TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

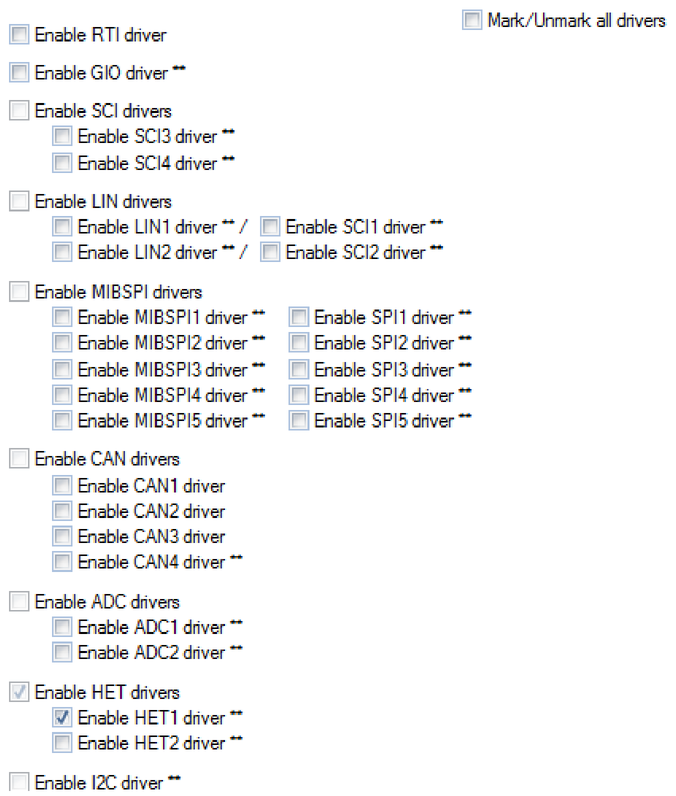
강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – GJ (박현우)
uc820@naver.com

목차

2. Cortex-R5F Hercules Safety MCU – (het_PWM 설정)

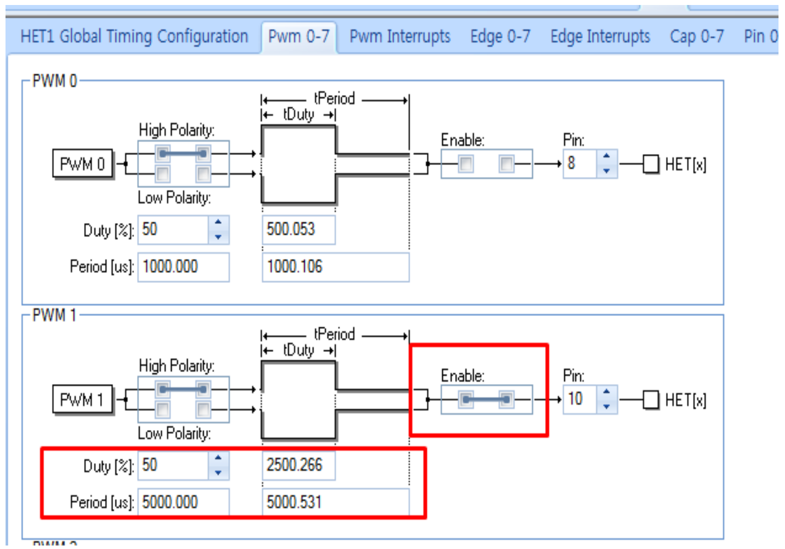
<1>



Mark/Unmark all drivers

- Enable RTI driver
- Enable GIO driver **
- Enable SCI drivers
  - Enable SCI3 driver **
  - Enable SCI4 driver **
- Enable LIN drivers
  - Enable LIN1 driver ** / Enable SCI1 driver **
  - Enable LIN2 driver ** / Enable SCI2 driver **
- Enable MIBSPI drivers
  - Enable MIBSPI1 driver ** — Enable SPI1 driver **
  - Enable MIBSPI2 driver ** — Enable SPI2 driver **
  - Enable MIBSPI3 driver ** — Enable SPI3 driver **
  - Enable MIBSPI4 driver ** — Enable SPI4 driver **
  - Enable MIBSPI5 driver ** — Enable SPI5 driver **
- Enable CAN drivers
  - Enable CAN1 driver
  - Enable CAN2 driver
  - Enable CAN3 driver
  - Enable CAN4 driver **
- Enable ADC drivers
  - Enable ADC1 driver **
  - Enable ADC2 driver **
- [✓] Enable HET drivers
  - [✓] Enable HET1 driver **
  - Enable HET2 driver **
- Enable I2C driver **

<2>

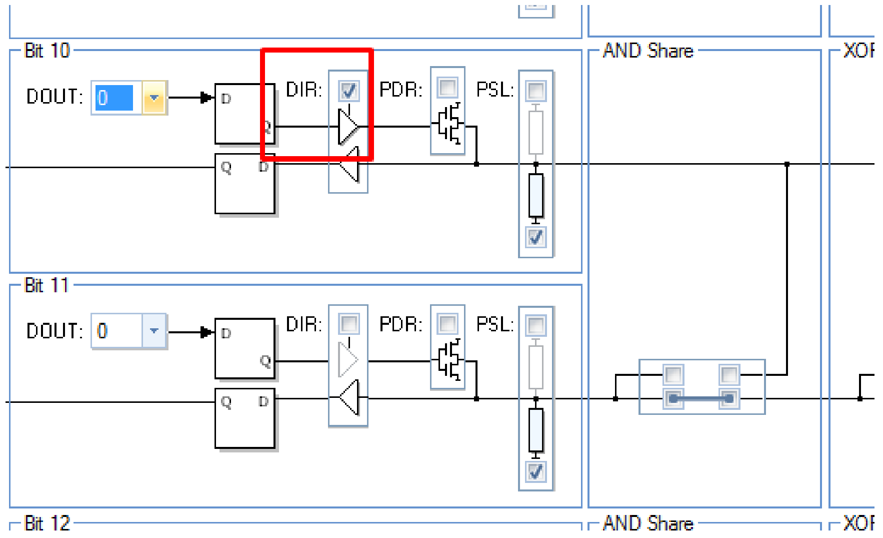


HET1 Global Timing Configuration | Pwm 0-7 | Pwm Interrupts | Edge 0-7 | Edge Interrupts | Cap 0-7 | Pin 0

PWM 0
- High Polarity / Low Polarity
- tPeriod / tDuty
- Duty [%]: 50 — 500.053
- Period [us]: 1000.000 — 1000.106
- Enable:
- Pin: 8 — HET[x]

PWM 1
- High Polarity / Low Polarity
- tPeriod / tDuty
- Duty [%]: 50 — 2500.266
- Period [us]: 5000.000 — 5000.531
- Enable: [✓]
- Pin: 10 — HET[x]

<3>

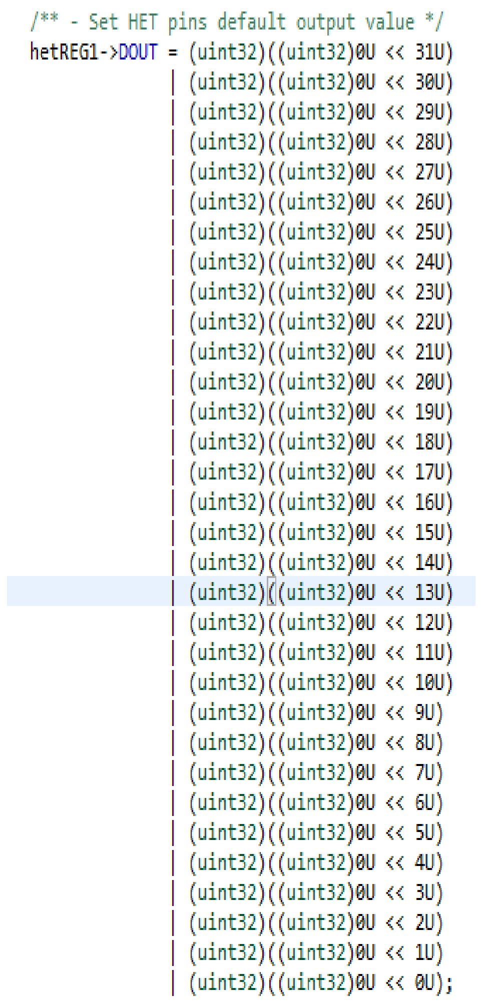


Bit 10
- DOUT: 0
- DIR: [✓]
- PDR:
- PSL:

Bit 11
- DOUT: 0
- DIR:
- PDR:
- PSL:

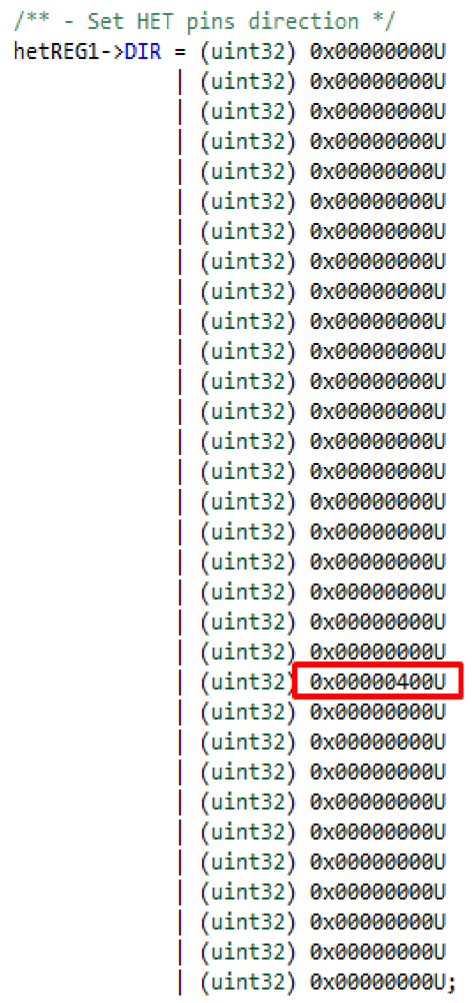
Bit 12

2. Cortex-R5F Hercules Safety MCU – (hetInit 분석 1)



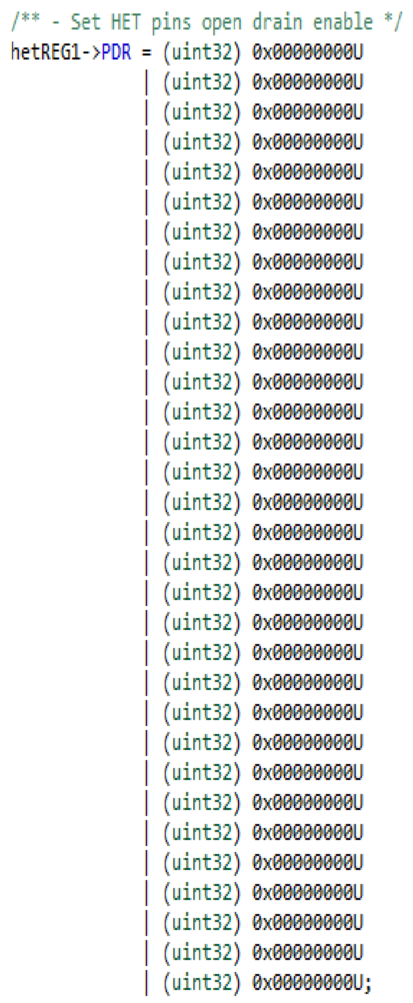
```c
/** - Set HET pins default output value */
hetREG1->DOUT = (uint32)((uint32)0U << 31U)
              | (uint32)((uint32)0U << 30U)
              | (uint32)((uint32)0U << 29U)
              | (uint32)((uint32)0U << 28U)
              | (uint32)((uint32)0U << 27U)
              | (uint32)((uint32)0U << 26U)
              | (uint32)((uint32)0U << 25U)
              | (uint32)((uint32)0U << 24U)
              | (uint32)((uint32)0U << 23U)
              | (uint32)((uint32)0U << 22U)
              | (uint32)((uint32)0U << 21U)
              | (uint32)((uint32)0U << 20U)
              | (uint32)((uint32)0U << 19U)
              | (uint32)((uint32)0U << 18U)
              | (uint32)((uint32)0U << 17U)
              | (uint32)((uint32)0U << 16U)
              | (uint32)((uint32)0U << 15U)
              | (uint32)((uint32)0U << 14U)
              | (uint32)((uint32)0U << 13U)
              | (uint32)((uint32)0U << 12U)
              | (uint32)((uint32)0U << 11U)
              | (uint32)((uint32)0U << 10U)
              | (uint32)((uint32)0U << 9U)
              | (uint32)((uint32)0U << 8U)
              | (uint32)((uint32)0U << 7U)
              | (uint32)((uint32)0U << 6U)
              | (uint32)((uint32)0U << 5U)
              | (uint32)((uint32)0U << 4U)
              | (uint32)((uint32)0U << 3U)
              | (uint32)((uint32)0U << 2U)
              | (uint32)((uint32)0U << 1U)
              | (uint32)((uint32)0U << 0U);
```

DOUT – 모두 0



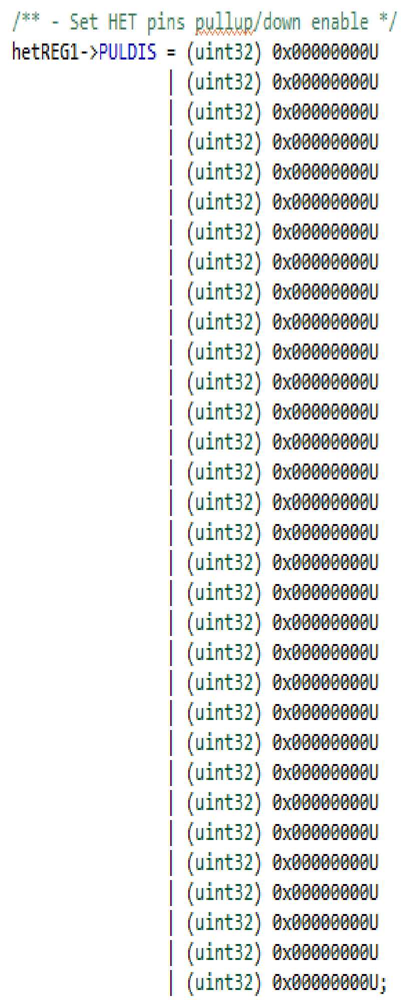
```c
/** - Set HET pins direction */
hetREG1->DIR = (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000400U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U;
```

DIR – 10bit 활성화



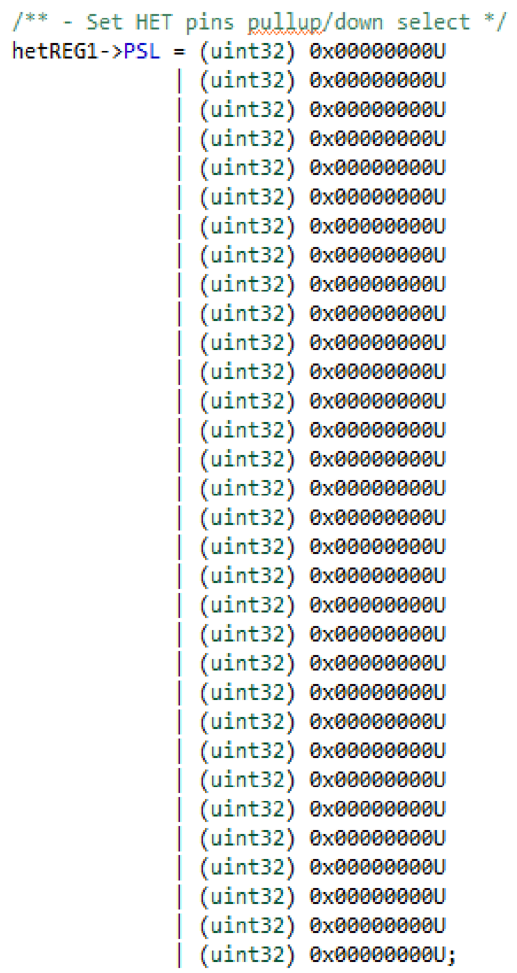
```c
/** - Set HET pins open drain enable */
hetREG1->PDR = (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U;
```

PDR – pin을 push/pull mode로 구성



```c
/** - Set HET pins pullup/down enable */
hetREG1->PULDIS = (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U
                | (uint32) 0x00000000U;
```

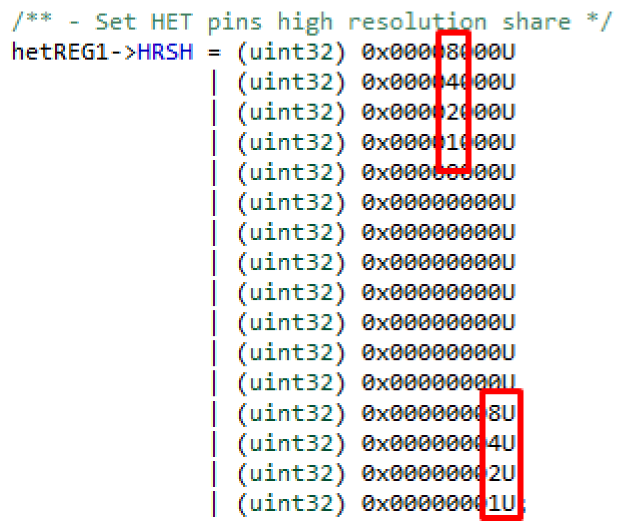
PULDIS – pull 활성화



```c
/** - Set HET pins pullup/down select */
hetREG1->PSL = (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U;
```

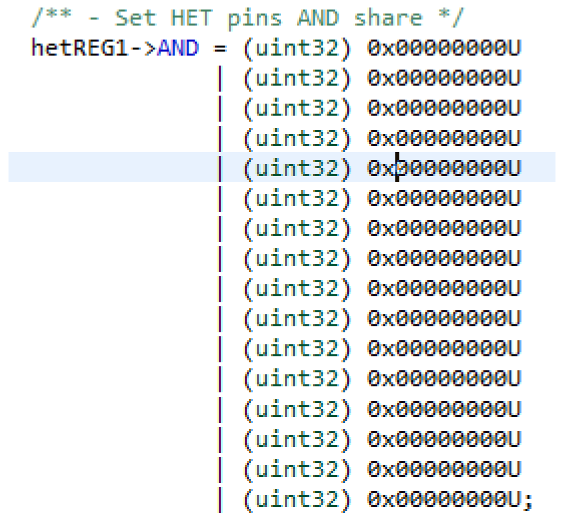
Pulldown
– 모든 bit 활성화

2. Cortex-R5F Hercules Safety MCU – (hetInit 분석 2)



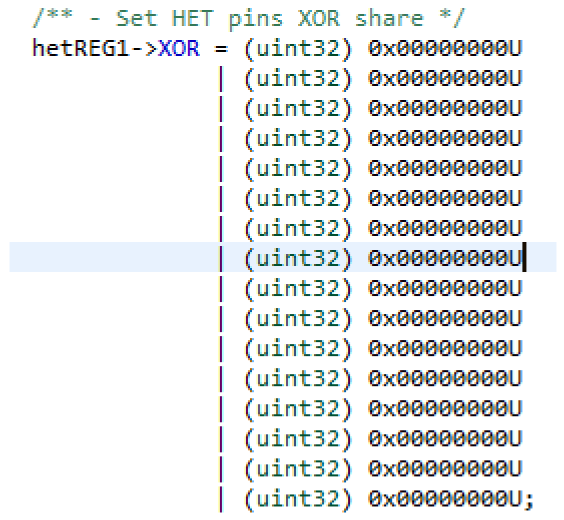
```
/** - Set HET pins high resolution share */
hetREG1->HRSH = (uint32) 0x00008000U
              | (uint32) 0x00004000U
              | (uint32) 0x00002000U
              | (uint32) 0x00001000U
              | (uint32) 0x00000000U
              | (uint32) 0x00000000U
              | (uint32) 0x00000000U
              | (uint32) 0x00000000U
              | (uint32) 0x00000000U
              | (uint32) 0x00000000U
              | (uint32) 0x00000000U
              | (uint32) 0x00000000U
              | (uint32) 0x00000008U
              | (uint32) 0x00000004U
              | (uint32) 0x00000002U
              | (uint32) 0x00000001U;
```

12 ~ 15 bit
: share 31/30 – share 25/24
0~3 bit
: share 7/6 – share 1/0



```
/** - Set HET pins AND share */
hetREG1->AND = (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U;
```

모든 share는 AND를 공유하지 않음.



```
/** - Set HET pins XOR share */
hetREG1->XOR = (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U;
```

모든 share는 XOR를 공유하지 않음.

2. Cortex-R5F Hercules Safety MCU – (hetInit 분석 3)

```
/** - Setup prescaler values
*     - Loop resolution prescaler
*     - High resolution prescaler
*/
hetREG1->PFR = (uint32)((uint32) 6U << 8U)
             | ((uint32) 0U);
```

9~10 bit set 1 : 6h , / 64

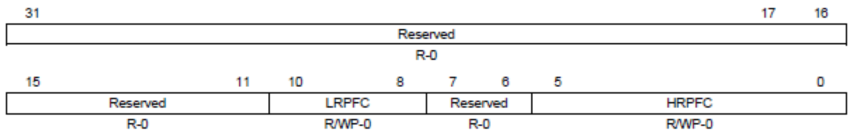
0~5 bit set 0 : 0, /1

0000 0000 0000 0000 0000 0110 0000 0000

23.4.2 *Prescale Factor Register (HETPFR)*

N2HET1: offset = FFF7 B804h; **N2HET2:** offset = FFF7 B904h

Figure 23-57. Prescale Factor Register (HETPFR)

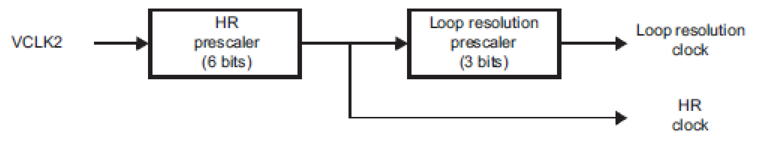


LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 23-17. Prescale Factor Register (HETPFR) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reads return 0. Writes have no effect.
10-8	LRPFC		Loop-Resolution Pre-scale Factor Code. LRPFC determines the loop-resolution prescale divide rate (lr).
		0	/1
		1h	/2
		2h	/4
		3h	/8
		4h	/16
		5h	/32
		6h	/64
		7h	/128
7-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	HRPFC		High-Resolution Pre-scale Factor Code. HRPFC determines the high-resolution prescale divide rate (hr).
		0	/1
		1h	/2
		2h	/3
		3h	/4
		:	:
		3Dh	/62
		3Eh	/63
		3Fh	/64

Figure 23-7. Prescaler Configuration



The following abbreviations and relations are used in this document:

1. hr: high resolution prescale factor (1, 2, 3, 4,..., 63, 64)
2. lr: loop resolution prescale factor (1, 2, 4, 8, 16, 32, 64,128)
3. ts: Time slots (cycles) available for instruction execution per loop. ts = hr x lr
4. HRP = high resolution clock period HRP = hr × T_{VCLK2} (ns)
5. LRP = loop resolution clock period LRP = lr × HRP (ns)

The loop resolution period (LRP) must be selected to be larger than the number of Time slots (VCLK2 cycles) required to complete the worst-case execution path through the N2HET program. Otherwise a program overflow condition may occur (see Section 23.2.1.4). Because of the relationship of time slots to the hr and lr prescalers as described in item 3 above, increasing either hr or lr increases the number of time slots available for program execution. However, lr would typically be increased first, since increasing hr results in a decrease in timer resolution since it reduces the clock to the High Resolution IO structures.

The divide rates hr and lr can be defined in the HETPFR register. Table 23-5 lists the bit field encodings for the prescale options.

Table 23-5. Prescale Factor Register Encoding

LRPFC - Loop Resolution		HRPFC - High Resolution	
HETPFR[10:8]	Prescale Factor lr	HETPFR[5:0]	Prescale Factor hr
000	/1	000000	/1
001	/2	000001	/2
010	/4	000010	/3
011	/8	000011	/4
100	/16	:	:
101	/32	111101	/62
110	/64	111110	/63
111	/128	111111	/64

N2HET Functional Description www.ti.com

23.2.3.1 Determining Loop Resolution

As an example, consider an application that requires high resolution of HRP = 62.5 ns, and loop resolution of LRP = 8 µs, and needs at least 250 time slots for the N2HET application program.

Assuming VCLK2 = 32 MHz the following shows which divide-by rates and which value in the Prescale Factor Register (HETPFR) is required for the above requirements:

$$\boxed{hr = 2} \rightarrow HRP = \frac{hr}{VCLK2} = \frac{2}{32MHz} = 62.5ns$$

$$\boxed{lr = 128} \rightarrow lr \times HRP = 128 \times 62.5ns = 8\ \mu s$$

$$ts = hr \times lr = 2 \times 128 = 256$$

$$hr = 2, lr = 128 \rightarrow HETPFR[31:0] = 0x00000701 \quad (29)$$

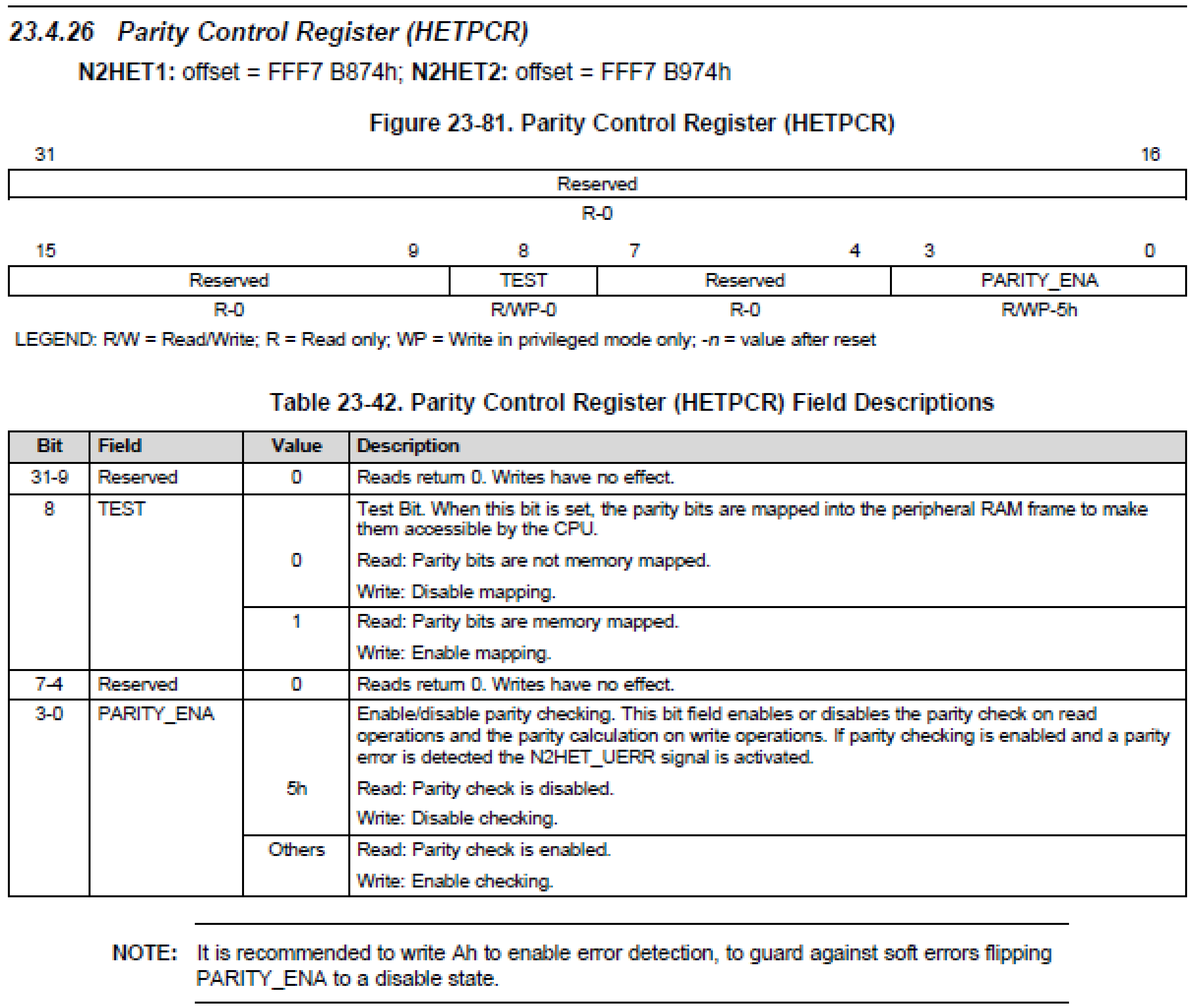
In the example above, if the loop resolution period needs to decrease from 8 µs to 4 µs, then only 128 time slots will be available for program execution. The program may need to be restructured as suggested in Section 23.2.1.6.

2. Cortex-R5F Hercules Safety MCU – (hetInit 분석 4)

```
/** - Parity control register
 *     - Enable/Disable Parity check
 */
hetREG1->PCR = (uint32) 0x00000005U;
```

0, 2번 bit set 1 : 5h, disable checking

8번 bit set 0 : disable mapping
0000 0000 0000 0000 0000 0000 0000 0101



23.4.26 Parity Control Register (HETPCR)

N2HET1: offset = FFF7 B874h; **N2HET2:** offset = FFF7 B974h

Figure 23-81. Parity Control Register (HETPCR)

31–16
Reserved
R-0

15–9	8	7–4	3–0
Reserved	TEST	Reserved	PARITY_ENA
R-0	R/WP-0	R-0	R/WP-5h

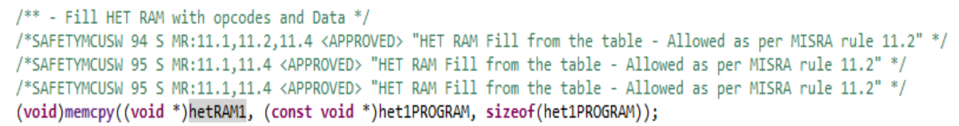
LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 23-42. Parity Control Register (HETPCR) Field Descriptions

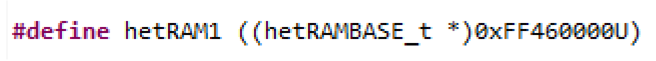
Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	TEST		Test Bit. When this bit is set, the parity bits are mapped into the peripheral RAM frame to make them accessible by the CPU.
		0	Read: Parity bits are not memory mapped. Write: Disable mapping.
		1	Read: Parity bits are memory mapped. Write: Enable mapping.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	PARITY_ENA		Enable/disable parity checking. This bit field enables or disables the parity check on read operations and the parity calculation on write operations. If parity checking is enabled and a parity error is detected the N2HET_UERR signal is activated.
		5h	Read: Parity check is disabled. Write: Disable checking.
		Others	Read: Parity check is enabled. Write: Enable checking.

NOTE: It is recommended to write Ah to enable error detection, to guard against soft errors flipping PARITY_ENA to a disable state.

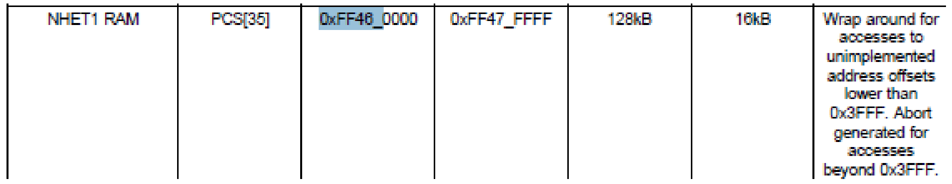
2. Cortex-R5F Hercules Safety MCU – (hetInit 분석 5)



```
/** - Fill HET RAM with opcodes and Data */
/*SAFETYMCUSW 94 S MR:11.1,11.2,11.4 <APPROVED> "HET RAM Fill from the table - Allowed as per MISRA rule 11.2" */
/*SAFETYMCUSW 95 S MR:11.1,11.4 <APPROVED> "HET RAM Fill from the table - Allowed as per MISRA rule 11.2" */
/*SAFETYMCUSW 95 S MR:11.1,11.4 <APPROVED> "HET RAM Fill from the table - Allowed as per MISRA rule 11.2" */
(void)memcpy((void *)hetRAM1, (const void *)het1PROGRAM, sizeof(het1PROGRAM));
```

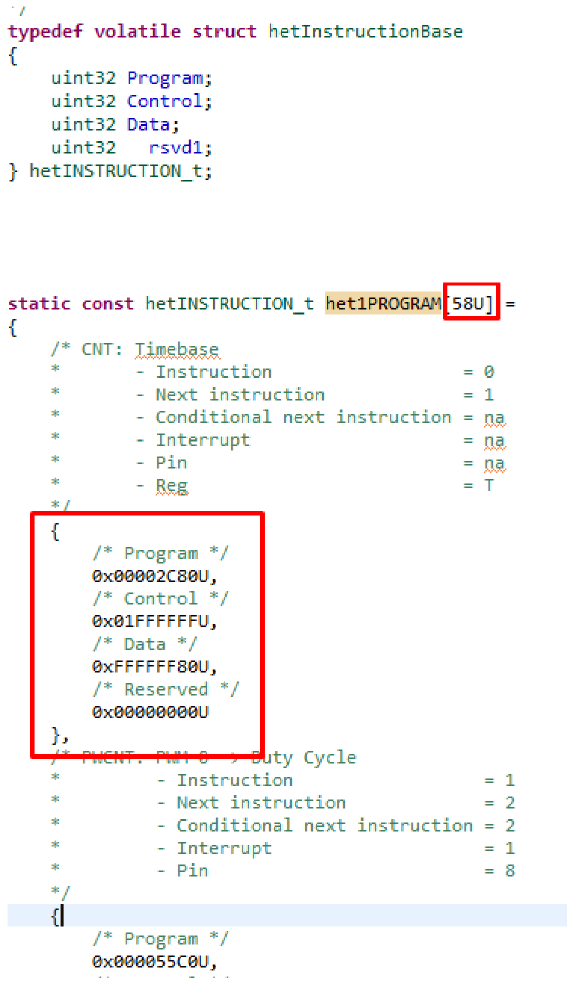


```
#define hetRAM1 ((hetRAMBASE_t *)0xFF460000U)
```

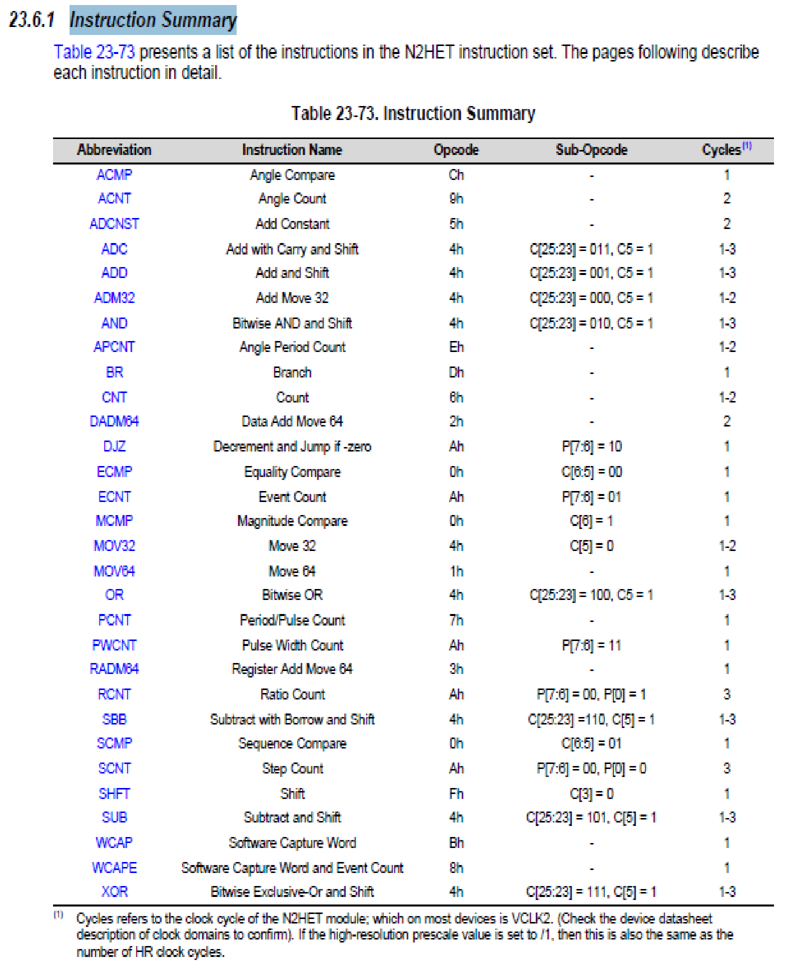


NHET1 RAM	PCS[35]	0xFF46_0000	0xFF47_FFFF	128kB	16kB	Wrap around for accesses to unimplemented address offsets lower than 0x3FFF. Abort generated for accesses beyond 0x3FFF.

hetRAM1에 het1PROGRAM의 내용을 복사



```
*/
typedef volatile struct hetInstructionBase
{
    uint32 Program;
    uint32 Control;
    uint32 Data;
    uint32   rsvd1;
} hetINSTRUCTION_t;


static const hetINSTRUCTION_t het1PROGRAM[58U] =
{
    /* CNT: Timebase
     *       - Instruction                  = 0
     *       - Next instruction             = 1
     *       - Conditional next instruction = na
     *       - Interrupt                    = na
     *       - Pin                          = na
     *       - Reg                          = T
     */
    {
        /* Program */
        0x00002C80U,
        /* Control */
        0x01FFFFFFU,
        /* Data */
        0xFFFFFF80U,
        /* Reserved */
        0x00000000U
    },
    /* PWCNT: PWM 0 -> Duty Cycle
     *         - Instruction                  = 1
     *         - Next instruction             = 2
     *         - Conditional next instruction = 2
     *         - Interrupt                    = 1
     *         - Pin                          = 8
     */
    {
        /* Program */
        0x000055C0U,
```



23.6.1 Instruction Summary

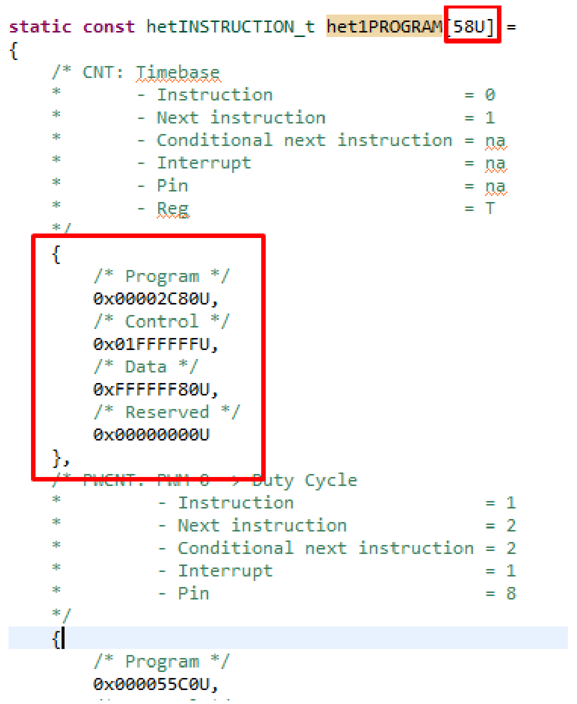
Table 23-73 presents a list of the instructions in the N2HET instruction set. The pages following describe each instruction in detail.

Table 23-73. Instruction Summary

Abbreviation	Instruction Name	Opcode	Sub-Opcode	Cycles[1]
ACMP	Angle Compare	Ch	-	1
ACNT	Angle Count	9h	-	2
ADCNST	Add Constant	5h	-	2
ADC	Add with Carry and Shift	4h	C[25:23] = 011, C5 = 1	1-3
ADD	Add and Shift	4h	C[25:23] = 001, C5 = 1	1-3
ADM32	Add Move 32	4h	C[25:23] = 000, C5 = 1	1-2
AND	Bitwise AND and Shift	4h	C[25:23] = 010, C5 = 1	1-3
APCNT	Angle Period Count	Eh	-	1-2
BR	Branch	Dh	-	1
CNT	Count	6h	-	1-2
DADM64	Data Add Move 64	2h	-	2
DJZ	Decrement and Jump if -zero	Ah	P[7:6] = 10	1
ECMP	Equality Compare	0h	C[6:5] = 00	1
ECNT	Event Count	Ah	P[7:6] = 01	1
MCMP	Magnitude Compare	0h	C[6] = 1	1
MOV32	Move 32	4h	C[5] = 0	1-2
MOV64	Move 64	1h	-	1
OR	Bitwise OR	4h	C[25:23] = 100, C5 = 1	1-3
PCNT	Period/Pulse Count	7h	-	1
PWCNT	Pulse Width Count	Ah	P[7:6] = 11	1
RADM64	Register Add Move 64	3h	-	1
RCNT	Ratio Count	Ah	P[7:6] = 00, P[0] = 1	3
SBB	Subtract with Borrow and Shift	4h	C[25:23] =110, C[5] = 1	1-3
SCMP	Sequence Compare	0h	C[6:5] = 01	1
SCNT	Step Count	Ah	P[7:6] = 00, P[0] = 0	3
SHFT	Shift	Fh	C[3] = 0	1
SUB	Subtract and Shift	4h	C[25:23] = 101, C[5] = 1	1-3
WCAP	Software Capture Word	Bh	-	1
WCAPE	Software Capture Word and Event Count	8h	-	1
XOR	Bitwise Exclusive-Or and Shift	4h	C[25:23] = 111, C[5] = 1	1-3

[1] Cycles refers to the clock cycle of the N2HET module; which on most devices is VCLK2. (Check the device datasheet description of clock domains to confirm). If the high-resolution prescale value is set to /1, then this is also the same as the number of HR clock cycles.

2. Cortex-R5F Hercules Safety MCU – (hetInit 분석 5)



```
static const hetINSTRUCTION_t het1PROGRAM[58U] =
{
    /* CNT: Timebase
    *       - Instruction                  = 0
    *       - Next instruction             = 1
    *       - Conditional next instruction = na
    *       - Interrupt                    = na
    *       - Pin                          = na
    *       - Reg                          = T
    */
    {
        /* Program */
        0x00002C80U,
        /* Control */
        0x01FFFFFFU,
        /* Data */
        0xFFFFFF80U,
        /* Reserved */
        0x00000000U
    },
    /* PWCNT: PWM 0 -> Duty Cycle
    *         - Instruction                  = 1
    *         - Next instruction             = 2
    *         - Conditional next instruction = 2
    *         - Interrupt                    = 1
    *         - Pin                          = 8
    */
    {
        /* Program */
        0x000055C0U,
```

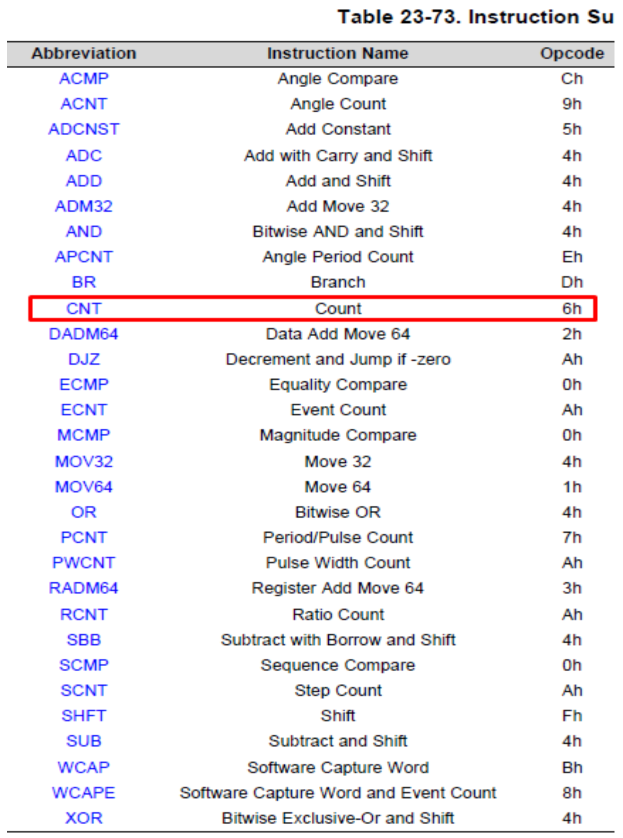
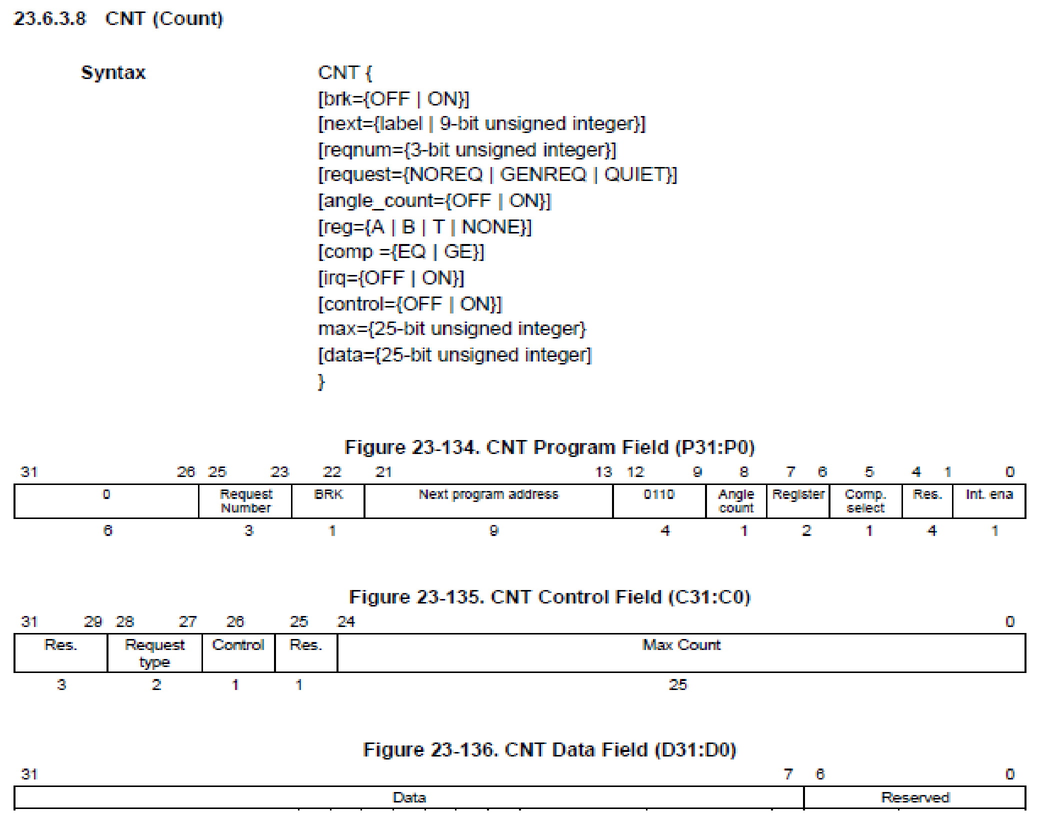


Table 23-73. Instruction Su

Abbreviation	Instruction Name	Opcode
ACMP	Angle Compare	Ch
ACNT	Angle Count	9h
ADCNST	Add Constant	5h
ADC	Add with Carry and Shift	4h
ADD	Add and Shift	4h
ADM32	Add Move 32	4h
AND	Bitwise AND and Shift	4h
APCNT	Angle Period Count	Eh
BR	Branch	Dh
CNT	Count	6h
DADM64	Data Add Move 64	2h
DJZ	Decrement and Jump if -zero	Ah
ECMP	Equality Compare	0h
ECNT	Event Count	Ah
MCMP	Magnitude Compare	0h
MOV32	Move 32	4h
MOV64	Move 64	1h
OR	Bitwise OR	4h
PCNT	Period/Pulse Count	7h
PWCNT	Pulse Width Count	Ah
RADM64	Register Add Move 64	3h
RCNT	Ratio Count	Ah
SBB	Subtract with Borrow and Shift	4h
SCMP	Sequence Compare	0h
SCNT	Step Count	Ah
SHFT	Shift	Fh
SUB	Subtract and Shift	4h
WCAP	Software Capture Word	Bh
WCAPE	Software Capture Word and Event Count	8h
XOR	Bitwise Exclusive-Or and Shift	4h



23.6.3.8 CNT (Count)

Syntax	CNT {
	[brk={OFF | ON}]
	[next={label | 9-bit unsigned integer}]
	[reqnum={3-bit unsigned integer}]
	[request={NOREQ | GENREQ | QUIET}]
	[angle_count={OFF | ON}]
	[reg={A | B | T | NONE}]
	[comp ={EQ | GE}]
	[irq={OFF | ON}]
	[control={OFF | ON}]
	max={25-bit unsigned integer}
	[data={25-bit unsigned integer]
	}

Figure 23-134. CNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5	4 1	0
0	Request Number	BRK	Next program address	0110	Angle count	Register	Comp. select	Res.	Int. ena
6	3	1	9	4	1	2	1	4	1

Figure 23-135. CNT Control Field (C31:C0)

31 29	28 27	26	25	24 0
Res.	Request type	Control	Res.	Max Count
3	2	1	1	25

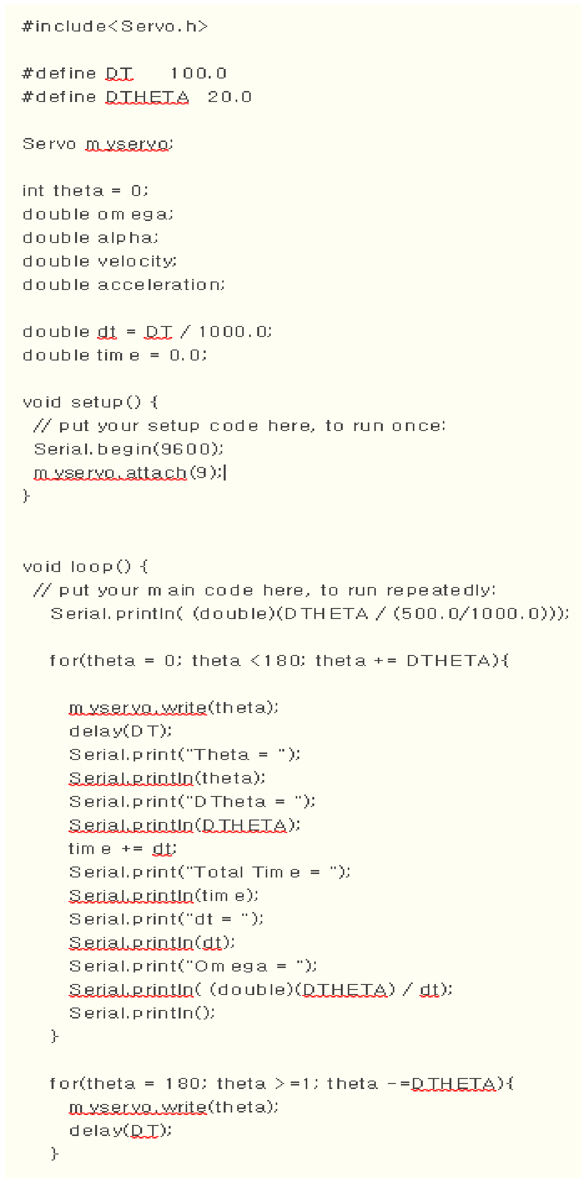
Figure 23-136. CNT Data Field (D31:D0)

31 7	6 0
Data	Reserved

	31~28	27~24	23~20	19~16	15~12	11~8	7~4	3~0	
Program :	0000	0000	0000	0000	0010	1100	1000	0000	→ 12~9 = 6h : CNT
	31~28	27~24	23~20	19~16	15~12	11~8	7~4	3~0	
Control :	0001	1111	1111	1111	1111	1111	1111	1111	→ Max Count : 2^25 = 32MB
	31~28	27~24	23~20	19~16	15~12	11~8	7~4	3~0	
DATA :	1111	1111	1111	1111	1111	1111	1000	0000	→ DATA : 2^25 = 32MB

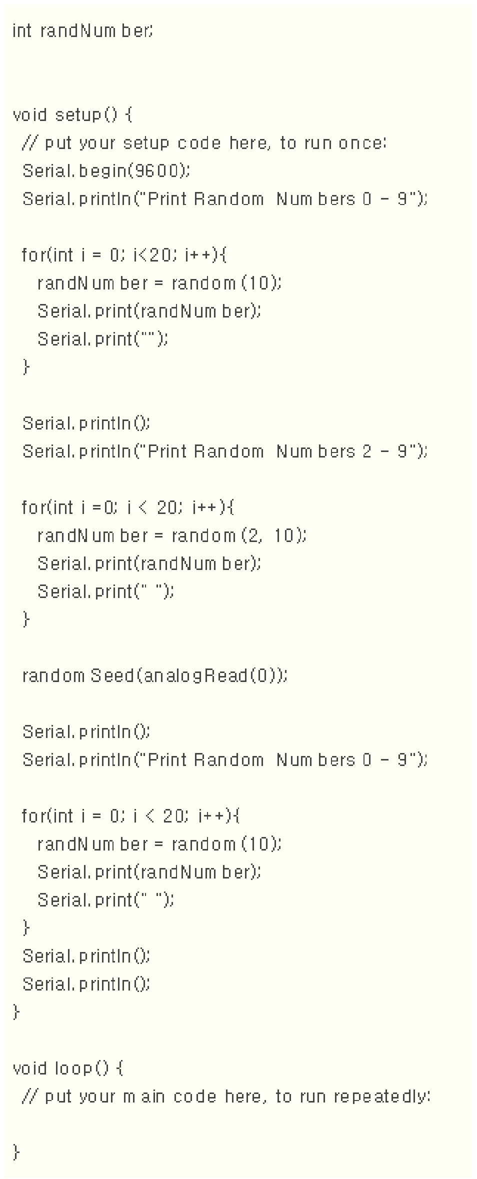
3. 수학 - 각속도와 각가속도 수식 코딩하기 (feat. Arduino PWM)

<1>



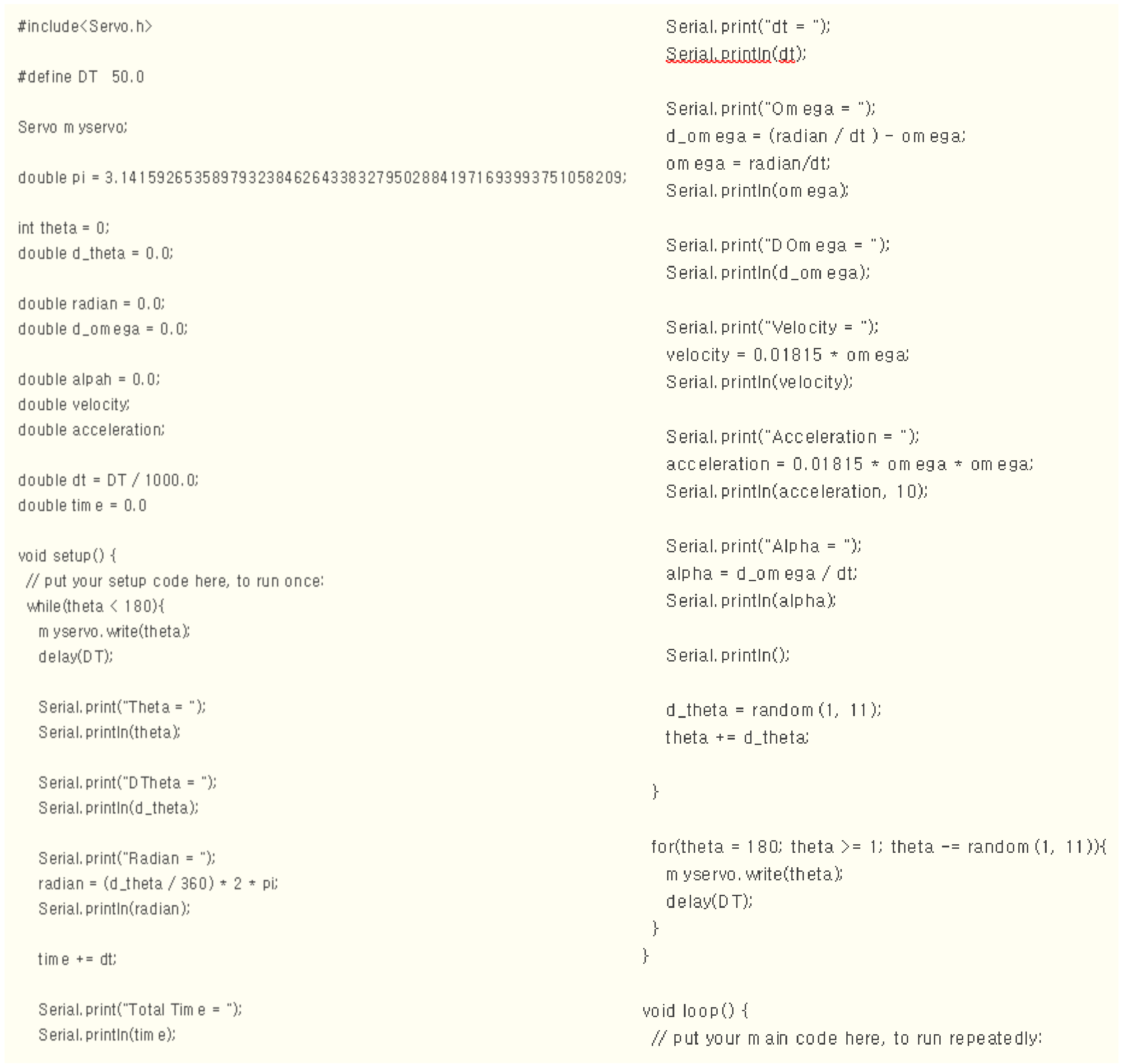
```
#include<Servo.h>

#define DT    100.0
#define DTHETA  20.0

Servo myservo;

int theta = 0;
double omega;
double alpha;
double velocity;
double acceleration;

double dt = DT / 1000.0;
double time = 0.0;

void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);
 myservo.attach(9);
}


void loop() {
 // put your main code here, to run repeatedly:
   Serial.println( (double)(DTHETA / (500.0/1000.0)));

   for(theta = 0; theta <180; theta += DTHETA){

     myservo.write(theta);
     delay(DT);
     Serial.print("Theta = ");
     Serial.println(theta);
     Serial.print("DTheta = ");
     Serial.println(DTHETA);
     time += dt;
     Serial.print("Total Time = ");
     Serial.println(time);
     Serial.print("dt = ");
     Serial.println(dt);
     Serial.print("Omega = ");
     Serial.println( (double)(DTHETA) / dt);
     Serial.println();
   }

   for(theta = 180; theta >=1; theta -=DTHETA){
     myservo.write(theta);
     delay(DT);
   }
```

<2>



```
int randNumber;


void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);
 Serial.println("Print Random  Numbers 0 - 9");

 for(int i = 0; i<20; i++){
   randNumber = random(10);
   Serial.print(randNumber);
   Serial.print("");
 }

 Serial.println();
 Serial.println("Print Random  Numbers 2 - 9");

 for(int i =0; i < 20; i++){
   randNumber = random(2, 10);
   Serial.print(randNumber);
   Serial.print(" ");
 }

 randomSeed(analogRead(0));

 Serial.println();
 Serial.println("Print Random  Numbers 0 - 9");

 for(int i = 0; i < 20; i++){
   randNumber = random(10);
   Serial.print(randNumber);
   Serial.print(" ");
 }
 Serial.println();
 Serial.println();
}

void loop() {
 // put your main code here, to run repeatedly:

}
```

<3>



```
#include<Servo.h>

#define DT   50.0

Servo myservo;

double pi = 3.14159265358979323846264338327950288419716939937510582097;

int theta = 0;
double d_theta = 0.0;

double radian = 0.0;
double d_omega = 0.0;

double alpah = 0.0;
double velocity;
double acceleration;

double dt = DT / 1000.0;
double time = 0.0

void setup() {
 // put your setup code here, to run once:
 while(theta < 180){
   myservo.write(theta);
   delay(DT);

   Serial.print("Theta = ");
   Serial.println(theta);

   Serial.print("DTheta = ");
   Serial.println(d_theta);

   Serial.print("Radian = ");
   radian = (d_theta / 360) * 2 * pi;
   Serial.println(radian);

   time += dt;

   Serial.print("Total Time = ");
   Serial.println(time);

   Serial.print("dt = ");
   Serial.println(dt);

   Serial.print("Omega = ");
   d_omega = (radian / dt ) - omega;
   omega = radian/dt;
   Serial.println(omega);

   Serial.print("DOmega = ");
   Serial.println(d_omega);

   Serial.print("Velocity = ");
   velocity = 0.01815 * omega;
   Serial.println(velocity);

   Serial.print("Acceleration = ");
   acceleration = 0.01815 * omega * omega;
   Serial.println(acceleration, 10);

   Serial.print("Alpha = ");
   alpha = d_omega / dt;
   Serial.println(alpha);

   Serial.println();

   d_theta = random(1, 11);
   theta += d_theta;

 }

 for(theta = 180; theta >= 1; theta -= random(1, 11)){
   myservo.write(theta);
   delay(DT);
 }
}

void loop() {
 // put your main code here, to run repeatedly:
```