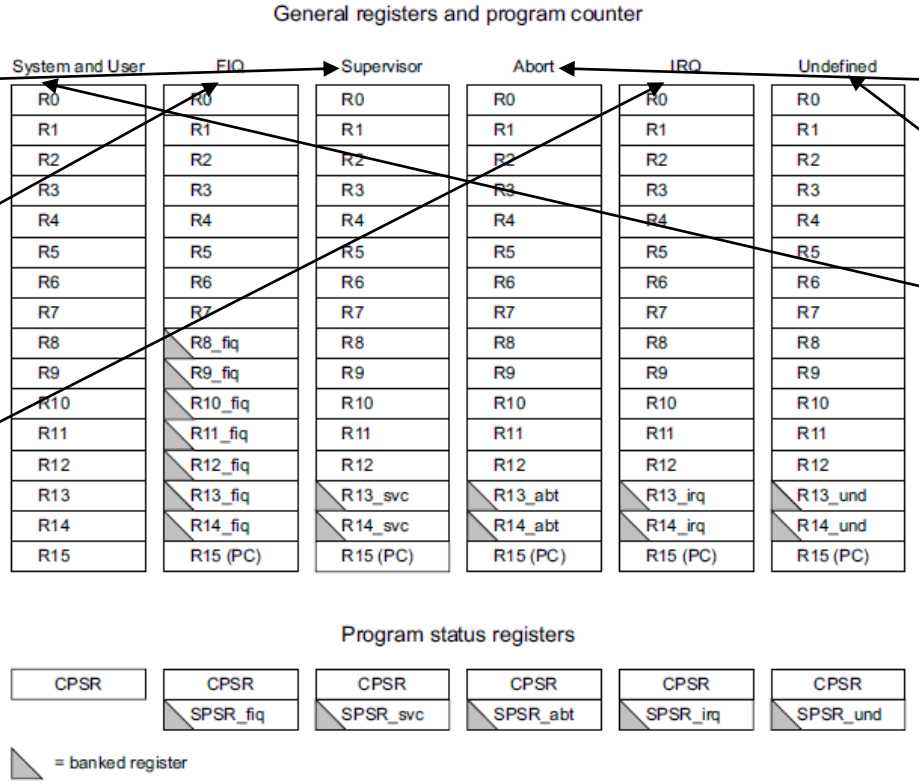
Interrupt vectors : 인터럽트 처리를 위한 branch를 저장하고 있음.

```
/* USER CODE END */

/* Startup Routine */
void _c_int00(void);
/* USER CODE BEGIN (4) */
/* USER CODE END */

#pragma CODE_STATE(_c_int00, 32)
#pragma INTERRUPT(_c_int00, RESET)
#pragma WEAK(_c_int00)

/* SourceId : STARTUP_SourceId_001 */
/* DesignId : STARTUP_DesignId_001 */
/* Requirements : HL_CONQ_STARTUP_SR1 */
void _c_int00(void)
{

/* USER CODE BEGIN (5) */
/* USER CODE END */

    /* Initialize Core Registers to avoid CCM Error */
    _coreInitRegisters_();

    /* Initialize Stack Pointers */
    _coreInitStackPointer_();

    /* Reset handler: the following instructions read from the system exception status register
     * to identify the cause of the CPU reset.
     */
    switch(getResetSource())
    {
        case POWERON_RESET:
        case DEBUG_RESET:
        case EXT_RESET:

/* USER CODE BEGIN (6) */
/* USER CODE END */

        /* Initialize L2RAM to avoid ECC errors right after power on */
        _memInit_();
```

_c_int00 : Cortex-R5F 시작 루틴을 위한 초기 함수

1. Cortex-R5F Hercules Safety MCU - boot code 분석 (_coreInitRegisters_ 1)

```
_coreInitRegisters_
    ; After reset, the CPU is in the Supervisor mode (M = 10011)
        mov r0, lr
        mov r1, #0x0000
        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
        msr spsr_cxsf, r1
        ; Switch to FIQ mode (M = 10001)
        cps #17
        mov lr, r0
        mov r8, #0x0000
        mov r9, #0x0000
        mov r10, #0x0000
        mov r11, #0x0000
        mov r12, #0x0000
        mrs r1, cpsr
        msr spsr_cxsf, r1
        ; Switch to IRQ mode (M = 10010)
        cps #18
        mov lr, r0
        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
```

Figure 3-3 Register organization

Note

For 16-bit Thumb instructions, the high registers, R8–R15, are not part of the standard register set. You can use special variants of the MOV instruction to transfer a value from a low register, in the range R0–R7, to a high register, and from a high register to a low register. The CMP instruction enables you to compare high register values with low register values. The ADD instruction enables you to add high register values to low register values. For more information, see the *ARM Architecture Reference Manual*.

coreInitRegisters : mode별 레지스터 초기화.
사용 이유 : hardware에 용도에 맞는 mode를 분리해서 설정해 둠으로써, context switching의 비용을 줄임.

1. Cortex-R5F Hercules Safety MCU - boot code 분석 (_coreInitRegisters_ 2)

```
; Floating Point Co-Processor Initialization. FPU needs to be enabled first.

        mrc   p15,     #0x00,       r2,       c1, c0, #0x02
        orr   r2,      r2,          #0xF00000
        mcr   p15,     #0x00,       r2,       c1, c0, #0x02
        mov   r2,      #0x40000000
        fmxr  fpexc,    r2
```

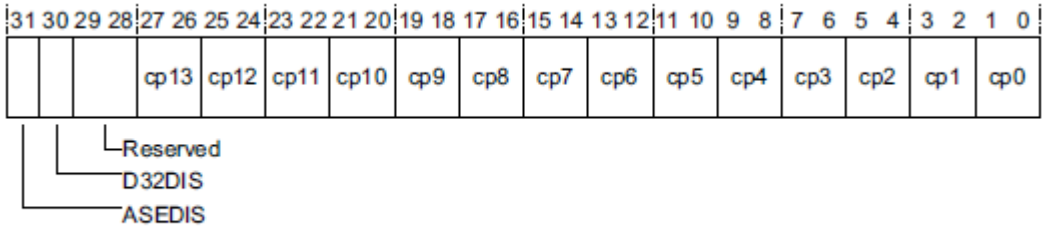


Figure 4-30 CPACR bit assignments

To access the CPACR, read or write CP15 with:

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

MCR p15, 0, <Rd>, c1, c0, 2 ; Write CPACR
```

FPU를 활성화 하기 위해서는 coprocessor 초기화가 필요하다.
즉, 부동소수점 계산을 하기 위함.

그러므로, MRC p15와 MCR p15의 코드를 사용하여 CPACR을 위와 같은 방식으로 설정해야 한다.

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (_coreInitRegisters_ 3)

```
        fmdrr d0,       r1,     r1
        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

    .endasmfunc
```

ARM11 프로세서

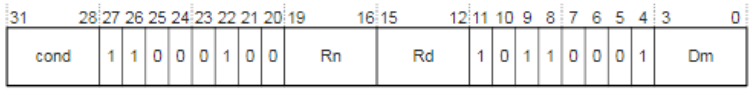
VFP11™ Vector Floating-point Coprocessor Technical Reference Manual — for ARM1136JF-S processor r1p5

Home > Programmer's Model > ARMv5TE coprocessor extensions > FMDRR

3.3.1. FMDRR

FMDRR transfers data from two ARM1136 registers to a VFP11 double-precision register. The ARM1136 registers do not have to be contiguous. Figure 3.1 shows the format of the FMDRR instruction.

Figure 3.1. FMDRR instruction format



Syntax

```
FMDRR {<cond>} <Dm>, <Rd>, <Rn>
```

where:

<cond>
Is the condition under which the instruction is executed. If <cond> is omitted, the AL (always) condition is used.

<Dm>
Specifies the destination double-precision VFP11 coprocessor register.

<Rd>
Specifies the source ARM1136 register for the lower 32 bits of the operand.

<Rn>
Specifies the source ARM1136 register for the upper 32 bits of the operand.

FMDRR : 두 개의 ARM1136 레지스터에서 VFP11 배정 밀도 레지스터로 데이터를 전송한다.

bl 을 여러 번 하는 이유는 VFP가 안정적으로 데이터 전송을 받기 위한 시간을 확보하기 위해서이다.

1. Cortex-R5F Hercules Safety MCU - boot code 분석 (_coreInitStackPointer_)

```
    .def     _coreInitStackPointer_
    .asmfunc

_coreInitStackPointer_

        cps   #17
        ldr   sp,       fiqSp
        cps   #18
        ldr   sp,       irqSp
        cps   #19
        ldr   sp,       svcSp
        cps   #23
        ldr   sp,       abortSp
        cps   #27
        ldr   sp,       undefSp
        cps   #31
        ldr   sp,       userSp
        bx    lr

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
```

Stack pointer를 쓴 이유는 mode에 맞는 주소 공간을 확보하기 위함.

즉, 데이터가 꼬이게 하지 않기 위해서 필요함.

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (getResetSource())

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

    if ((SYS_EXCEPTION & (uint32)POWERON_RESET) != 0U)
    {
        /* power-on reset condition */
        rst_source = POWERON_RESET;

        /* Clear all exception status Flag and proceed since it's power up */
        SYS_EXCEPTION = 0x0000FFFFU;
    }
```

6.4 Warm Reset (nRST)

This is a bidirectional reset signal. The internal circuitry drives the signal low on detecting any device reset condition. An external circuit can assert a device reset by forcing the signal low. On this terminal, the output buffer is implemented as an open drain (drives low only). To ensure an external reset is not arbitrarily generated, TI recommends that an external pullup resistor is connected to this terminal.

This terminal has a glitch filter. It also has an internal pullup

6.4.1 *Causes of Warm Reset*

Table 6-5. Causes of Warm Reset

DEVICE EVENT	SYSTEM STATUS FLAG
Power-Up Reset	Exception Status Register, bit 15
Oscillator fail	Global Status Register, bit 0
PLL slip	Global Status Register, bits 8 and 9
Watchdog exception	Exception Status Register, bit 13
Debugger reset	Exception Status Register, bit 11
CPU Reset (driven by the CPU STC)	Exception Status Register, bit 5
Software Reset	Exception Status Register, bit 4
External Reset	Exception Status Register, bit 3

```
typedef enum
{
    POWERON_RESET        = 0x8000U,  /**< Alias for Power On Reset     */
    OSC_FAILURE_RESET    = 0x4000U,  /**< Alias for Osc Failure Reset  */
    WATCHDOG_RESET       = 0x2000U,  /**< Alias for Watch Dog Reset    */
    WATCHDOG2_RESET      = 0x1000U,  /**< Alias for Watch Dog 2 Reset  */
    DEBUG_RESET          = 0x0800U,  /**< Alias for Debug Reset        */
    INTERCONNECT_RESET   = 0x0080U,  /**< Alias for Interconnect Reset */
    CPU0_RESET           = 0x0020U,  /**< Alias for CPU 0 Reset        */
    SW_RESET             = 0x0010U,  /**< Alias for Software Reset     */
    EXT_RESET            = 0x0008U,  /**< Alias for External Reset     */
    NO_RESET             = 0x0000U   /**< Alias for No Reset           */
}resetSource_t;

#define SYS_EXCEPTION        (*(volatile uint32 *)0xFFFFFFE4U)
```

SYS_EXCEPTION : POWERON_RESET이 1로 셋팅이 되어 있다면,

0x0000FFFFU함으로써 다른 all exception status flag를 초기화.

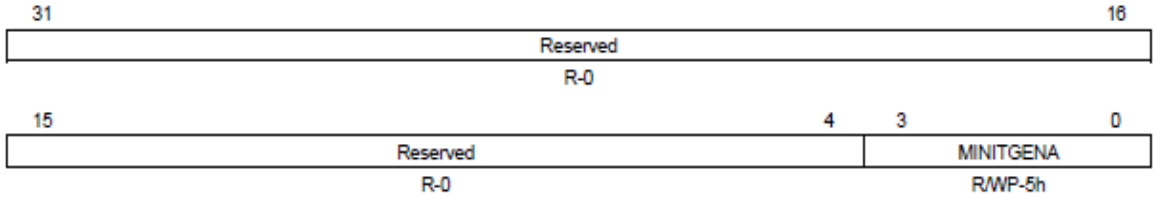
1. Cortex-R5F Hercules Safety MCU - boot code 분석 (_memInit_ 1)

```
;Initialize RAM memory

       .def _memInit_
       .asmfunc

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

       ldr   r11,  MSIENA      ;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
       beq   mloop

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

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.

MINITGCR : 전역 메모리 하드웨어 초기화

1. Cortex-R5F Hercules Safety MCU - boot code 분석 (_memInit_ 2)

```
;Initialize RAM memory

       .def _memInit_
       .asmfunc

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

       ldr   r11,  MSIENA      ;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
       beq   mloop

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

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.**

MSINENA : 메모리 자체 테스트를 위한 PBIST 컨트롤러 활성화와 메모리 모듈 초기화

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (_memInit_ 3)

```
;Initialize RAM memory

       .def _memInit_
       .asmfunc

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

       ldr   r11,  MSIENA      ;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
       beq   mloop

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


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.

MSTCGSTAT : 메모리 하드웨어 초기화 상태와 메모리 셀프 테스트 상태를 보여준다.

beq를 사용하여, 메모리 초기화가 잘 되었는지를 계속 확인한다.

(beq = branch equal 명령어를 사용하여 for문을 asm에서 함.)

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (_coreEnableEventBusExport_)

```
;
; Enable Event Bus Export
; SourceId : CORE_SourceId_006
; DesignId : CORE_DesignId_007
; Requirements: HL_CONQ_CORE_SR6

    .def     _coreEnableEventBusExport_
    .asmfunc

_coreEnableEventBusExport_

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

    .endasmfunc
```

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 register : privileged mode에서는 레지스터 접근이 항상 가능해야 하기 때문에 위와 같은 코드를 사용함.

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (_coreEnableEventBusExport_)

```
;
; Enable Event Bus Export
; SourceId : CORE_SourceId_006
; DesignId : CORE_DesignId_007
; Requirements: HL_CONQ_CORE_SR6

    .def     _coreEnableEventBusExport_
    .asmfunc

_coreEnableEventBusExport_

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

    .endasmfunc
```

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
```

coreEnableEventBusExport : external bus 허용

위 코드 : privileged mode에서는 레지스터 접근이 항상 가능해야 하기 때문임.

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (_coreEnableEventBusExport_)

```
;
; Enable Event Bus Export
; SourceId : CORE_SourceId_006
; DesignId : CORE_DesignId_007
; Requirements: HL_CONQ_CORE_SR6

    .def     _coreEnableEventBusExport_
    .asmfunc

_coreEnableEventBusExport_

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

    .endasmfunc
```

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
```

coreEnableEventBusExport : external bus 허용

위 코드 : privileged mode에서는 레지스터 접근이 항상 가능해야 하기 때문임.

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – setupPLL())

```
void setupPLL(void)
{

/* USER CODE BEGIN (3) */
/* USER CODE END */

    /* Disable PLL1 and PLL2 */
    systemREG1->CSDISSET = 0x00000002U | 0x00000040U;
    /*SAFETYMCUSW 28 D MR:NA <APPROVED> "Hardware status bit read check" */
    while((systemREG1->CSDIS & 0x42U) != 0x42U)
    {
    /* Wait */
    }

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

systemREG1->CSDISSET = 0x42 = 0100 0010

1번, 6번 bit set = 1 : PLL1, PLL2 disabled

2.5.1.11 Clock Source Disable Set Register (CSDISSET)

The CSDISSET register, shown in Figure 2-18 and described in Table 2-30, sets clock sources to the disabled state.

Figure 2-18. Clock Source Disable Set Register (CSDISSET) (offset = 34h)

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
SETCLKSR7 OFF	SETCLKSR6 OFF	SETCLKSR5 OFF	SETCLKSR4 OFF	SETCLKSR3 OFF	Reserved	SETCLKSR1 OFF	SETCLKSR0 OFF
R/WP-1	R/WP-1	R/WP-0	R/WP-0	R/WP-1	R-1	R/WP-1	R/WP-0

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

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. **Note: 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. **Note: 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).**

NOTE: A list of the available clock sources is shown in the Table 2-29.

6.6 Clocks

6.6.1 *Clock Sources*

Table 6-11 lists the available clock sources on the device. Each clock source can be enabled or disabled using the CSDISx registers in the system module. The clock source number in the table corresponds to the control bit in the CSDISx register for that clock source.

Table 6-11 also lists the default state of each clock source.

Table 6-11. Available Clock Sources

CLOCK SOURCE NO.	NAME	DESCRIPTION	DEFAULT STATE
0	OSCIN	Main Oscillator	Enabled
1	PLL1	Output From PLL1	Disabled
2	Reserved	Reserved	Disabled
3	EXTCLKIN1	External Clock Input 1	Disabled
4	CLK80K	Low-Frequency Output of Internal Reference Oscillator	Enabled
5	CLK10M	High-Frequency Output of Internal Reference Oscillator	Enabled
6	PLL2	Output From PLL2	Disabled
7	EXTCLKIN2	External Clock Input 2	Disabled

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – setupPLL())

```
void setupPLL(void)
{

/* USER CODE BEGIN (3) */
/* USER CODE END */

    /* Disable PLL1 and PLL2 */
    systemREG1->CSDISSET = 0x00000002U | 0x00000040U;
    /*SAFETYMCUSW 28 D MR:NA <APPROVED> "Hardware status bit read check" */
    while((systemREG1->CSDIS & 0x42U) != 0x42U)
    {
    /* Wait */
    }

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

systemREG1->GBLSTAT = 0011 0000 0001

0, 8, 9 번 bit set = 1 :

PLL과 OSC fail behavior 초기화

2.5.1.48 Global Status Register (GLBSTAT)

The GLBSTAT register, shown in Figure 2-55 and described in Table 2-67, indicates the FMzPLL (PLL1) slip status and the oscillator fail status.

NOTE: PLL and OSC fail behavior

The device behavior after a PLL slip or an oscillator failure is configured in the PLLCTL1 register.

Figure 2-55. Global Status Register (GLBSTAT) (offset = ECh)

31	16
Reserved	
R-0	

15	10	9	8	7	1	0
Reserved		FBSLIP	RFSLIP	Reserved		OSCFAIL
R-0		R/W1C-n	R/W1C-n	R-0		R/W1C-n

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to Clear; -n = value after reset

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

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9	FBSLIP		PLL over cycle slip detection. (cleared by nPORRST, maintains its previous value for all other resets).
		0	*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		PLL under cycle slip detection. (cleared by nPORRST, maintains its previous value for all other resets).
		0	*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		Oscillator fail flag bit. (cleared by nPORRST, maintains its previous value for all other resets).
		0	*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.

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – setupPLL())

```
/**    - 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
                     | (uint32)((uint32)0x1FU << 24U)
                     | (uint32)0x00000000U
                     | (uint32)((uint32)(8U - 1U)<< 16U)
                     | (uint32)(0x9500U);
```

2.5.1.25 PLL Control Register 1 (PLLCTL1)

The PLLCTL1 register, shown in Figure 2-32 and described in Table 2-44, controls the output frequency of PLL1 (Clock Source 1 - FMzPLL). It also controls the behavior of the device if a PLL slip or oscillator failure is detected.

Figure 2-32. PLL Control Register 1 (PLLCTL1) (offset = 70h)

31	30	29	28	24
ROS	BPOS		PLLDIV	
R/WP-0	R/WP-1h		R/WP-Fh	

23	22	21	16
ROF	Reserved	REFCLKDIV	
R/WP-0	R-0	R/WP-3h	

15	0
PLLMUL	
R/WP-4100h	

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

Table 2-44. PLL Control Register 1 (PLLCTL1) Field Descriptions

Bit	Field	Value	Description
31	ROS		Reset on PLL Slip.
		0	Do not reset system when PLL slip is detected.
		1	Reset when PLL slip is detected.
			Note: BPOS (Bits 30-29) must also be enabled for ROS to be enabled.
30-29	BPOS		Bypass of PLL Slip.
		2h	Bypass on PLL Slip is disabled. If a PLL Slip is detected no action is taken.
		Others	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.
			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.
28-24	PLLDIV		PLL Output Clock Divider
			R = PLLDIV + 1 $f_{PLL\ CLK} = f_{post\_ODCLK} / R$
		0	$f_{PLL\ CLK} = f_{post\text{-}ODCLK} / 1$
		1h	$f_{PLL\ CLK} = f_{post\text{-}ODCLK} / 2$
		:	:
		1Fh	$f_{PLL\ CLK} = f_{post\text{-}ODCLK} / 32$
23	ROF		Reset on Oscillator Fail.
		0	Do not reset system when oscillator is out of range.
		1	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.
22	Reserved	0	Value has no effect on PLL operation.
21-16	REFCLKDIV		Reference Clock Divider
			NR = REFCLKDIV + 1 $f_{INT\ CLK} = f_{OSCIN} / NR$
		0	$f_{INT\ CLK} = f_{OSCIN} / 1$
		1h	$f_{INT\ CLK} = f_{OSCIN} / 2$
		:	:
		3Fh	$f_{INT\ CLK} = f_{OSCIN} / 64$
15-0	PLLMUL		PLL Multiplication Factor
			NF = (PLLMUL / 256) + 1, valid multiplication factors are from 1 to 256. $f_{VCO\ CLK} = f_{INT\ CLK} \times NF$
		0h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 1$
		100h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 2$
		:	:
		5B00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 92$
		5C00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 93$
		:	:
		FF00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 256$

0011 1111 0000 0111 1001 0101 0000 0000

30-29 : 2h, Bypass of PLL slip
28-24 : 1Fh, f(pll clk) = f(post-odclk)/32
21-16: 7h, f(int clk) = f(oscn)/8
15-0 : 95h , f(vco clk) = f(int clk) x 1

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – setupPLL())

```
/**    - 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)
                     |  (uint32)((uint32)7U << 12U)
                     |  (uint32)((uint32)(1U - 1U) << 9U)
                     |  (uint32)61U;
```

2.5.1.26 PLL Control Register 2 (PLLCTL2)

The PLLCTL2 register, shown in Figure 2-33 and described in Table 2-45, controls the modulation characteristics and the output divider of the PLL.

Figure 2-33. PLL Control Register 2 (PLLCTL2) (offset = 74h)

31	30 – 22	21	20 – 16
FMENA	SPREADINGRATE	Rsvd	MULMOD
R/WP-0	R/WP-1FFh	R-0	R/WP-0

15 – 12	11 – 9	8 – 0
MULMOD	ODPLL	SPR_AMOUNT
R/WP-7h	R/WP-0	R/WP-0

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

0011 1111 1100 0111 0000 0000 0011 1101

30-22 : 255h, f(mod) = f(int clk)/ (2 x 256)
20-12 : No adder to NF
11-9 : 0h, f(post-odclk) = f(vco clk)/1
8-0 : NV = 62/2048

Table 2-45. PLL Control Register 2 (PLLCTL2) Field Descriptions

Bit	Field	Value	Description
31	FMENA		Frequency Modulation Enable.
		0	Disable frequency modulation.
		1	Enable frequency modulation.
30-22	SPREADINGRATE		NS = SPREADINGRATE + 1 $f_{mod} = f_{ss} = f_{INT\ CLK}/(2 \times NS)$
		0	$f_{mod} = f_{ss} = f_{INT\ CLK} / (2 \times 1)$
		1h	$f_{mod} = f_{ss} = f_{INT\ CLK} / (2 \times 2)$
		:	:
		1FFh	$f_{mod} = f_{ss} = f_{INT\ CLK} / (2 \times 512)$
21	Reserved	0	Value has no effect on PLL operation.
20-12	MULMOD		Multiplier Correction when Frequency Modulation is enabled. When FMENA = 0, MUL_when_MOD = 0; when FMENA = 1, MUL_when_MOD = (MULMOD / 256)
		0	No adder to NF.
		8h	MUL_when_MOD = 8/256
		9h	MUL_when_MOD = 9/256
		:	:
		1FFh	MUL_when_MOD = 511/256
11-9	ODPLL		Internal PLL Output Divider OD = ODPLL + 1 $f_{post\text{-}ODCLK} = f_{VCO\ CLK}/OD$ **Note: PLL output clock is gated off, if ODPLL is changed while the PLL is active.**
		0	$f_{post\text{-}ODCLK} = f_{VCO\ CLK} / 1$
		1h	$f_{post\text{-}ODCLK} = f_{VCO\ CLK} / 2$
		:	:
		7h	$f_{post\text{-}ODCLK} = f_{VCO\ CLK} / 8$
8-0	SPR_AMOUNT		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.
		0	NV = 1/2048
		1h	NV = 2/2048
		:	:
		1FFh	NV = 512/2048

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – setupPLL())

```
/**    - 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)
                    | (uint32)((uint32)0x1FU << 24U)
                    | (uint32)((uint32)(8U - 1U)<< 16U)
                    | (uint32)(0x9500U);
```

2.5.2.1 PLL Control Register 3 (PLLCTL3)

The PLLCTL3 register is shown in Figure 2-59 and described in Table 2-72; controls the settings of PLL2 (Clock Source 6 - FPLL).

Figure 2-59. PLL Control Register 3 (PLLCTL3) (offset = 00h)

31 – 29	28 – 24	23 – 22	21 – 16
ODPLL2	PLLDIV2	Reserved	REFCLKDIV2
R/WP-0	R/WP-4h	R-0	R/WP-0

15 – 0
PLLMUL2
R/WP-1300h

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

0001 1111 0000 0111 1001 0101 0000 0000

31-29 : 0h, f(post_odclk2) = f(output_clk2/1)
28-24 : 1Fh, f(pll2 clk) = f(post_odclk2)/32
21-16 : 7h, f(intclk2) – f(oscine)/8
15-0 : 9500h, f(vcoclk2) = f(intclk2) x 150

Table 2-72. PLL Control Register 3 (PLLCTL3) Field Descriptions

Bit	Field	Value	Description
31-29	ODPLL2		Internal PLL Output Divider OD2 = ODPLL2 + 1, ranges from 1 to 8. $f_{post\_ODCLK2} = f_{output\_CLK2} / OD2$ **Note: PLL output clock is gated off if ODPLL2 is changed while the PLL#2 is active.**
		0	$f_{post\_ODCLK2} = f_{output\_CLK2} / 1$
		1h	$f_{post\_ODCLK2} = f_{output\_CLK2} / 2$
		:	:
		7h	$f_{post\_ODCLK2} = f_{output\_CLK2} / 8$
28-24	PLLDIV2		PLL2 Output Clock Divider R2 = PLLDIV2 + 1, ranges from 1 to 32. $f_{PLL2\ CLK} = f_{post\_ODCLK2} / R2$
		0	$f_{PLL2\ CLK} = f_{post\_ODCLK2} / 1$
		1h	$f_{PLL2\ CLK} = f_{post\_ODCLK2} / 2$
		:	:
		1Fh	$f_{PLL2\ CLK} = f_{post\_ODCLK2} / 32$
23-22	Reserved	0	Value has no effect on PLL operation.
21-16	REFCLKDIV2		Reference Clock Divider NR2 = REFCLKDIV2 + 1, ranges from 1 to 64. $f_{INTCLK2} = f_{OSCIN} / NR2$ **Note: This value should not be changed while the PLL2 is active.**
		0	$f_{INTCLK2} = f_{OSCIN} / 1$
		1h	$f_{INTCLK2} = f_{OSCIN} / 2$
		:	:
		3Fh	$f_{INTCLK2} = f_{OSCIN} / 64$
15-0	PLLMUL2		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):
		100h	$f_{VCOCLK2} = f_{INTCLK2} \times 1$
		:	:
		5B00h	$f_{VCOCLK2} = f_{INTCLK2} \times 92$
		5C00h	$f_{VCOCLK2} = f_{INTCLK2} \times 93$
		:	:
		FF00h	$f_{VCOCLK2} = f_{INTCLK2} \times 256$

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – periphInit())

```
void periphInit(void)
{

/* USER CODE BEGIN (9) */
/* USER CODE END */

    /** - Disable Peripherals before peripheral powerup*/
    systemREG1->CLKCNTL &= 0xFFFFFEFFU;

    /** - 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;

/* USER CODE BEGIN (10) */
/* USER CODE END */

}
```

1111 1111 1111 1111 1111 1110 1111 1111

8 bit set = 0 : 전역 페리페럴/페리페럴 메모리 프레임들을 리셋한다.

2.5.1.42 Clock Control Register (CLKCNTL)

The CLKCNTL register, shown in Figure 2-49 and described in Table 2-61, controls peripheral reset and the peripheral clock divide ratios.

NOTE: VCLK and VCLK2 clock ratio restrictions.

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. When increasing the frequency (decreasing the divider), first change the VCLK2R field and then change the VCLKR field. When reducing the frequency (increasing the divider), first change the VCLKR field and then change the VCLK2R field.

You should do a read-back between the two writes. This assures that there are enough clock cycles between the two writes.

Figure 2-49. Clock Control Register (CLKCNTL) (offset = D0h)

31 – 28	27 – 24	23 – 20	19 – 16
Reserved	VCLK2R	Reserved	VCLKR
R-0	R/WP-1h	R-0	R/WP-1h

15 – 9	8	7 – 0
Reserved	PENA	Reserved
R-0	R/WP-0	R-0

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

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		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.**
		0	The VCLK2 speed is HCLK divided by 1.
		:	:
		Fh	The VCLK2 speed is HCLK divided by 16.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	VCLKR		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.**
		0	The VCLK speed is HCLK divided by 1.
		:	:
		Fh	The VCLK speed is HCLK divided by 16.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	PENA		Peripheral enable bit. The application must set this bit before accessing any peripheral.
		0	The global peripheral/peripheral memory frames are in reset.
		1	All peripheral/peripheral memory frames are out of reset.
7-0	Reserved	0	Reads return 0. Writes have no effect.

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – periphInit())

```
void periphInit(void)
{

/* USER CODE BEGIN (9) */
/* USER CODE END */

    /** - Disable Peripherals before peripheral powerup*/
    systemREG1->CLKCNTL &= 0xFFFFFEFFU;

    /** - 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;

/* USER CODE BEGIN (10) */
/* USER CODE END */

}
```

1111 1111 1111 1111 1111 1111 1111 1111

pcrREG1->PSWRDWNCLR0 - 3 :

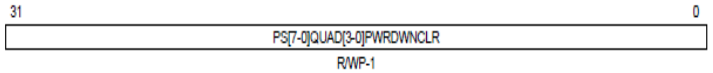
PSPWRDWNSET0-3 ~ PSPWRDWNCLR0 - 3 을 0으로 클리어 한다.

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

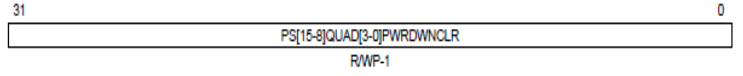
Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0] PWRDWNCLR	0	Peripheral select quadrant clock power-down clear. *Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding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

2.5.3.22 Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-93 and described in Table 2-107.

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 2-93. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) (offset = A4h)



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

Table 2-107. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) Field Descriptions

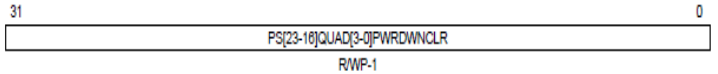
Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0] PWRDWNCLR	0	Peripheral select quadrant clock power-down clear. *Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET1 and PSPWRDWNCLR1 registers is cleared to 0.

2.5.3.23 Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-94 and described in Table 2-108.

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 2-94. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) (offset = A8h)



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

Table 2-108. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) Field Descriptions

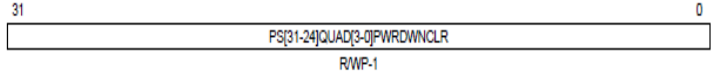
Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0] PWRDWNCLR	0	Peripheral select quadrant clock power-down clear. *Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET2 and PSPWRDWNCLR2 registers is cleared to 0.

2.5.3.24 Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-95 and described in Table 2-109.

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 2-95. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR) (offset = ACh)



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

Table 2-109. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3) Field Descriptions

Bit	Field	Value	Description
31-0	PS[31-24]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 PSPWRDWNSET3 and PSPWRDWNCLR3 registers is cleared to 0.

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – periphInit())

```
void periphInit(void)
{

/* USER CODE BEGIN (9) */
/* USER CODE END */

    /** - Disable Peripherals before peripheral powerup*/
    systemREG1->CLKCNTL &= 0xFFFFFEFFU;

    /** - 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;

/* USER CODE BEGIN (10) */
/* USER CODE END */

}
```

0000 0000 0000 0000 0000 0001 0000 0000

8 bit set = 1 : 전역 페리페럴/페리페럴 메모리 프레임들을 활성화한다.

2.5.1.42 Clock Control Register (CLKCNTL)

The CLKCNTL register, shown in Figure 2-49 and described in Table 2-61, controls peripheral reset and the peripheral clock divide ratios.

NOTE: VCLK and VCLK2 clock ratio restrictions.

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. When increasing the frequency (decreasing the divider), first change the VCLK2R field and then change the VCLKR field. When reducing the frequency (increasing the divider), first change the VCLKR field and then change the VCLK2R field.

You should do a read-back between the two writes. This assures that there are enough clock cycles between the two writes.

Figure 2-49. Clock Control Register (CLKCNTL) (offset = D0h)

31 – 28	27 – 24	23 – 20	19 – 16
Reserved	VCLK2R	Reserved	VCLKR
R-0	R/WP-1h	R-0	R/WP-1h

15 – 9	8	7 – 0
Reserved	PENA	Reserved
R-0	R/WP-0	R-0

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

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		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.**
		0	The VCLK2 speed is HCLK divided by 1.
		:	:
		Fh	The VCLK2 speed is HCLK divided by 16.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	VCLKR		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.**
		0	The VCLK speed is HCLK divided by 1.
		:	:
		Fh	The VCLK speed is HCLK divided by 16.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	PENA		Peripheral enable bit. The application must set this bit before accessing any peripheral.
		0	The global peripheral/peripheral memory frames are in reset.
		1	All peripheral/peripheral memory frames are out of reset.
7-0	Reserved	0	Reads return 0. Writes have no effect.

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – muxInit())

```
void muxInit(void){

/* USER CODE BEGIN (1) */
/* USER CODE END */

    /* Enable Pin Muxing */
    pinMuxReg->KICKER0 = 0x83E70B13U;
    pinMuxReg->KICKER1 = 0x95A4F1E0U;

/* USER CODE BEGIN (2) */
/* USER CODE END */

    pinMuxReg->PINMUX[0] = PINMUX_BALL_N19_AD1EVT | PINMUX_BALL_D4_EMIF_ADDR_00 | PIN

    pinMuxReg->PINMUX[1] = PINMUX_BALL_C5_EMIF_ADDR_07 | PINMUX_BALL_C6_EMIF_ADDR_08

    pinMuxReg->PINMUX[2] = PINMUX_BALL_C9_EMIF_ADDR_11 | PINMUX_BALL_C10_EMIF_ADDR_12

    pinMuxReg->PINMUX[3] = PINMUX_BALL_C13_EMIF_ADDR_15 | PINMUX_BALL_D14_EMIF_ADDR_1

    pinMuxReg->PINMUX[4] = PINMUX_BALL_C15_EMIF_ADDR_19 | PINMUX_BALL_C16_EMIF_ADDR_2

    pinMuxReg->PINMUX[5] = 0U;

    pinMuxReg->PINMUX[6] = 0U;

    pinMuxReg->PINMUX[7] = 0U;

    pinMuxReg->PINMUX[8] = PINMUX_BALL_D16_EMIF_BA_1;

    pinMuxReg->PINMUX[9] = PINMUX_BALL_R4_EMIF_nCAS | PINMUX_BALL_N17_EMIF_nCS_0 | PI

    pinMuxReg->PINMUX[10] = PINMUX_BALL_K17_EMIF_nCS_3 | PINMUX_BALL_M17_EMIF_nCS_4 |

    pinMuxReg->PINMUX[11] = PINMUX_BALL_D17_EMIF_nWE | PINMUX_BALL_E9_ETMDATA_08 | PI
```

0 : 1000 0011 1110 0111 0000 1000 0001 0011
1 : 1001 0101 1010 0100 1111 0001 1110 0000

위와 같이 설정하면, PINMMR registers가 활성화 된다.

6.7.3 KICK_REG0: Kicker Register 0

This register forms the first part of the unlock sequence for being able to update the I/O multiplexing control registers (PINMMRnn).

Figure 6-12. KICK_REG0: Kicker Register 0 (Offset = 38h)

31	16
KICK0	
R/W-0	

15	0
KICK0	
R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

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.

6.7.4 KICK_REG1: Kicker Register 1

This register forms the second part of the unlock sequence for being able to update the I/O multiplexing control registers (PINMMRnn).

Figure 6-13. KICK_REG1: Kicker Register 1 (Offset = 3Ch)

31	16
KICK1	
R/W-0	

15	0
KICK1	
R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

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.

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – muxInit())

```
/**
< 0x110 - 1A4 : Output Pin Multiplexing Control Registers (38 registers)
; 0x250 - 0x29C : Input Pin Multiplexing Control Registers (20)
; 0X390 - 3DC : Special Functionality Control Registers (20) */
```

```
pinMuxReg->PINMUX[0] = PINMUX_BALL_N19_AD1EVT | PINMUX_BALL_D4_EMIF_ADDR_00 | PINMUX_BALL_D5_EMIF_ADDR_01 | PINMUX_BALL_C4_EMIF_
pinMuxReg->PINMUX[1] = PINMUX_BALL_C5_EMIF_ADDR_07 | PINMUX_BALL_C6_EMIF_ADDR_08 | PINMUX_BALL_C7_EMIF_ADDR_09 | PINMUX_BALL_C8_
pinMuxReg->PINMUX[2] = PINMUX_BALL_C9_EMIF_ADDR_11 | PINMUX_BALL_C10_EMIF_ADDR_12 | PINMUX_BALL_C11_EMIF_ADDR_13 | PINMUX_BALL_C
pinMuxReg->PINMUX[3] = PINMUX_BALL_C13_EMIF_ADDR_15 | PINMUX_BALL_D14_EMIF_ADDR_16 | PINMUX_BALL_C14_EMIF_ADDR_17 | PINMUX_BALL_
pinMuxReg->PINMUX[4] = PINMUX_BALL_C15_EMIF_ADDR_19 | PINMUX_BALL_C16_EMIF_ADDR_20 | PINMUX_BALL_C17_EMIF_ADDR_21;
pinMuxReg->PINMUX[5] = 0U;
pinMuxReg->PINMUX[32] = PINMUX_BALL_A13_N2HET1_17 | PINMUX_BALL_J1_N2HET1_18 | PINMUX_BALL_B13_N2HET1_19 | PINMUX_BALL_P2_N2HET
pinMuxReg->PINMUX[33] = PINMUX_BALL_H4_N2HET1_21 | PINMUX_BALL_B3_N2HET1_22 | PINMUX_BALL_J4_N2HET1_23 | PINMUX_BALL_P1_N2HET1_
pinMuxReg->PINMUX[34] = PINMUX_BALL_A14_N2HET1_26 | PINMUX_BALL_K19_N2HET1_28 | PINMUX_BALL_B11_N2HET1_30 | PINMUX_BALL_D8_N2HE
pinMuxReg->PINMUX[35] = PINMUX_BALL_D7_N2HET2_02 | PINMUX_BALL_D3_N2HET2_12 | PINMUX_BALL_D2_N2HET2_13 | PINMUX_BALL_D1_N2HET2_
pinMuxReg->PINMUX[36] = PINMUX_BALL_P4_N2HET2_19 | PINMUX_BALL_T5_N2HET2_20 | PINMUX_BALL_T4_MII_RXCLK | PINMUX_BALL_U7_MII_TX_
pinMuxReg->PINMUX[37] = PINMUX_BALL_E2_N2HET2_03 | PINMUX_BALL_N3_N2HET2_07;
pinMuxReg->PINMUX[80] = (SIGNAL_AD2EVT_T10 | 0x02020200U);
pinMuxReg->PINMUX[81] = 0x02020202U;
pinMuxReg->PINMUX[82] = 0x02020202U;
pinMuxReg->PINMUX[83] = (SIGNAL_GIOA_0_A5 | 0x00020202U);
pinMuxReg->PINMUX[84] = SIGNAL_GIOA_1_C2 | SIGNAL_GIOA_2_C1 | SIGNAL_GIOA_3_E1 | SIGNAL_GIOA_4_A6;
pinMuxReg->PINMUX[93] = SIGNAL_N2HET1_25_M3 | SIGNAL_N2HET1_27_A9 | SIGNAL_N2HET1_29_A3 | SIGNAL_N2HET1_31_J17;
pinMuxReg->PINMUX[94] = SIGNAL_N2HET2_00_D6 | SIGNAL_N2HET2_01_D8 | SIGNAL_N2HET2_02_D7 | SIGNAL_N2HET2_03_E2;
pinMuxReg->PINMUX[95] = SIGNAL_N2HET2_04_D13 | SIGNAL_N2HET2_05_D12 | SIGNAL_N2HET2_06_D11 | SIGNAL_N2HET2_07_N3;
pinMuxReg->PINMUX[96] = SIGNAL_N2HET2_08_K16 | SIGNAL_N2HET2_09_L16 | SIGNAL_N2HET2_10_M16 | SIGNAL_N2HET2_11_N16;
pinMuxReg->PINMUX[97] = SIGNAL_N2HET2_12_D3 | SIGNAL_N2HET2_13_D2 | SIGNAL_N2HET2_14_D1 | SIGNAL_N2HET2_15_K4;
pinMuxReg->PINMUX[98] = SIGNAL_N2HET2_16_L4 | SIGNAL_N2HET2_18_N4 | SIGNAL_N2HET2_20_T5 | SIGNAL_N2HET2_22_T7;
pinMuxReg->PINMUX[99] = SIGNAL_nTZ1_1_N19 | SIGNAL_nTZ1_2_F1 | SIGNAL_nTZ1_3_J3;
pinMuxReg->PINMUX[161] = 0x02020200U;
pinMuxReg->PINMUX[162] = 0x02020202U;
pinMuxReg->PINMUX[163] = 0x00020202U;
```

❖ Mux 레지스터 setting

Output Pin Control registers – 38개
Input Pin Control register – 20개
Special Functionality Control register – 20개

PWM pin – 7개 사용할 수 있음!

```
PINMUX_GATE_EMIF_CLK_ENABLE(OFF);
PINMUX_EMIF_OUTPUT_ENABLE(OFF);
PINMUX_GIOA_DISABLE_HET1_ENABLE(OFF);
PINMUX_GIOB_DISABLE_HET2_ENABLE(OFF);
PINMUX_ETHERNET_SELECT(MII);
PINMUX_ALT_ADC_TRIGGER_SELECT(1);

PINMUX_ETPWM1_EQEPERR_ENABLE(EQEPERR12);
PINMUX_ETPWM2_EQEPERR_ENABLE(EQEPERR12);
PINMUX_ETPWM3_EQEPERR_ENABLE(EQEPERR12);
PINMUX_ETPWM4_EQEPERR_ENABLE(EQEPERR12);
PINMUX_ETPWM5_EQEPERR_ENABLE(EQEPERR12);
PINMUX_ETPWM6_EQEPERR_ENABLE(EQEPERR12);
PINMUX_ETPWM7_EQEPERR_ENABLE(EQEPERR12);
PINMUX_ETPWM_TIME_BASE_SYNC_ENABLE(OFF);
PINMUX_ETPWM_TZ1_ENABLE(ASYNC);
PINMUX_ETPWM_TZ2_ENABLE(ASYNC);
PINMUX_ETPWM_TZ3_ENABLE(ASYNC);
PINMUX_ETPWM_EPWM1SYNCI_ENABLE(ASYNC);

PINMUX_ETPWM_SOC1A_ENABLE(ON);
PINMUX_ETPWM_SOC2A_ENABLE(ON);
PINMUX_ETPWM_SOC3A_ENABLE(ON);
PINMUX_ETPWM_SOC4A_ENABLE(ON);
PINMUX_ETPWM_SOC5A_ENABLE(ON);
PINMUX_ETPWM_SOC6A_ENABLE(ON);
PINMUX_ETPWM_SOC7A_ENABLE(ON);

PINMUX_EQEP1A_FILTER_ENABLE(OFF);
PINMUX_EQEP1B_FILTER_ENABLE(OFF);
PINMUX_EQEP1I_FILTER_ENABLE(OFF);
PINMUX_EQEP1S_FILTER_ENABLE(OFF);
PINMUX_EQEP2A_FILTER_ENABLE(OFF);
PINMUX_EQEP2B_FILTER_ENABLE(OFF);
```

1. Cortex-R5F Hercules Safety MCU – boot code 분석 (systemInit() – muxInit())

```
    pinMuxReg->PINMUX[174] |= (uint32)(~(0XFEFFFFFFU));

    PINMUX_TEMP1_ENABLE(OFF);
    PINMUX_TEMP2_ENABLE(OFF);
    PINMUX_TEMP3_ENABLE(OFF);

    /* Disable Pin Muxing */
    pinMuxReg->KICKER0 = 0x00000000U;
    pinMuxReg->KICKER1 = 0x00000000U;

/* USER CODE BEGIN (4) */
/* USER CODE END */
}
```

Mux 설정이 완료되면

Mux에 다른 값이 셋팅되지 못하도록

Mux를 disable해 둔다.

MUX의 기능 →

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

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
144h	E10	ETMDATA[15]	13[0]	EMIF_nDQM[0]	13[1]								
	K15	ETMDATA[16]	13[8]	EMIF_DATA[00]	13[9]								
	L15	ETMDATA[17]	13[16]	EMIF_DATA[01]	13[17]								
	M15	ETMDATA[18]	13[24]	EMIF_DATA[02]	13[25]								
148h	N15	ETMDATA[19]	14[0]	EMIF_DATA[03]	14[1]								
	E5	ETMDATA[20]	14[8]	EMIF_DATA[04]	14[9]								
	F5	ETMDATA[21]	14[16]	EMIF_DATA[05]	14[17]								
	G5	ETMDATA[22]	14[24]	EMIF_DATA[06]	14[25]								
14Ch	K5	ETMDATA[23]	15[0]	EMIF_DATA[07]	15[1]								
	L5	ETMDATA[24]	15[8]	EMIF_DATA[08]	15[9]	N2HET2[24]	15[10]	MIBSPI5NCS[4]	15[11]				
	M5	ETMDATA[25]	15[16]	EMIF_DATA[09]	15[17]	N2HET2[25]	15[18]	MIBSPI5NCS[5]	15[19]				
	N5	ETMDATA[26]	15[24]	EMIF_DATA[10]	15[25]	N2HET2[26]	15[26]						
150h	P5	ETMDATA[27]	16[0]	EMIF_DATA[11]	16[1]	N2HET2[27]	16[2]						
	R5	ETMDATA[28]	16[8]	EMIF_DATA[12]	16[9]	N2HET2[28]	16[10]	GIOA[0]	16[11]				
	R6	ETMDATA[29]	16[16]	EMIF_DATA[13]	16[17]	N2HET2[29]	16[18]	GIOA[1]	16[19]				
	R7	ETMDATA[30]	16[24]	EMIF_DATA[14]	16[25]	N2HET2[30]	16[26]	GIOA[3]	16[27]				
154h	R8	ETMDATA[31]	17[0]	EMIF_DATA[15]	17[1]	N2HET2[31]	17[2]	GIOA[4]	17[3]				
	R9	ETMTRACECLKIN	17[8]	EXTCLKIN2	17[9]			GIOA[5]	17[11]				
	R10	ETMTRACECLKOUT	17[16]					GIOA[6]	17[19]				
	R11	ETMTRACECTL	17[24]					GIOA[7]	17[27]				
158h	B15	FRAYTX1	18[0]					GIOA[2]	18[3]				
	B8	FRAYTX2	18[8]					GIOB[0]	18[11]				
	B16	FRAYTXEN1	18[16]					GIOB[1]	18[19]				
	B9	FRAYTXEN2	18[24]					GIOB[2]	18[27]				
15Ch	C1	GIOA[2]	19[0]			N2HET2[00]	19[2]					ePQEP2I	19[5]
	E1	GIOA[3]	19[8]			N2HET2[02]	19[10]						
	B5	GIOA[5]	19[16]					EXTCLKIN	19[19]			ePWM1A	19[21]
	H3	GIOA[6]	19[24]			N2HET2[04]	19[26]					ePWM1B	19[29]
160h	M1	GIOA[7]	20[0]			N2HET2[06]	20[2]					ePWM2A	20[5]
	F2	GIOB[2]	20[8]					DCAN4TX	20[11]				
	W10	GIOB[3]	20[16]					DCAN4RX	20[19]				
	J2	GIOB[5]	20[24]	nERROR1	20[25]								
164h	F1	GIOB[7]	21[0]	nERROR2	21[1]							nTZ1_2	21[5]
	R2	MIBSPI1NCS[0]	21[8]	MIBSPI1SOMI[1]	21[9]	MII_TXD[2]	21[10]					ECAP6	21[13]
	F3	MIBSPI1NCS[1]	21[16]			MII_COL	21[18]	N2HET1[17]	21[19]			eQEP1S	21[21]
	G3	MIBSPI1NCS[2]	21[24]			MDIO	21[26]	N2HET1[19]	21[27]				