I DSP,Xilinx zynq FPGA,MCU 및

Xilinx

zynq FPGA 프로그래밍 전문가 과정

강사 -INNOVA LEE(이상훈)

Gccom il3r@gmail.com

학생 - 윤지완

Yoonjw789 @naver.com

<서브 모터 구동(180도 회전후 다시 원상복귀되는 코드)-여기서 for문을 360으로 바꾸면 한바퀴가 돈다.>

```
#include<servo.h>
#define DT 100.0      //100ms=0.1초
#define DTHETA 20.0 //각도의 변화량
servo myservo;
int theta = 0; //초기 각도=0
double omega; //각속도
double alpha; // 각가속도=각속도의 변화량
double velocity;
double acceleration;
double dt = DT/1000.0; // 0.1
double time=0;
void setup()
{
   serial begine(9600);
   myservo.attack(9);
}
void loop()
{
   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);   //printf와 같은 동작 (입출력)
      time += dt;
      serial print("total time = ");
      serial println(time);
      serial print("dt = ");
      serial println(dt);
      serial print("omega = ");
      serial println((double)(DTHETA)/dt); //omega=200 radian/sec이 부분이 말이 안된다.
      serical println();
      //delay(1500);
   }
   for(theta = 180;theta>=1;theta-=DTHETA)
   {
      myservo.write(theta);
      delay(DT);
   }
}
```

<random 난수>

```
#includeservo.h>

int randunumber;

void setup()
{
   serial begine(9600);
   serial println("print dandom numbers 0-9");
```

```
  for(int i=0;i<20;i++)
  {
    randnumver = dandom(10);
    serial print(randnumver);
    serial print("");
  }
  serial println();
  sserial println("pint randomnumver 2-9");
  for(int i=0;i<20;i++0
  {
    randnumver= random(2,10);
    serial.print(randnumber);
    serial.print("");
  }
  randseed(analogread(0));
  serial.println();
  serail.println("print random numb0-9")
  for(int i=0;i<20;i++)
  {
    randnumber= random(10);
    serial.print(randnumber);
    serial.print("");
  }
    serial.println();
    seiral.println();
}
void loop()
```

```
{
<랜덤 난수를 이용해 서브 모터를 랜덤으로 돌아가게 하는 코드>
#include <Servo.h>

#define DT 50.0   //0.05s
Servo myservo;
double pi= 3.14159265358979323846264338327950288419716939937510582O9;

int theta = 0;
double d_theta =0.0;
double radian = 0.0;
double omega = 0.0;
double d_omega = 0.0;
double alpha = 0.0;
double velocity;
double acceleration;
double dt = DT/1000.0;
double time= 0.0;
void setup()
{
   Serial.begin(9600);
   randomSeed(analogRead(0));
   myservo.attach(9);
}

void loop()
{
```

```
while(theta<180)
{
  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= ");
  Serial.println(omega);
  d_omega=(radian/dt)-omega;
  omega= radian/dt;
  Serial.print("velocity =");
  velocity= 0.0185*omega;
  Serial.println(velocity);
  Serial.print("acceletation = ");
  acceleration = 0.0185*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);
  }
}
```

```
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);
```

```
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) 0x00000400U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U
             | (uint32) 0x00000000U;
```


23.4.18 NHET Direction Register (HETDIR)

N2HET1: offset = FFF7 B84Ch; **N2HET2:** offset = FFF7 B94Ch

Figure 23-73. N2HET Direction Register (HETDIR)

31	16
HETDIR	
R/W-0	

15	0
HETDIR	
R/W-0	

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

Table 23-34. N2HET Direction Register (HETDIR) Field Descriptions

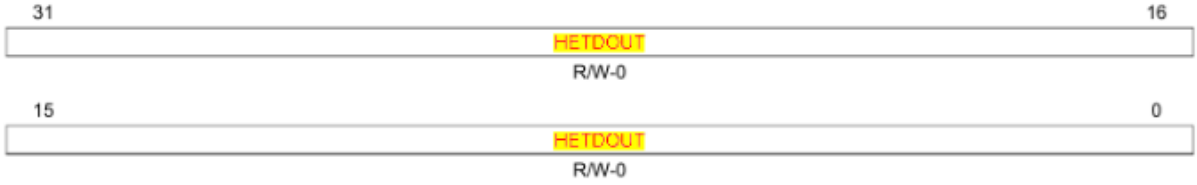
Bit	Field	Value	Description
31-0	HETDIR[n]		Data direction of NHET pins
		0	Pin HET[n] is an input (and its output buffer is tristated).
		1	Pin HET[n] is an output.

HET PIN을 모두 0으로 셋팅해 INPUT으로 사용한다.

23.4.20 N2HET Data Output Register (HETDOUT)

N2HET1: offset = FFF7 B854h; **N2HET2:** offset = FFF7 B954h

Figure 23-75. N2HET Data Output Register (HETDOUT)



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

Table 23-36. N2HET Data Output Register (HETDOUT) Field Descriptions

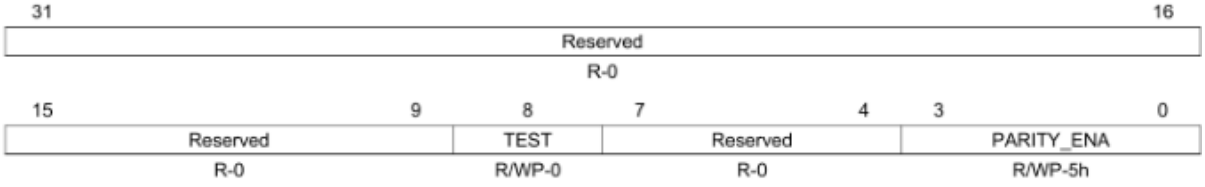
Bit	Field	Value	Description
31-0	HETDOUT[n]		Data out write. Writes to this bit will only take effect when the pin is configured as an output. The current logic state of the pin will be displayed by this bit even when the pin state is changed by writing to HETDSET or HETDCLR.
		0	Pin HET[n] is at logic low (0).
		1	Pin HET[n] is at logic high (1) if the HETPDR[n] bit = 0 or the output is in high-impedance state if the HETPDR[n] bit = 1.

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

23.4.26 Parity Control Register (HETPCR)

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

Figure 23-81. Parity Control Register (HETPCR)



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.

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

23.4.2 Prescale Factor Register (HETPFR)

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

Figure 23-57. Prescale Factor Register (HETPFR)

31																17	16
Reserved																	
R-0																	
15					11	10		8	7		6	5					0
Reserved						LRPFC			Reserved			HRPFC					
R-0						R/WP-0			R-0			R/WP-0					

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

```
/** - 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
```

23.4.24 N2HET Pull Disable Register (HETPULDIS)

Values in this register enable or disable the pull-up/-down functionality of the pins.
N2HET1: offset = FFF7 B864h; **N2HET2:** offset = FFF7 B964h

Figure 23-79. N2HET Pull Disable Register (HETPULDIS)

31	16
HETPULDIS	
R/W-n	

15	0
HETPULDIS	
R/W-n	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; n is device dependent, see device specific data sheet

Table 23-40. N2HET Pull Disable Register (HETPULDIS) Field Descriptions

Bit	Field	Value	Description
31-0	HETPULDIS[n]		Pull disable for N2HET pins
		0	The pull functionality is enabled on pin HET[n].
		1	The pull functionality is disabled on pin HET[n].

```
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;
```

23.4.23 N2HET Open Drain Register (HETPDR)

Values in this register enable or disable the open drain capability of the data pins.
N2HET1: offset = FFF7 B860h; **N2HET2:** offset = FFF7 B960h

Figure 23-78. N2HET Open Drain Register (HETPDR)

31	16
HETPDR	
R/W-0	

15	0
HETPDR	
R/W-0	

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

Table 23-39. N2HET Open Drain Register (HETPDR) Field Descriptions

Bit	Field	Value	Description
31-0	HETPDR[n]		Open drain control for HET[n] pins
		0	The pin is configured in push/pull mode.
		1	The pin is configured in open drain mode. The HETDOUT register controls the state of the output buffer:
			HETDOUT[n] = 0 The output buffer of pin HET[n] is driven low.
			HETDOUT[n] = 1 The output buffer of pin HET[n] is tristated.

```
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) 0x00000008U
              | (uint32) 0x00000004U
              | (uint32) 0x00000002U
              | (uint32) 0x00000001U;
```

23.4.13 HR Share Control Register (HETHRSH)

N2HET1: offset = FFF7 B834h; **N2HET2:** offset = FFF7 B934h

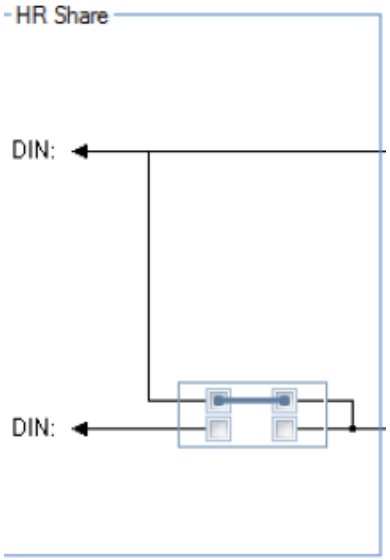
Figure 23-68. HR Share Control Register (HETHRSH)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
HR SHARE31/30	HR SHARE29/28	HR SHARE27/26	HR SHARE25/24	HR SHARE23/22	HR SHARE21/20	HR SHARE19/18	HR SHARE17/16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
HR SHARE15/14	HR SHARE13/12	HR SHARE11/10	HR SHARE9/8	HR SHARE7/6	HR SHARE5/4	HR SHARE3/2	HR SHARE1/0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

Table 23-29. HR Share Control Register (HETHRSH) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	HRSHARE n+1 / n		HR Share Bits Enables the share of the same pin for two HR structures. For example, if bit HRSHARE1/0 is set, the pin HET[0] will then be connected to both HR input structures 0 and 1. **Note:** If HR share bits are used, pins not connected to HR structures (the odd number pin in each pair) can be accessed as general inputs/outputs.
		0	HR Input of HET[n+1] and HET[n] are not shared.
		1	HR Input of HET[n+1] and HET[n] are shared; both measure pin HET[n].




```
#define hetPORT2 ((gioPORT_t *)0xFFF7B94CU)

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

Connecting Module	Memory Protection Scheme	Address Range		SYS.MSINENA Register Bit #	L2RAMW.MEMINT_ENA Register Bit #[3]
		Base Address	Ending Address		
L2 SRAM	ECC	0x08000000	0x0800FFFF	0	0
L2 SRAM	ECC	0x08010000	0x0801FFFF	0	1
L2 SRAM	ECC	0x08020000	0x0802FFFF	0	2
L2 SRAM	ECC	0x08030000	0x0803FFFF	0	3
L2 SRAM	ECC	0x08040000	0x0804FFFF	0	4
L2 SRAM	ECC	0x08050000	0x0805FFFF	0	5
L2 SRAM	ECC	0x08060000	0x0806FFFF	0	6
L2 SRAM	ECC	0x08070000	0x0807FFFF	0	7
MIBSPI5 RAM[4]	ECC	0xFF0A0000	0xFF0BFFFF	12	n/a
MIBSPI4 RAM[4]	ECC	0xFF060000	0xFF07FFFF	19	n/a
MIBSPI3 RAM[4]	ECC	0xFF0C0000	0xFF0DFFFF	11	n/a
MIBSPI2 RAM[4]	ECC	0xFF080000	0xFF09FFFF	18	n/a
MIBSPI1 RAM[4]	ECC	0xFF0E0000	0xFF0FFFFF	7	n/a
DCAN4 RAM	ECC	0xFF180000	0xFF19FFFF	20	n/a
DCAN3 RAM	ECC	0xFF1A0000	0xFF1BFFFF	10	n/a
DCAN2 RAM	ECC	0xFF1C0000	0xFF1DFFFF	6	n/a
DCAN1 RAM	ECC	0xFF1E0000	0xFF1FFFFF	5	n/a
MIBADC2 RAM	Parity	0xFF3A0000	0xFF3BFFFF	14	n/a
MIBADC1 RAM	Parity	0xFF3E0000	0xFF3FFFFF	8	n/a
NHET2 RAM	Parity	0xFF440000	0xFF45FFFF	15	n/a
NHET1 RAM	Parity	0xFF460000	0xFF47FFFF	3	n/a
HET TU2 RAM	Parity	0xFF4C0000	0xFF4DFFFF	16	n/a
HET TU1 RAM	Parity	0xFF4E0000	0xFF4FFFFF	4	n/a
DMA RAM	ECC	0xFFF80000	0xFFF80FFF	1	n/a
VIM RAM	ECC	0xFFF82000	0xFFF82FFF	2	n/a
FlexRay TU RAM	ECC	0xFF500000	0xFF51FFFF	13	n/a

23.2.2 N2HET RAM Organization

The N2HET RAM is organized into two sections. The first contains the N2HET program itself. The second contains parity protection bits for the N2HET program.

Each N2HET instruction is 96-bits wide but aligned to a 128-bit boundary. Instructions consist of three 32-bit fields: Program, Control, and Data. Instructions are separated by a fourth unimplemented address to force alignment to 128-bit boundaries.

The integrity of the N2HET program can be protected by Parity. Parity protection is enabled through the N2HET Parity Control Register (HETPCR).

Table 23-1 shows the base addresses for N2HET RAM and N2HET Parity RAM.

Table 23-1. N2HET RAM Base Addresses

N2HET1 Base Address	N2HET2 Base Address	Memory
0xFF46_0000	0xFF44_0000	N2HET Instruction RAM (Program/Control/Data)
0xFF46_2000	0xFF44_2000	N2HET Parity RAM

Table 23-2. N2HET RAM Bank Structure

N2HET Address	Host CPU or DMA Address Space				
Instruction	Program Field Address	Control Field Address	Data Field Address	Reserved Address	N2HET RAM Bank
000h	XX0000h	XX0004h	XX0008h	XX000Ch	A
001h	XX0010h	XX0014h	XX0018h	XX001Ch	B
002h	XX0020h	XX0024h	XX0028h	XX002Ch	C
003h	XX0030h	XX0034h	XX0038h	XX003Ch	D
004h	XX0040h	XX0044h	XX0048h	XX004Ch	A
:	:	:	:	:	:
03Fh	XX03F0h	XX03F4h	XX03F8h	XX03FCh	D
040h	XX0400h	XX0404h	XX0408h	XX040Ch	A
:	:	:	:	:	:
1FFh	XX1FF0h	XX1FF4h	XX1FF8h	XX1FFCh	D

```
/* CNT: Timebase    //타이머전용 기계어
   *        - Instruction                = 0
   *        - Next instruction           = 1
   *        - Conditional next instruction = na
   *        - Interrupt                  = na
   *        - Pin                        = na
   *        - Reg                        = T
   */
   {
```

```
/* Program */
```

0x00002C80U,

/* Control */

0x01FFFFFU,

/* Data */

0xFFFFFF80U,

/* Reserved */

0x00000000U

이 값들이 N2HET BANK에 PROGRAM,CONTROL DATA에 들어가는 값들이다.

p:0x2c80 // 13bit은 next program address c언어에서 보면 cnt는 배열의 0번째이다 next라는것은 다음배열로 가라는 것이다.

// 10,11bit opcode.

c:0x01fffffff

// 0x 0000 0001 1111 1111 1111 1111 1111 1111 (0~24번까지 bit를 사용하는데 0~24bit는 max_count의 특별한 bit이다.

이 부분은 program filed에서 각각 사용하는 동작들을 얼마만큼 사용할수 있는지 조건을 알려주는 것이다.

데이터 필드에 허용되는 최대 카운트 값을 정의하는 25 비트 정수 값을 지정합니다. 데이터 필드의 카운트가 max와 같으면 데이터 필드가 0으로 재설정되고 Z 시스템 플래그가 1로 설정됩니다.

D:0xffffff80

0x 1111 1111 1111 1111 1111 1111 1000 0000

0~6bit를 0으로 설정

7~31bit까지 1로 설정

0~6bit까지는 DATA field 를 보면 reserved로 되어있다 이 부분은 읽을수고 쓸스도없는 그냥 황무지 같은 bit이다.

7~31bit까지는 카운터 역활을 하는 bit들이다.이 녀석들은 731bit에 정수를 지정하는 역활을 한다.

Cycles	One or two One cycle (time mode), two cycles (angle mode)
Register modified	Selected register (A, B or T)
Description	This instruction defines a virtual timer. The counter value stored in the data field [D31:7] is incremented unconditionally on each execution of the instruction when in time mode (angle count bit [P8] = 0). When the count reaches the maximum count specified in the control field, the counter is reset. It takes one cycle in this mode. In angle mode (angle count bit [P8] = 1), CNT needs data from the software angle generator (SWAG). When in angle count mode the angle increment value will be 0 or 1. It takes two cycles in this mode.

angle_count	Specifies when the counter is incremented. A value of ON causes the counter value to be incremented only if the new angle flag is set (NAF_global = 1). A value of OFF increments the counter each time the CNT instruction is executed. Default value for this field is OFF.
comp	When set to EQ the counter is reset, when it is equal to the maximum count. When set to GE the counter is reset, when it is greater or equal to the maximum count. Default: GE.
irq	ON generates an interrupt when the counter overflows to zero. The interrupt is not generated until the data field is reset to zero. If irq is set to OFF, no interrupt is generated. Default: OFF.
max	Specifies the 25-bit integer value that defines the maximum count value allowed in the data field. When the count in the data field is equal to max, the data field is reset to 0 and the Z system flag is set