TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-05-04 (48회차)

강사 - Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 - 정유경

ucong@naver.com

< Coretex R5 부트코드 분석 >

Reference

- Cortex R5 (revision r1p2) Technical Reference Manual
- (spnu563a)TMS570LC4357 Hercules MCU Technical Reference Manual

코드분석

Code Compiler Studio를 켜고, 이전에 실행했던 External_Led예제를 연다
먼저 부트코드가 있는 HL_sys_intvecs.asm을 본다

리셋 엔트리 _c_int00로 분기한다 -> F3

HL_sys_startup.c

coreInitRegisters_()

아무것도 안뜨고 헤더파일만 뜬다.
어셈블리어로 되어있기 때문에 여기서는 볼 수 가 없다.

*. R14(linked register, lr): 어디에서 branch했는지 알려준다
R13(stack pointer, sp): 현재 스택이 쌓여있는 위치
R15(program counter, pc): 현재 instruction을 fetch해 온 위치
*. CPSR(current program status register)
[31:28] N(Negative),Z(Zero),C(Carry),V(oVerflow)
[7] IRQ, [6] FIQ
[5] Thumb or ARM mode 결정
[4:0] modes
*. SPSR(saved program status register): Mode change할 때 cpsr을 백업한다. 이전 모드로 돌아갈 때 사용한다

HL_sys_core.asm

```
; Initialize CPU Registers
; SourceId : CORE_SourceId_001
; DesignId : CORE_DesignId_001
; Requirements: HL_CORE_CORE_SR2

    .def    _coreInitRegisters_
    .asmfunc

_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)
```

리셋 직후 CPU는 커널모드에 있다.

복귀주소를 r0에 저장하고 r1~r13까지의 레지스터를 0으로 초기화한다.

cpsr레지스터의 값을 r1에 저장하고 다시 이 r1의 값을 spsr 레지스터에 저장한다.

supervisor mode에서 FIQ mode로 변경한다.

```
        ; 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
```

CPS: 동작모드를 변경한다. M = 10001b = 17 = FIQ모드

FIQ모드에서 사용하는 레지스터인 r8~r12를 초기화한다.*/

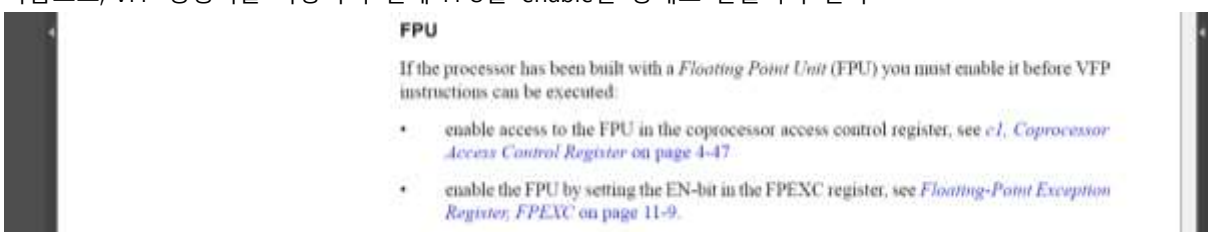
// 이번엔 r0를 lr레지스터에 저장하고 있다 ??

r1에 cpsr register의 값을 저장한 후 그 값을 FIQ 모드에 있는 spsr레지스터에 저장한다

아래는 동일한 동작을 반복하고 있다.

즉, cps명령어로 모드를 변경해주면서, r0값을 lr에 저장하고, cpsr레지스터의 값을 spsr레지스터에 저장한다

다음으로, VFP 명령어를 사용하기 전에 FPU를 enable한 상태로 만들어야 한다



FPU

If the processor has been built with a *Floating Point Unit* (FPU) you must enable it before VFP instructions can be executed:

- enable access to the FPU in the coprocessor access control register, see *c1, Coprocessor Access Control Register* on page 4-47
- enable the FPU by setting the EN-bit in the FPEXC register, see *Floating-Point Exception Register, FPEXC* on page 11-9.

따라서 CPACR, FREXC 레지스터를 설정해준다.

*. CPACR(Coprocessor Access Control Register): Sets access rights for coprocessors : The only coprocessor that the Cortex-R5F CPU includes is the FPU, CP10, and CP11. This register enables software to determine if the FPU exists in the CPU.

// c1, c0, 2 로 찾는다

To access the CPACR, read or write CP15 with:

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

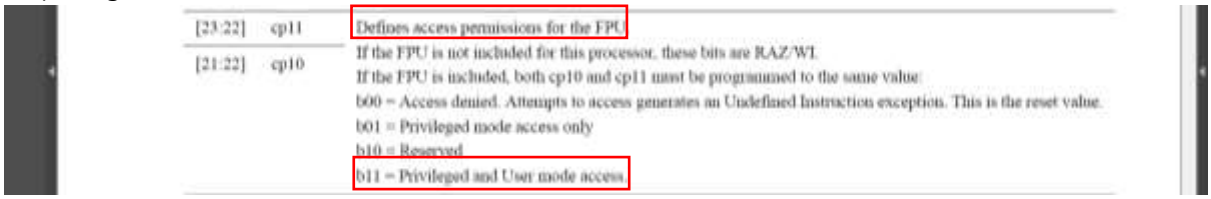
MCR p15, 0, <Rd>, c1, c0, 2 ; Write CPACR

임을 알 수 있다. 따라서 코드가 의미하는 바는 다음과 같다.

CPACR을 read하여 r2에 저장한다. r2와 0xF00000를 or 연산한 결과를 r2에 저장한다.

r2 값을 CPACR에 write. 결과적으로 [23:20]에 1111이 저장된다.

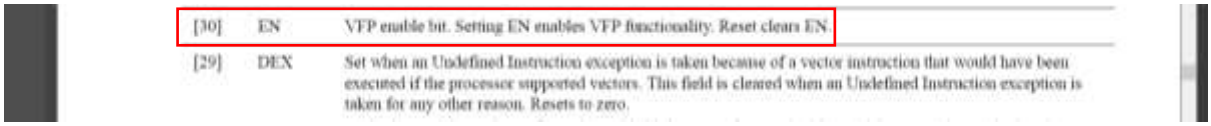
즉, privileged와 user mode에서 FPU(부동소수점연산)을 사용하겠다는 의미*/



[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	

fmxr명령어를 사용하여 r2의 값을 fpexc레지스터에 저장한다. r2 = (0100)000 0000

즉, 30번 비트를 1로 set한다.



[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.

*. Fpexc 레지스터: Floating-Point Exception Register *. Fmxr 명령어: Transfer contents between an ARM register and a VFP system register.

double(8바이트= 64비트)형 레지스터인 VFP 레지스터에 두개의 32비트 범용 레지스터의 값을 저장하여 조기화해주고 있다. Fmdrr <Dm> <Rd> <Rn>에서 뒤쪽의<Rn>이 상위 32비트를 담당하고, 앞쪽의 <Rd>가 하위 32비트를 담당하므로 주의하자.

```
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
```

의미없는 코드처럼 보이지만 사실 fmdrr명령어가 수행되기 위한 시간을 벌어주는 코드이다. 즉 delay를 하기 위하여 4clock을 소모한 것이다.

(분기시 파이프라인이 깨지므로 stall이 일어나는 것을 이용하였다)

*. B, BL, BX, BLS : 분기 명령어(점프) B, BL [점프할 label], BX, BLX [점프할 Register]

HL_sys_startup.c

```
95 void _c_int00(void)
96 {
97
98 /* USER CODE BEGIN (5) */
99 /* USER CODE END */
100
101     /* Initialize Core Registers to avoid CCM Error */
102     _coreInitRegisters_();
103
104     /* Initialize Stack Pointers */
105     _coreInitStackPointer_();
106
107     /* Reset handler: the following instructions read from the system exception status register
108      * to identify the cause of the CPU reset.
109      */
110
```

coreInitStackPointer(); // F3 x - 어셈으로 되어있으므로 HL_sys_core.asm를 검색

HL_sys_core.asm

동작모드별로 별도의 스택을 가지고 있다.

```
143     .def    _coreInitStackPointer_
144     .asmfunc
145
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
170
```

동작모드를 바꾸어 가면서 각 모드별로 sp를 설정하고 있다.

동작모드별 스택을 따로 잡아주는 이유: 인터럽트시 모두 꺼지면 안되므로

userSp .word는 #define 과 같은 의미이다.

0x08000000을 기준으로 0x1000을 user mode stack의 top으로 하여 스택공간을 잡았다.

즉, user mode에 4096bytes(4KB), 나머지 mode에는 256bytes를 스택공간으로 할당하였다

HL_sys_startup.c

```
/* 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:
```

Poweron_Reset만 신경쓰면 된다.

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;
    }
    else if ((SYS_EXCEPTION & (uint32)EXT_RESET) != 0U)
    {
```

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

따라서 SYS_EXCEPTION 은 0xFFFFFFE4U 주소의 값을 제어하는 포인터임을 알 수 있다.

TMS570LC4357 Hercules MCU Technical Reference Manual에서 system control register 검색

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

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

전원 넣으면 POST(power-on-reset)가 실행되면서 SYSESR레지스터의 15번 비트가 1로 셋팅된다.

POWERON_*RESET*은 0x8000의 값을 가지므로 첫번째 if문 안으로 들어가게 된다.

```
    /* power-on reset condition */
    rst_source = POWERON_RESET;

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

rst_source = *POWERON_RESET*; // 0x8000

SYS_EXCEPTION = 0x0000FFFFU; // SYSESR 레지스터의 상위 16비트는 모두 0, 하위 16비트를 모두 1로 셋팅해주고 있다.

나머지 else if 는 건너뛴다

```
else
{
    /* No reset occurred. */
    rst_source = NO_RESET;
}

return rst_source;
}
```

반환값은 reset의 종류에 따라 달라진다. 여기서는 0x8000 리턴

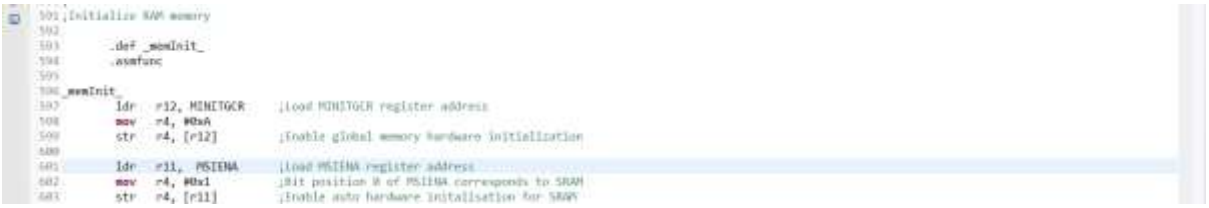
HL_sys_startup.c

switch()의 인자로 0x8000(POWERON_RESET)이 들어왔으므로

case *POWERON_RESET*: // 이부분이 실행되지만, 어떠한 명령도 적혀 있지 않다.

break 가 없으므로 나머지 case부분도 다 실행한다.

HL_sys_core.asm



MINITGCR 레지스터의 **주소**를 r12레지스터로 읽어온 후, R4레지스터에 0xA 값을 복사한다.

R12가 가리키는 곳 즉, MINITGCR 레지스터에 r4의 값을 저장한다.

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 initializaSon 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.**

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.**

r5에 MSTCGSTAT의 주소를 저장한다.

r5가 가리키는 MSTCGSTAT의 값을 r4에 저장한다

r4의 8번비트가 읽었을 때 1인지 본다(and연산)

beq mloop // and해서 0이 나오면, mloop를 돈다

8번 비트가 1일 경우 "Hardware Initialization of all memory is completed" 이므로 하드웨어 초기화 잘 될 때까지 루프를 돌면서 기다린다는 의미이다.
어셈블리어의 for문에 해당한다.

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.

r4에 5를 복사해넣는다.
r12가 가리키는 곳 즉, MINITGCR 레지스터에 5값을 저장한다. 즉, 하드웨어 초기화를 완료했으므로, global memory hardware initialization을 disable시켜준다.

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.**

HL_sys_startup.c

```
/* Enable CPU Event Export */
/* This allows the CPU to signal any single-bit or double-bit errors detected
 * by its ECC logic for accesses to program flash or data RAM.
 */
_coreEnableEventBusExport_();
```

HL_sys_core.asm

```
; Enable Event Bus Export
; SourceId : CORE_SourceId_000
; DesignId : CORE_DesignId_007
; Requirements: HL_CORE_CORE_SR8

    .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
```

PMCR레지스터를 r0에 읽어온다
R0 = R0 | 0x10 하여 4번 비트에 1을 셋팅한다.
PMCR 레지스터에 r0레지스터 값을 저장한다.

> *. 외부에서 모니터링하는 버스를 'JTag'라고 한다.
> HW가 CPU를 디버깅하는데에 필요하다.

즉, 4번비트에 1을 셋팅한다는 것은 디버깅을 허용한다는 의미이다.

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
```

Table 6-2 PMCR Register bit assignments

Bits	Name	Function
[31:24]	IMP	Implementer code 0x41 = ARM
[23:16]	IDCODE	Identification code 0x15 = Cortex-R5
[15:11]	N	Specifies the number of counters implemented 0x3 = three counters implemented
[10:6]	Reserved	RAZ on reads. *Should Be Zero or Preserved* (SBZP) on writes
[5]	DP	Disable PMCCNTR when prohibited, that is, when non-invasive debug is not enabled. 0 = Count is enabled in prohibited regions. This is the reset value. 1 = Count is disabled in prohibited regions.
[4]	X	Enable export of the events to the event bus for an external monitoring block, for example the ETM, to trace events. 0 = Export disabled. This is the reset value. 1 = Export enabled.

HL_sys_startup.c

```
/* Check if there were ESM group1 errors during power-up.
 * These could occur during efuse auto-load or during reads from Flash OTP
 * during power-up. Device operation is not reliable and not recommended
 * in this case. */
if ((esmREG->SR1[2]) != 0U)
{
    esmGroup1Notification(esmREG,esmREG->SR1[2]);
}
```

if ((esmREG->SR1[2]) != 0U) // 정상작동으로 가정한다.(1은 에러, 0은 정상작동)

#define esmREG ((esmBASE_t *)0xFFFFF500U) 구조체 포인터로 정의되어 있다.
SR1을 들어가보면 다음과 같은데, 코드에서는 2번째가 0인지를 검사하고 있으므로, 주소 0x0020에 무엇이 있는지 SR1을 검색해본다. 검색 후 Offset이 20h 인지 확인한다.

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

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

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

HL_sys_startup.c

```
/* Initialize System - Clock, Flash settings with Efuse self check */
systemInit();
```

HL_system.c

```
void systemInit(void)
{
/* USER CODE BEGIN (35) */
/* USER CODE END */

    /* Configure PLL control registers and enable PLLs.
     * The PLL takes (127 + 1024 * NR) oscillator cycles to acquire lock.
     * This initialization sequence performs all the tasks that are not
     * required to be done at full application speed while the PLL locks.
     */
    setupPLL();
```

setupPLL()

```
73 void setupPLL(void)
74 {
75
76 /* USER CODE BEGIN (1) */
77 /* USER CODE END */
78
79     /* Disable PLL1 and PLL2 */
80     systemREG1->CSDISSET = 0x00000002U | 0x00000004U;
81     /*SAFETYMCUSW 28 D MR:NA <APPROVED> "Hardware status bit read check" */
82     while((systemREG1->CSDIS & 0x42U) != 0x42U)
83     {
84     /* Wait */
85     }
86
87     /* Clear Global Status Register */
88     systemREG1->GBLSTAT = 0x301U;
```

/* Disable PLL1 and PLL2 */ 01000010 (1,6bit set)

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

7-3	SETCLKSR[7-3]OFF		Set clock source[7-3] to the disabled state.
		0	*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		Set clock source[1-0] to the disabled state.
		0	*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).**

앞에서 CSDISSET 이 잘 되었는지 확인한다. 확인하는 동안에는 while루프를 반복하며 기다린다. CSDIS가 0x42로 잘 set되면 while루프를 빠져나간다

Figure 2-17. Clock Source Disable Register (CSDIS) (offset = 30h)

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		Clock source[7-3] off.
		0	Clock source[7-3] is enabled.
		1	Clock source[7-3] is disabled.
			Note: 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		Clock source[1-0] off.
		0	Clock source[1-0] is enabled.
		1	Clock source[1-0] is disabled.
			Note: On wakeup, only clock sources 0, 4, and 5 are enabled.

/* Clear Global Status Register */ 0,8,9 bit set

uint32 GBLSTAT; /* 0x00EC */ system control registers로 검색한다.

보면 PLL Slip을 검사하고 있다.*/

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

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.

다음으로 Configure PLL control registers부분이다.

```
/**   - Setup pll control register 1:
*     - Setup reset on oscillator slip
*     - Setup bypass on pll slip
*     - Setup fll output clock divider to max before lock
*     - Setup reset on oscillator fail
*     - Setup reference clock divider
*     - Setup fll multiplier
*/
systemREG1->PLLCTL1 =  (uint32)0x00000000U
                     | (uint32)0x20000000U
                     | (uint32)((uint32)0x1FU << 24U)
                     | (uint32)0x00000000U
                     | (uint32)((uint32)(8U - 1U)<< 16U)
                     | (uint32)(0x9500U);
```

```
systemREG1->PLLCTL1 =  (uint32)0x00000000U
                     | (uint32)0x20000000U // 29
                     | (uint32)((uint32)0x1FU << 24U) // 24,25,26,27,28
                     | (uint32)0x00000000U
                     | (uint32)((uint32)(8U - 1U)<< 16U) // 16,17,18
                     | (uint32)(0x9500U); // 15,12,10,8
```

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 $f_{PLL\ CLK} = f_{pwm\text{-}osc\ clk} / R$
		0	$f_{PLL\ CLK} = f_{pwm\text{-}osc\ clk} / 1$
		1h	$f_{PLL\ CLK} = f_{pwm\text{-}osc\ clk} / 2$
		:	:
		1Fh	$f_{PLL\ CLK} = f_{pwm\text{-}osc\ clk} / 32$
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	$f_{INT\ CLK} = f_{OSCIN} / 64$

REFCLKDIV가 7이므로 7+1 즉, 8분주

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	$f_{VCO\ CLK} = f_{INT\ CLK} \times 256$

PLLMUL이 9500이므로 0x9500/256+1은 (5+16x9)+1 = 150

즉,150 체배가 된다.

```
/**   - 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;
```

systemREG1->PLLCTL2 = (uint32)((uint32)255U << 22U) // 22~28

| (uint32)((uint32)7U << 12U)

| (uint32)((uint32)(1U - 1U) << 9U)

| (uint32)61U;

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 $f_{mod} = f_{\Delta} = f_{INT\ CLK} / (2 \times NS)$
		0	$f_{mod} = f_{\Delta} = f_{INT\ CLK} / (2 \times 1)$
		1h	$f_{mod} = f_{\Delta} = f_{INT\ CLK} / (2 \times 2)$
		:	:
		1FFh	$f_{mod} = f_{\Delta} = f_{INT\ CLK} / (2 \times 512)$

SPREADINGRATE가 255이므로 (255+1)x2 = 512분주

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

7이없으므로 해당되는게 없다. 따라서 "no adder to NF"

그리고 0을 9비트 쉬프트했으므로 [11:9]의 ODPLL의 값은 0이다. 따라서 1분주 그대로 쓴다

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

61은 64-3 = 01110010 = 7dh

SPR_AMOUNT 가 61이므로 NV = (61+1)/2048

```
/**   - 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);
```

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. $f_{PXE\_ODCLK2} = f_{VBAK\_CLK2} / OD2$ **Note: PLL output clock is gated off if ODPLL2 is changed while the PLL#2 is active.**
		0	$f_{PXE\_ODCLK2} = f_{VBAK\_CLK2} / 1$
		1h	$f_{PXE\_ODCLK2} = f_{VBAK\_CLK2} / 2$
28-24	PLLDIV2		PLL2 Output Clock Divider R2 = PLLDIV2 + 1, ranges from 1 to 32. $f_{PLL2\_CLK} = f_{VBAK\_ODCLK2} / R2$
		0	$f_{PLL2\_CLK} = f_{VBAK\_ODCLK2} / 1$
		1h	$f_{PLL2\_CLK} = f_{VBAK\_ODCLK2} / 2$
		:	:
		1Fh	$f_{PLL2\_CLK} = f_{VBAK\_ODCLK2} / 32$
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	$f_{INTCLK2} = f_{OSCIN} / 64$

REFCLKDIV2가 7이므로 NR2=8 따라서 8분주

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	$f_{VCOCLK2} = f_{INTCLK2} \times 256$

PLLMUL2 가 0x9500이므로 NF2는 150이다. 따라서 150체배가 된다.

이렇게 PLL3개의 기본적인 셋팅을 완료했다.

다음은 Enable PLL(s) to start up or Lock부분이다.

```
/** - Enable PLL(s) to start up or lock */
systemREG1->CSDIS = 0x00000000U
                  | 0x00000000U
                  | 0x00000008U
                  | 0x00000000U
                  | 0x00000000U
                  | 0x00000000U
                  | 0x00000000U
                  | 0x00000004U;
}
```

10001100 : 2,3,7 bit가 1로 셋팅

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		Clock source[7-3] off.
		0	Clock source[7-3] is enabled.
		1	Clock source[7-3] is disabled.
			Note: 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		Clock source[1-0] off.
		0	Clock source[1-0] is enabled.
		1	Clock source[1-0] is disabled.
			Note: On wakeup, only clock sources 0, 4, and 5 are enabled.

따라서 Clock source[3:7]을 disable 한다.

Table 2-29. Clock Sources Table

Clock Source #	Clock Source Name
Clock Source 0	Oscillator
Clock Source1	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

systemInit()

```
/* Enable clocks to peripherals and release peripheral reset */
periphInit();
```

periphInit()

Disable Peripherals before peripheral powerup

Release peripherals from reset and enable clocks to all peripherals

Power-up all peripherals

```
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;
```

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.

Enable Peripherals

```
/** - Enable Peripherals */
systemREG1->CLKCNTL |= 0x00000100U;

/* USER CODE BEGIN (18) */
/* USER CODE END */

}
```

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.

systemInit()

```
    periphInit();

/* USER CODE BEGIN (17) */
/* USER CODE END */

    /* Configure device-level multiplexing and I/O multiplexing */
    muxInit();
```

Configure device-level multiplexing and I/O multiplexing
멀티플렉싱을 해주는 부분이다.

muxInit()
Enable Pin Muxing

```
void muxInit(void){

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

    /* Enable Pin Muxing */
    pinMuxReg->KICKER0 = 0x83E70B13;
    pinMuxReg->KICKER1 = 0x95A4F1E0;
```

KICKER로 검색한다
Kicker0,1레지스터를 특정 값으로 초기화 시켜서
PINMMRnn 레지스터들에 대한 CPU의 쓰기권한을 허용해준다.
즉, PIN MUXING을 하기 위한 조건이 설정되었다.

*. PINMMRnn: Pin Multiplexing Control Registers로 output, in, special의 3가지로 나누어진다.
*. CPU의 핀들은 여러가지 기능이 한 개의 pin에 선택적으로 제공된다. 이를 pin multiplexing(Pin Mux)이라고 한다. http://jake.dothome.co.kr/category/linux/

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.

PINMMR을 메뉴얼에서 검색해본다.

Table 6-1로 연결된다.

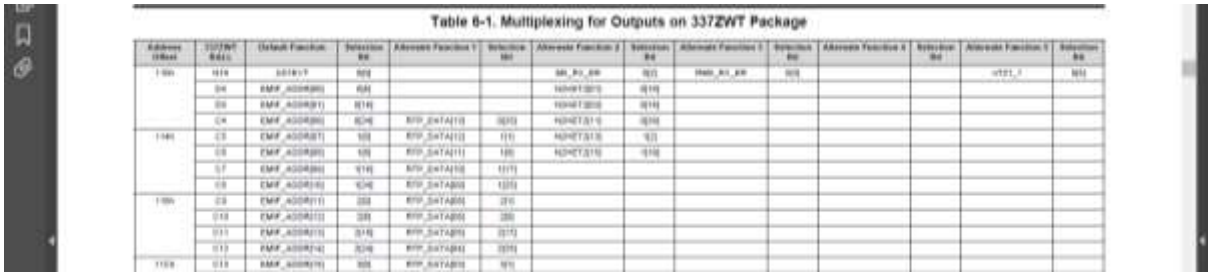


Table 6-1. Multiplexing for Outputs on 337ZWT Package

Selection bit에 따라 pin의 기능을 정의해 주고 있는 것으로 보인다.

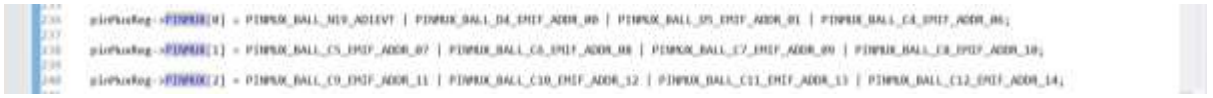
우선, PINMUX는 180개의 배열 요소를 가지고 있다.

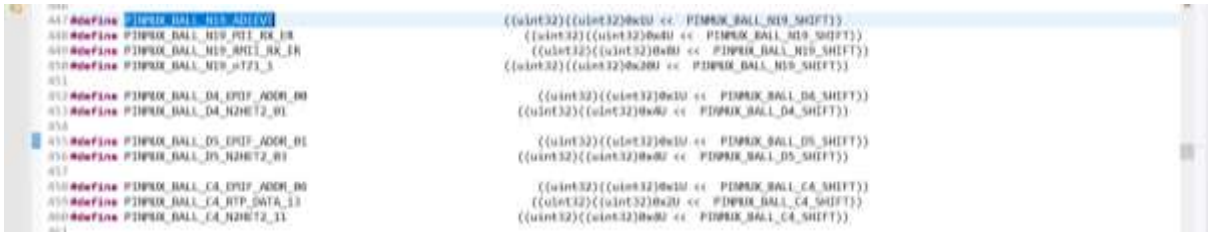
각각의 요소가 레지스터를 나타낸다.

0x110 - 1A4 까지는 Output Pin Multiplexing Control Registers (38 registers)

0x250 - 0x29C는 input Pin Multiplexing Control Registers (20..?)

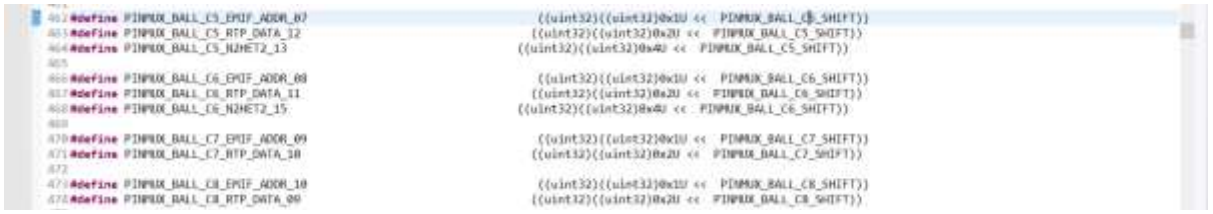
0X390 - 3DC는 Special Functionality Control Registers (20..?)





따라서 PINMUX[0]는

1<<0 | 1<<8| 1<<16| 1<<24 = 0000 0001 0000 0001 0000 0001 0000 0001 = 0x01010101



마찬가지 방법으로 PINMUX[1]은

1<<0 | 1<<8 | 1<<16 | 1<<24 = 0X01010101로 PINMUX[0]과 동일한 값이 나온다.

나머지도 구해보면 같은 값이 나온다. 이유는...?

아래부분에는 뭔가 on/off하는 매크로들이 나온다.

Pin을 multiplexing하고 사용할 수 있도록 enable/disable하는 과정인 것 같다.

PWM 7개 -> 모터 7개 제어가 가능하다

Pin Muxing이 완료되었다.

9,8,1,0 bit를 set 하여 prefetch 모드를 활성화 한다.

*. Prefetch mode :명령어를 미리 가져와서 캐시해놓는다

Table 7-13. Flash Read Control Register (FRDCNTL) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	RWAIT	0-Fh	Random/data Read Wait State The random read wait state bits indicate how many wait states are added to a Flash read access. Address wait state is fixed to 1 HCLK cycle. **Note:** The required wait states for each HCLK frequency can be found in the device-specific data sheet.
7-2	Reserved	0	Reads return 0. Writes have no effect.
1	PFUENB		Prefetch Enable for Port B
		0	Prefetch Mode is disabled.
		1	Prefetch Mode is enabled. (Recommended)
0	PFUENA		Prefetch Enable for Port A
		0	Prefetch Mode is disabled.
		1	Prefetch Mode is enabled. (Recommended)

두개의 매크로로 어떤 주소의 값을 제어하고 있다. 매뉴얼을 살펴보면 FSM을 사용하기 위한 레지스터 셋팅을 해주고 있다.

Table 7-43. FSM Register Write Enable Register (FSM_WR_ENA) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	WR_ENA		FSM Write Enable
		5h	This register must contain 5h in order to write to any other register in the range FFF8 7200h to FFF8 72FFh. This is the first register to be written when setting up the FSM.
		All other values	For all other values, the FSM registers cannot be written.

5h값을 셋팅하여 FSM write를 활성화 하고있다.

Table 7-44. EPROM Emulation Configuration Register (EEPROM_CONFIG) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	EWAIT	0-Fh	EEPROM Wait state Counter Replaces the RWAIT count in the EEPROM register. The same formulas that apply to RWAIT apply to EWAIT in the EEPROM bank.
15-0	Reserved	0	Reads return 0. Writes have no effect.

19-16 에 9(1001), 1bit를 1로 셋팅해주고 있다.

FSM_WR_ENA_HL에 2값을 셋팅해서 FSM 레지스터에 쓰기를 비활성화 시키고

flashWREG구조체의 FMPWRMODE멤버가 가리키는 곳의 값을 변경한다.

3<<14|3<<2|3<<0 이므로 15,14,3,2,1,0 bit를 1로 셋팅한다.

따라서 Bank 0,1,7의 Power Mode가 active mode로 변경된다.

Table 7-26. Flash Bank Power Mode Register (FBPWRMODE) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	505h	Do not write to these register bits.
15-14	BANKPWR7		Bank 7 Power Mode.
		0	Bank sleep mode
		1h	Bank standby mode
		2h	Reserved
		3h	Bank active mode
13-4	Reserved	3FFh	Do not write to these register bits.
3-2	BANKPWR1		Bank 1 Power Mode.
		0	Bank sleep mode
		1h	Bank standby mode
		2h	Reserved
		3h	Bank active mode
1-0	BANKPWR0		Bank 0 Power Mode.
		0	Bank sleep mode
		1h	Bank standby mode
		2h	Reserved
		3h	Bank active mode