TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – 문한나

mhn97@naver.com

부팅코드 분석해보기

Getting Started | HL_sys_startup.c | HL_sys_core.h | HL_sys_core.asm

```c

/* 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_();

/* USER CODE BEGIN (7) */
/* USER CODE END */

/* USER CODE BEGIN (8) */
/* USER CODE END */

```

coreInitRegisters();

Getting Started | HL_sys_startup.c | HL_sys_core.h | HL_sys_core.asm

```c
*   @note: Use this macro for UNDEF Mode Stack length (in bytes)
*/
#define UNDEF_STACK_LENGTH  0x00000100U

/* System Core Interface Functions */

/** @fn void _coreInitRegisters_(void)
*   @brief Initialize Core register
*/
void _coreInitRegisters_(void);

/** @fn void _coreInitStackPointer_(void)
*   @brief Initialize Core stack pointer
*/
void _coreInitStackPointer_(void);

/** @fn void _getCPSRValue_(void)
*   @brief Get CPSR Value
*/
uint32 _getCPSRValue_(void);

/** @fn void _checkMemInitOn_(void)
*   @brief Wait until Mem Init is complete if initiated already.
*/
void _checkMemInitOn_(void);

/** @fn void _gotoCPUIdle_(void)
*   @brief Take CPU to Idle state
*/
void _gotoCPUIdle_(void);

/** @fn void _coreEnableIrqVicOffset_(void)
*   @brief Enable Irq offset propagation via Vic controller
*/
void _coreEnableIrqVicOffset_(void);

/** @fn void _coreEnableVfp_(void)
*   @brief Enable vector floating point unit
```

어셈블리어를 봐야함

Getting Started | HL_sys_startup.c | HL_sys_core.h | HL_sys_core.asm

```asm


_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 r0, lr

: 복귀 주소를 r0에 넣음

mov r1, #0x0000

: 초기화

mrs r1, cpsr

: cpsr 을 r1 레지스터에 저장

msr spsr_ cxsf, r1

: r1에서 읽어 온 데이터를 spsr_ cxsf 에 넣어줌

(spsr 레지스터는 모드마다 하드웨어적으로 하나씩 구현되어 있음)

; Switch to FIQ mode (M = 10001)

: M = 10001로 모드 변경

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.

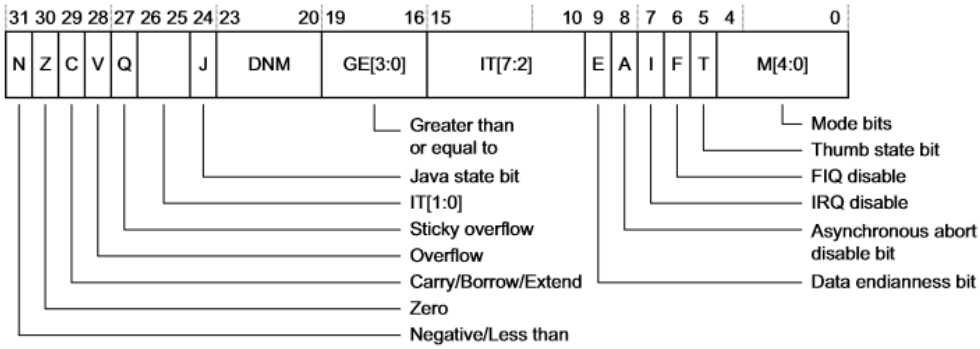

Figure 3-4 Program status register

3.7.11 The M bits

M[4:0] are the mode bits. These bits determine the processor operating mode as Table 3-3 shows.

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

각각 모드마다 사용하는 레지스터가 다름. 그래서 사용하는 레지스터 초기화 하는 작업

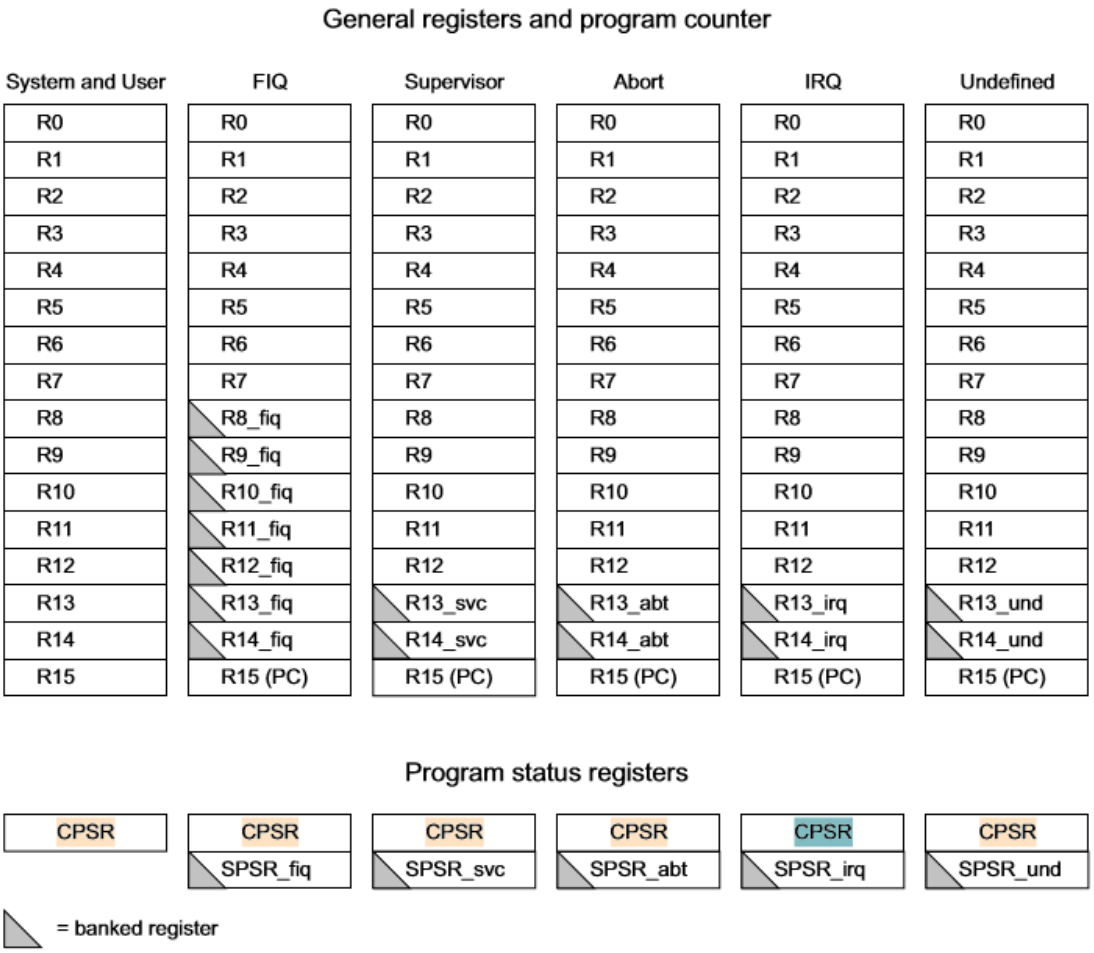

Figure 3-3 Register organization

Getting Started | HL_sys_startup.c | HL_sys_core.h | HL_sys_core.asm

```
100
101
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
107
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
124         bl    next1
125 next1
126         bl    next2
127 next2
128         bl    next3
129 next3
130         bl    next4
131 next4
132         bx    r0
133
134     .endasmfunc
```

```
mrc   p15,    #0x00,    r2,     c1, c0, #0x02
orr   r2,     r2,       #0xF00000
mcr   p15,    #0x00,    r2,     c1, c0, #0x02
```

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
```

CPACR에 접근하기 위한 코드

[23:22]	cp11	Defines access permissions for the FPU. If the FPU is not included for this processor, these bits are RAZ/WI. 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.
[21:22]	cp10	

#0xF00000 21~23번째 비트를 1로 세팅

따라서 b11 = Privileged and User mode access

```
mov   r2,     #0x40000000
fmxr  fpexc,  r2
```

0100이니까 30번 비트를 보면 됨

11.3.3 Floating-Point Exception Register, FPEXC

The FPEXC Register characteristics are:

Purpose Provides global enable and disable control of the VFP extension, and indicate how the state of this extension is recorded.

Usage constraints
- The FPEXC Register is accessible in Privileged modes only.
- Clearing EN disables VFP functionality, causing all VFP instructions apart from privileged system register accesses to generate an Undefined Instruction exception.

Configurations Use this register if the device is configured as a Cortex-R5F processor.

Attributes See Table 11-6.

Figure 11-4 shows the bit assignments.

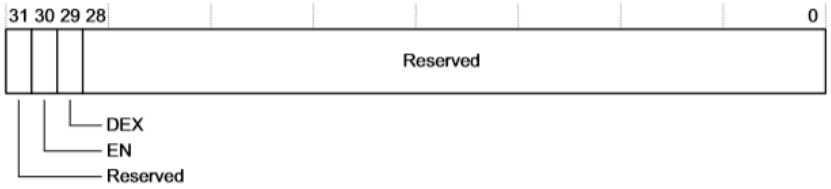

Figure 11-4 FPEXC Register bit assignments

Table 11-6 shows the bit assignments.

Table 11-6 FPEXC Register bit assignments

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.

VFP를 가능하게 함. 사용한 후 0으로 다시 바꿈

```
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
```

32 bit 두개를 64 bit 로 사용(더블 전용 레지스터)

전부 0으로 초기화

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

31–28	27	26	25	24	23	22	21	20	19–16	15–12	11	10	9	8	7	6	5	4	3–0
cond	1	1	0	0	0	1	0	0	Rn	Rd	1	0	1	1	0	0	0	1	Dm

```
bl    next1
next1
        bl    next2
next2
        bl    next3
next3
        bl    next4
```

위에서 하고 있는 설정(fmddr)이 끝날 때까지 시간을 벌기 위한 코드

Call 하면 파이프라인이 깨지기 때문

```
next4
        bx    r0
```

r0 로 점프. r0 에는 복귀주소가 들어가 있음

coreInitRegisters(); 끝!!

```c
void _coreInitStackPointer_(void);
```

Getting Started | HL_sys_startup.c | HL_sys_core.h | HL_sys_core.asm

```asm

_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

    .endasmfunc
```

위에서 설정한 모드 각각 스택 공간을 할당해줌

User -> 4096 / 나머지 -> 256

.endasmfunc

함수 끝!!

```c
switch(getResetSource())
{
    case POWERON_RESET:
    case DEBUG_RESET:
    case EXT_RESET:
```

Getting Started | HL_sys_startup.c | HL_sys_core.h | HL_sys_core.asm | HL_system.c

```c
/* Requirements : HL_CONQ_SYSTEM_SR9 */
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;
    }
    else if ((SYS_EXCEPTION & (uint32)EXT_RESET) != 0U)
    {
        /* Reset caused due to External reset. */
        rst_source = EXT_RESET;
        SYS_EXCEPTION = (uint32)EXT_RESET;
    }
```

SYS_EXCEPTION

```c
#define SYS_DOZE_MODE        0x000F3F02U
#define SYS_SNOOZE_MODE      0x000F3F03U
#define SYS_SLEEP_MODE       0x000FFFFFU
#define LPO_TRIM_VALUE       (((*(volatile uint32 *)0xF00801B4U) & 0xFFFF0000U)>>16U)
#define SYS_EXCEPTION        (*(volatile uint32 *)0xFFFFFFE4U)
```


Table 2-18. Primary System Control Registers (continued)

Offset	Acronym	Register Description	Section
BCh	SSIR4	System Software Interrupt Request 4 Register	Section 2.5.1.38
C0h	RAMGCR	RAM Control Register	Section 2.5.1.39
C4h	BMMCR1	Bus Matrix Module Control Register 1	Section 2.5.1.40
CCh	CPURSTCR	CPU Reset Control Register	Section 2.5.1.41
D0h	CLKCNTL	Clock Control Register	Section 2.5.1.42
D4h	ECPCNTL	ECP Control Register	Section 2.5.1.43
DCh	DEVCR1	DEV Parity Control Register 1	Section 2.5.1.44
E0h	SYSECR	System Exception Control Register	Section 2.5.1.45
E4h	SYSESR	System Exception Status Register	Section 2.5.1.46
E8h	SYSTASR	System Test Abort Status Register	Section 2.5.1.47
ECh	GLBSTAT	Global Status Register	Section 2.5.1.48
F0h	DEVID	Device Identification Register	Section 2.5.1.49
F4h	SSIVEC	Software Interrupt Vector Register	Section 2.5.1.50
F8h	SSIF	System Software Interrupt Flag Register	Section 2.5.1.51

2.5.1.46 System Exception Status Register (SYSESR)

The SYSESR register, shown in Figure 2-53 and described in Table 2-65, shows the source for different resets encountered. Previous reset source status bits are not automatically cleared if new resets occur. After reading this register, the software should clear any flags that are set so that the source of future resets can be determined. Any bit in this register can be cleared by writing a 1 to the bit.

Figure 2-53. System Exception Status Register (SYSESR) (offset = E4h)

31						16
Reserved						
R-0						

15	14	13	12	11	10	8
PORST	OSCRST	WDRST	Reserved	DBGRST	Reserved	
R/WC-X	R/WC-X*	R/WC-X*	R-0	R/WC-X*	R-0	

7	6	5	4	3	2	0
ICSTRST	Reserved	CPURST	SWRST	EXTRST	Reserved	
R/WC-X*	R/WC-X*	R/WC-X*	R/WC-X*	R/WC-X*	R-0	

LEGEND: R/W = Read/Write; R = Read only; C = Clear; X = value is unchanged after reset; X* = 0 after PORST but unchanged after other resets; -*n* = value after reset

```
if ((SYS_EXCEPTION & (uint32)POWERON_RESET) != 0U)
```

POWERON_RESET

```
105 */
106 typedef enum
107 {
108     POWERON_RESET        = 0x8000U,  /**< Alias for Power On Reset     */
109     OSC_FAILURE_RESET    = 0x4000U,  /**< Alias for Osc Failure Reset  */
110     WATCHDOG_RESET       = 0x2000U,  /**< Alias for Watch Dog Reset    */
111     WATCHDOG2_RESET      = 0x1000U,  /**< Alias for Watch Dog 2 Reset  */
112     DEBUG_RESET          = 0x0800U,  /**< Alias for Debug Reset        */
113     INTERCONNECT_RESET   = 0x0080U,  /**< Alias for Interconnect Reset */
114     CPU0_RESET           = 0x0020U,  /**< Alias for CPU 0 Reset        */
115     SW_RESET             = 0x0010U,  /**< Alias for Software Reset     */
116     EXT_RESET            = 0x0008U,  /**< Alias for External Reset     */
117     NO_RESET             = 0x0000U   /**< Alias for No Reset           */
118 }resetSource_t;
119
```

0x8000U 15 번째 비트 세팅

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

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15	PORST		Power-on reset. This bit is set when a power-on reset occurs, either internally asserted by the VMON or externally asserted by the nPORRST pin.
		0	No power-on reset has occurred since this bit was last cleared.
		1	A reset was caused by a power-on reset. (This bit should be cleared after being read so that subsequent resets can be properly identified as not being power-on resets.)
14	OSCRST		Reset caused by an oscillator failure or PLL cycle slip. This bit is set when a reset is caused by an oscillator failure or PLL slip. Write 1 will clear this bit. Write 0 has no effect.
			Note: The action taken when an oscillator failure or PLL slip is detected must configured in the PLLCTL1 register.
		0	No reset has occurred due to an oscillator failure or a PLL cycle slip.
		1	A reset was caused by an oscillator failure or a PLL cycle slip.
13	WDRST		Watchdog reset flag. This bit is set when the last reset was caused by the digital watchdog (DWD). Write 1 will clear this bit. Write 0 has no effect.
		0	No reset has occurred because of the DWD.
		1	A reset was caused by the DWD.
12	Reserved	0	Reads return 0. Writes have no effect.
11	DBGRST		Debug reset flag. This bit is set when the last reset was caused by the debugger reset request. Write 1 will clear this bit. Write 0 has no effect.
		0	No reset has occurred because of the debugger.
		1	A reset was caused by the debugger.
10-8	Reserved	0	Reads return 0. Writes have no effect.
7	ICSTRST		Interconnect reset flag. This bit is set when the last CPU reset was caused by the entering and exiting of interconnect self-test check. While the interconnect is under self-test check, the CPU is also held in reset until the interconnect self-test is complete.
		0	No CPUx reset has occurred because of an interconnect self-test check.
		1	A reset has occurred to the CPUx because of the interconnect self-test check.
6	Reserved	0	Reads return 0. Writes have no effect.

```
rst_source = POWERON_RESET;
```

```
rst_source = 0x8000U
```

```
SYS_EXCEPTION = 0x0000FFFFU;
```

```
#define SYS_DOZE_MODE        0x000F3F02U
#define SYS_SNOOZE_MODE      0x000F3F03U
#define SYS_SLEEP_MODE       0x000FFFFFU
#define LPO_TRIM_VALUE       (((*(volatile uint32 *)0xF00801B4U) & 0xFFFF0000U)>>16U)
#define SYS_EXCEPTION        (*(volatile uint32 *)0xFFFFFFE4U)
```

0~15 번비트까지 1 로 세팅

```
    else
    {
        /* No_reset occured. */
        rst_source = NO_RESET;
    }

    return rst_source;
}
```

```
return rst_source;
```

0x8000 리턴한다

```
119
120         /* Initialize L2RAM to avoid ECC errors right after power on */
121         _memInit_();
122
```

Getting Started | HL_sys_startup.c | HL_sys_core.h | HL_sys_core.asm | HL_system.c | HL_system.h

```
590 ;-----------------------------------------------------------------------------
591 ;Initialize RAM memory
592
593         .def _memInit_
594         .asmfunc
595
596 _memInit_
597         ldr   r12, MINITGCR      ;Load MINITGCR register address
598         mov   r4, #0xA
599         str   r4, [r12]          ;Enable global memory hardware initialization
600
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 initalisation for SRAM
604 mloop                            ;Loop till memory hardware initialization comletes
605         ldr   r5, MSTCGSTAT
606         ldr   r4, [r5]
607         tst   r4, #0x100
608         beq   mloop
609
610         mov   r4, #5
611         str   r4, [r12]          ;Disable global memory hardware initialization
612         bx lr
613         .endasmfunc
614
615 ;-----------------------------------------------------------------------------
```

ldr r12, MINITGCR

```
90    uint32 MSTGCR;                /* 0x0058 */
91    uint32 MINITGCR;              /* 0x005C */
92    uint32 MSINENA;               /* 0x0060 */
```

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)

31 – 16	15 – 4	3 – 0
Reserved	Reserved	MINITGENA
R-0	R-0	R/WP-5h

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.

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

Global memory hardware initialization is enabled.

```
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

Write: The bit is cleared to 0.
Write: A write of 0 has no effect.

beq   mloop
```

r4 와 #0x100 의 &연산 한 값이 0 이면 계속 루프를 돈다.(8 번 비트가 1 이 될 때 까지 계속 돔)

Read: Hardware initialization of all memory is completed.

```
uint32 MSTFAIL;                /* 0x0064 */
uint32 MSTCGSTAT;              /* 0x0068 */
uint32 MINISTAT;               /* 0x006C */
```

System and Peripheral Control Registers www.ti.com

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	15–9	8	7–1	0
Reserved	Reserved	MINIDONE	Reserved	MSTDONE
R-0	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	0	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.** *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	0	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.** *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.

하드웨어를 전부 초기화한 후 루프를 빠져나감

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

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.

Global memory hardware initialization is disabled.

bx lr

복귀주소로 돌아감!!

Getting Started | HL_sys_startup.c | HL_sys_core.h | HL_sys_core.asm | HL_system.c | HL_system.h | HL_reg_system.h

```
134         /* This allows the CPU to signal any single-bit or double-bit errors detected
135          * by its ECC logic for accesses to program flash or data RAM.
136          */
137         _coreEnableEventBusExport_();
138
139 /* USER CODE BEGIN (10) */
140 /* USER CODE END */
141
142         /* Check if there were ESM group3 errors during power-up.
143          * These could occur during eFuse auto-load or during reads from flash OTP
144          * during power-up. Device operation is not reliable and not recommended
145          * in this case. */
146         if ((esmREG->SR1[2]) != 0U)
147         {
148            esmGroup3Notification(esmREG,esmREG->SR1[2]);
149         }
150
151         /* Initialize System - Clock, Flash settings with Efuse self check */
152         systemInit();
153
154 /* USER CODE BEGIN (11) */
155 /* USER CODE END */
156
157         /* Enable IRQ offset via Vic controller */
158         _coreEnableIrqVicOffset_();
159
160         /* Initialize VIM table */
161         vimInit();
162
163 /* USER CODE BEGIN (12) */
164 /* USER CODE END */
165         /* Configure system response to error conditions signaled to the ESM group1 */
166         /* This function can be configured from the ESM tab of HALCoGen */
167         esmInit();
168
169 /* USER CODE BEGIN (13) */
170 /* USER CODE END */
171
```

coreEnableEventBusExport();

```
    .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
```