TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/5/4
수업일수	48 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

Cortex-R5F Boot 코드 분석

asm

Cortex-R5F Boot 코드 분석

- HL_sys_intvecs.asm

인터럽트 벡터테이블 - 인터럽트 처리하기 위한 루틴들을 가지고있는 구간

```
    .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]


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

_c_int00 에서는 각 엔트리를 리셋한다.

_c_int00

```
 96 void _c_int00(void)
 97 {
 98
 99 /* USER CODE BEGIN (5) */
100 /* USER CODE END */
101
102     /* Initialize Core Registers to avoid CCM Error */
103     _coreInitRegisters_();
104
105     /* Initialize Stack Pointers */
106     _coreInitStackPointer_();
107
108     /* Reset handler: the following instructions read from the system exception status register
109      * to identify the cause of the CPU reset.
110      */
111     switch(getResetSource())
112     {
113         case POWERON_RESET:
114         case DEBUG_RESET:
115         case EXT_RESET:
116
117 /* USER CODE BEGIN (6) */
118 /* USER CODE END */
119
120         /* Initialize L2RAM to avoid ECC errors right after power on */
121         _memInit_();
122
123 /* USER CODE BEGIN (7) */
124 /* USER CODE END */
```

: 103

coreInitRegisters() CCM 에러를 피하기 위해 core 레지스터를 초기화 하는 작업을 한다.

: 106

coreInitStackPointer() : 스택 포인터를 초기화 함.

coreInitRegisters

```
113 /* System Core Interface Functions */
114
115 /** @fn void _coreInitRegisters_(void)
116 *   @brief Initialize Core register
117 */
118 void _coreInitRegisters_(void);
119
```

F3 을 누르면 _coreInitRegisters_를 찾을 수 없는데 이는 어셈블리어로 되어있다는 의미이다.

coreInitRegisters(1)

```
46
47      .def     _coreInitRegisters_
48      .asmfunc
49
50
51  _coreInitRegisters_
52
53      ; After reset, the CPU is in the Supervisor mode (M = 10011)
54          mov r0, lr
55          mov r1, #0x0000
56          mov r2, #0x0000
57          mov r3, #0x0000
58          mov r4, #0x0000
59          mov r5, #0x0000
60          mov r6, #0x0000
61          mov r7, #0x0000
62          mov r8, #0x0000
63          mov r9, #0x0000
64          mov r10, #0x0000
65          mov r11, #0x0000
66          mov r12, #0x0000
67          mov r13, #0x0000
68          mrs r1, cpsr
69          msr spsr_cxsf, r1
70          ; Switch to FIQ mode (M = 10001)
71          cps #17
72          mov lr, r0
73          mov r8, #0x0000
74          mov r9, #0x0000
75          mov r10, #0x0000
76          mov r11, #0x0000
77          mov r12, #0x0000
78          mrs r1, cpsr
79          msr spsr_cxsf, r1
80          ; Switch to IRQ mode (M = 10010)
81          cps #18
82          mov lr, r0
83          mrs r1,cpsr
84          msr spsr_cxsf, r1
85          ; Switch to Abort mode (M = 10111)
86          cps #23
87          mov lr, r0
88          mrs r1,cpsr
89          msr spsr_cxsf, r1
```

HL_sys_core.asm 에서 _coreInitRegisters_를 찾음

여기서는 각 모드별로 cpsr 과 spsr 을 분리시키는 작업을 한다.

리셋 이후(컴퓨터 전원이 켜진 이후)에 CPU 는 Supervisor mode 가 된다.

Supervisor : SWI(software interrupt)가 실행될 때 입력되는 운영체제의 보호모드.
IRQ : 일반 인터럽트

Abort : 내부 인터럽트의 종류 중 하나.(0 으로 나눌 때)

: 54 r0 에 lr 의 값을 넣는다. 돌아가야 할 복귀주소를 r0 에 저장.

: 55~67 r1~r13 의 레지스터를 0 으로 초기화한다.

: 68 cpsr 의 값을 r1 에 저장한다.

: 69 r1 의 값을 spsr_cxsf 로 저장한다.

이는 슈퍼바이저 모드 전용 cpsr 을 만드는 작업을 한다. arm 에서는 모드가 여러 개가 존재하기 때문에 context switch 가 일어날 때 레지스터에 저장되는데 만약 이를 소프트웨어적으로 처리하게 되면 많은 비용을 치뤄야 하기에 하드웨어적으로 이를 구성하였다.

f : CPSR 의 Flag 에 해당. 24~31 비트에 해당하는 것을 사용한다.(NZCV)

s : CPSR 의 Status 에 해당. 23~16 비트에 해당.

x : CPSR 의 Extension 에 해당. 15~8 비트에 해당. 확장적으로 추가된 것.

c : CPSR 의 Control 에 해당 0~7 비트에 해당.

spsr_cxsf 는 f,s,x,c 를 모두 사용한다. 모든 비트를 활성화함

:71 cps 는 현재 동작모드를 변경하는 명령어.

coreInitRegisters(2)

```
90          ; Switch to Undefined Instruction Mode (M = 11011)
91          cps #27
92          mov lr, r0
93          mrs r1,cpsr
94          msr spsr_cxsf, r1
95          ; Switch to System Mode ( Shares User Mode registers ) (M = 11111)
96          cps #31
97          mov lr, r0
98          mrs r1,cpsr
99          msr spsr_cxsf, r1
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
```

Undefined Instruction : 정의되지 않은 명령어가 왔을 때.(arm 64 비트, 하위호환)

여기까지 supervisor 모드, FIQ 모드, IRQ 모드, abort 모드, Undefined Instruction 모드, System 모드를 초기화 하였다.

: 98~99

mrs - 레지스터와 스페셜 레지스터의 제어

mrc, mcr - 레지스터와 코프로세서의 동작

MRC p15, 0, , c1, c0, 2 ; Read CPACR(Coprocessor Access Control Register, vfp 플로팅 포인트 유닛활성화를 어셈블리어로 만든 것.)

cpacr 값을 r2 에 넣음.

r2 와 0xF00000 을 or 연산함. 20~23 비트를 1 로 세팅함. cp10, cp11 을 1 로 세팅하는 것임. 부동소수점을 사용하기 위한 작업.

MCR p15, 0, , c1, c0, 2 ; Write CPACR

mcr - 레지스터에 있는 값을 코프로세서를 통해 적음. 부동소수점 처리를 할수 있게 된다.

: 105 r2 에 0x40000000 를 넣음. 30 번째 비트 활성화

[30] EN VFP enable bit. Setting EN enables VFP functionality. Reset clears EN

:106 fmxr : Transfer contents between an ARM register and a VFP system register

r2 의 레지스터 정보를 fpexc 에 전송하여 30 번 비트를 킴

: 108~123 fmdrr : Transfer contents between two ARM registers and a double-precision floating-point register.

double-precision floating-point register(32 비트인데도 double 을 사용할 수 있도록 하는 레지스터)

fmdrr - double 을 처리하는 레지스터를 0 으로 초기화시킴

:124~132 부동소수점 유닛은 1 클록에 끝나지 않아 끝날 때 까지 지연시킴. fmdrr 이 끝날 때 까지 지연한다. for 문은 파이프라인이 깨지기 때문에 이 방법을 사용함.

:132 bx - r0 에 저장되어 있던 복귀주소로 점프함

coreInitStackPointer

```
137 ;-------------------------------------------------------------------------------
138 ; Initialize Stack Pointers
139 ; SourceId : CORE_SourceId_002
140 ; DesignId : CORE_DesignId_002
141 ; Requirements: HL_CONQ_CORE_SR3
142
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
```

스택을 할당해 주어야한다.

동작모드마다 별도의 스택을 부여함.

:148 슈퍼바이저 모드로 바꿔줌

: 149~159

각 모드마다 sp 에 값들을 넣어줌.

:162~167

어셈블리어로 #define 을 하는 것.

.word 하면 오른쪽에 계산된 값들로 치환됨.

사용자에는 4096, 나머지 공간에는 256 의 스택공간을 준다.

```
443
444 /* SourceId : SYSTEM_SourceId_008 */
445 /* DesignId : SYSTEM_DesignId_008 */
446 /* Requirements : HL_CONQ_SYSTEM_SR9 */
447 resetSource_t getResetSource(void)
448 {
449     register resetSource_t rst_source;
450
451     if ((SYS_EXCEPTION & (uint32)POWERON_RESET) != 0U)
452     {
453         /* power-on reset condition */
454         rst_source = POWERON_RESET;
455
456         /* Clear all exception status Flag and proceed since it's power up */
457         SYS_EXCEPTION = 0x0000FFFFU;
458     }
459     else if ((SYS_EXCEPTION & (uint32)EXT_RESET) != 0U)
460     {
461         /* Reset caused due to External reset. */
462         rst_source = EXT_RESET;
```

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

```
106 typedef enum
107 {
108     POWERON_RESET       = 0x8000U,  /**< Alias for Power On Reset     */
```

SYS_EXCEPTION = 0Xffffffe4u

POWERON_RESET = 0x8000u

SYS_EXCEPTION & POWERON_RESET 은 15 번째 비트가 1 이 되어 if 문 만족한다.

:454 rst_source 에 0x8000u 를 넣고

:457 SYS_EXCEPTION 에 0x0000ffffu 를 넣음.

:456 모든 excption 상태플래그 clear 하고 부팅중에 있으면 마저 일처리해라.

그리고 rst_source 를 리턴함.

memInit

```
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
```

메모리를 초기화시킴

: 597 MINITGCR 레지스터의 주소를 load 하여 r12 에 저장

:598 r4 에 10 을 저장

:599 MINITGCR 에 10 을 전달함.

: 601 MSINENA 인데 오타가 났음. MSINENA 레지스터 주소 값을 r11 에 넣음.

:602 r4 에 1 을 넣음

:603 1 을 MSINENA 에 전달함.

:607~610 for 문의 어셈블리어임

r4 랑 0x100(8 번비트) and 함 0 이면 eq 가 만족되고 beq - loop 를 실행하는데 1 이면 만족하지 않아 다음 연산인 mov 를 실행함.

: 610 r4 에 5 를 넣음

: 611 MINITGCR 에 5 를 넣음 초기화하지 말라고 막아 놓는 것.

: 612 호출하는 곳으로 돌아 감 (memInit)

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.

A 가 들어오면 global memory hardware initialization 가 활성화 된다.

다른 값이 면 활성화 하지 않음.

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.**

1 이 들어오게 되면

PBIST controller 를 활성화시킴.

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.

_c_int00 (2)

```
132
133         /* Enable CPU Event Export */
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
```

coreEnavleEventBusExport

```
228 ;---------------------------------------------------------------------------
229 ; Enable Event Bus Export
230 ; SourceId : CORE_SourceId_006
231 ; DesignId : CORE_DesignId_007
232 ; Requirements: HL_CONQ_CORE_SR6
233
234     .def     _coreEnableEventBusExport_
235     .asmfunc
236
237 _coreEnableEventBusExport_
238
239         mrc   p15, #0x00, r0,         c9, c12, #0x00
240         orr   r0,  r0,    #0x10
241         mcr   p15, #0x00, r0,         c9, c12, #0x00
242         bx    lr
243
244     .endasmfunc
```

: 239 PMCR 레지스터 값을 r0 로 가져옴

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

PMCR(Performance monitoring registers)

: 240 4 번비트를 1 로 세팅

디버깅을 허용하겠다. 외부에서 접근하는 버스를 허용해주겠다 라는 어셈블리어

: 241 코프로세서에 세팅함

:242 복귀주소로 돌아감

[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.

_c_int00 (3)

```
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();
```

: 143 eFuse - 과전류 들어왔을 때 알아서 끊어지는 것. Flash 쪽에서 문제가 발생하면 자동으로 문제를 차단해주는 기능이 들어가 있음. 과전류가 들어올 때 보호해주는 기능이 있다.

SR1[2] == 0x001C 이므로 if 문 만족

```
108 #define esmREG ((esmBASE_t *)0xFFFFF500U)
```

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

Table 16-10. ESM Status Register 2 (ESMSR2) Field Descriptions

Bit	Field	Value	Description
31-0	ESF2		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; no interrupt is pending. Write: Leaves the bit unchanged.
		1	Read: Error occurred; interrupt is pending. Write: Clears the bit. ESMSSR2 is not impacted by this action. **Note:** In normal operation the flag gets cleared when reading the appropriate vector in the ESMIOFFHR offset register. Reading ESMIOFFHR will not clear the ESMSR1 and the shadow register ESMSSR2.

에러가 발생하여 인터럽트가 지연된다.

```
78 #pragma WEAK(esmGroup3Notification)
79 void esmGroup3Notification(esmBASE_t *esm, uint32 channel)
80 {
81 /*  enter user code between the USER CODE BEGIN and USER CODE END. */
82 /* USER CODE BEGIN (5) */
83 /* USER CODE END */
84     for(;;)
85     {
86     }/* Wait */
87 /* USER CODE BEGIN (6) */
88 /* USER CODE END */
89 }
```

esmGroup3Notification 은 아무것도 하는 일이 없음.

systemInit(1)

```
339 void systemInit(void)
340 {
341 /* USER CODE BEGIN (15) */
342 /* USER CODE END */
343
344     /* Configure PLL control registers and enable PLLs.
345      * The PLL takes (127 + 1024 * NR) oscillator cycles to acquire lock.
346      * This initialization sequence performs all the tasks that are not
347      * required to be done at full application speed while the PLL locks.
348      */
349     setupPLL();
350
351
352 /* USER CODE BEGIN (16) */
353 /* USER CODE END */
354
355     /* Enable clocks to peripherals and release peripheral reset */
356     periphInit();
357
358 /* USER CODE BEGIN (17) */
359 /* USER CODE END */
360
361     /* Configure device-level multiplexing and I/O multiplexing */
362     muxInit();
363
364 /* USER CODE BEGIN (18) */
365 /* USER CODE END */
366
367     /** - Set up flash address and data wait states based on the target CPU clock frequency
368      * The number of address and data wait states for the target CPU clock frequency are specified
369      * in the specific part's datasheet.
370      */
371     setupFlash();
372
373 /* USER CODE BEGIN (19) */
374 /* USER CODE END */
375
376     /** - Configure the LPO such that HF LPO is as close to 10MHz as possible */
377     trimLPO();
```

클록설정

setupPLL(1)

```
72 /* Requirements : HL_CONQ_SYSTEM_SR5 */
73 void setupPLL(void)
74 {
75
76 /* USER CODE BEGIN (3) */
77 /* USER CODE END */
78
79     /* Disable PLL1 and PLL2 */
80     systemREG1->CSDISSET = 0x00000002U | 0x00000040U;
81     /*SAFETYMCUSW 28 D MR:NA <APPROVED> "Hardware status bit read check" */
82     while((systemREG1->CSDIS & 0x42U) != 0x42U)
83     {
84     /* Wait */
85     }
```

:80 6 번비트와 1 번비트를 세팅함

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).**

setupPLL(2)

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

			channel on which it is routed.
12-0	Reserved	0	Reads return 0. Writes have no effect.

읽을 땐 0 을 리턴하고 쓸때는 아무런 효과가 없음.

setupPLL(3)

```
93    /**   - Setup pll control register 1:
94    *     - Setup reset on oscillator slip
95    *     - Setup bypass on pll slip
96    *     - setup Pll output clock divider to max before Lock
97    *     - Setup reset on oscillator fail
98    *     - Setup reference clock divider
99    *     - Setup Pll multiplier
100   */
101   systemREG1->PLLCTL1 =  (uint32)0x00000000U
102                        | (uint32)0x20000000U
103                        | (uint32)((uint32)0x1FU << 24U)
104                        | (uint32)0x00000000U
105                        | (uint32)((uint32)(8U - 1U)<< 16U)
106                        | (uint32)(0x9500U);
```

: 101~106 28~24, 16~18, 8,10,12,15 번째 비트를 1 로 세팅.

f_PLLCLK = f_POST-ODCLK / 32

f_INTCLK = f_OSCIN / 8

f_VCOCLK = f_INTCLK x 150

setupPLL(4)

```
107
108   /**   - Setup pll control register 2
109   *     - Setup spreading rate
110   *     - Setup bandwidth adjustment
111   *     - Setup internal Pll output divider
112   *     - Setup spreading amount
113   */
114   systemREG1->PLLCTL2 =  (uint32)((uint32)255U << 22U)
115                        | (uint32)((uint32)7U << 12U)
116                        | (uint32)((uint32)(1U - 1U) << 9U)
117                        | (uint32)61U;
```

22~29, 12~14, 5, 4, 3, 2, 0 비트 1 로 세팅

10 번째 비트 0 으로 세팅

f_mod = f_s = f_INTCLK / 512

meanless

NV = 62 / 2048

f_POST-ODCLK = f_VCOCLK / 1

setupPLL(5)

```
121     /**   - Setup pll2 control register :
122     *     - setup Pll output clock divider to max before Lock
123     *     - Setup reference clock divider
124     *     - Setup internal Pll output divider
125     *     - Setup Pll multiplier
126     */
127     systemREG2->PLLCTL3 = (uint32)((uint32)(1U - 1U) << 29U)
128                         | (uint32)((uint32)0x1FU << 24U)
129                         | (uint32)((uint32)(8U - 1U)<< 16U)
130                         | (uint32)(0x9500U);
```

29 번 0 으로 픽스

28~24, 18~16, 15, 12, 10, 8 비트 1 로 세팅

f_POST-ODCLK2 = f_OUTPUT-CLK2 / 1

f_PLL2CLK = f_POST-ODCLK2 / 32

f_INTCLK2 = f_OSCIN / 8

f_VCOCLK2 = f_INTCLK2 x 150

setupPLL(6)

```
132     /** - Enable PLL(s) to start up or Lock */
133     systemREG1->CSDIS = 0x00000000U
134                       | 0x00000000U
135                       | 0x00000008U
136                       | 0x00000080U
137                       | 0x00000000U
138                       | 0x00000000U
139                       | 0x00000000U
140                       | 0x00000004U;
```

2,3 번째 비트 1 로 세팅