TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/5/10
수업일수	51 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

수치미분

등각속도의 수치미분

```
#include <Servo.h>

#define DT    100.0 //시간변화량 0.1 초
#define DTHETA    20.0    //각도의 변화량이 20 도

Servo myservo;
int theta = 0; //각도회전을 안하니 0 으로시작
double omega; //각속도
double alpha; //각가속도
double velocity; //속도
double acceleration; //가속도

double dt = DT / 1000.0; //0.1
double time = 0.0; //초기값 0 으로 세팅

void setup() {
Serial.begin(9600);
myservo.attach(9); //9 번핀을 pwm 으로사용
}

void loop() {
Serial.println((double)(DTHETA / (500.0/1000.0))); //DTHETA 를 1/2 로 나눔, DTHETA*2 = 40
for(theta = 0; theta < 180; theta += DTHETA) // theta 의 증가폭을 20 으로 바꿈
{
   myservo.write(theta);//0 이들어옴
   delay(DT);//0.1 초 딜레이
   Serial.print("Theta = ");
```

```
  Serial.println(theta); //theta 값 출력
  Serial.print("DTheta = ");
  Serial.println(DTHETA); //DTHETA 20 출력
  time += dt; //0.1 초 지남을 알수있음
  Serial.print("Total Time = ");
  Serial.println(time); //총시간
  Serial.print("dt = ");
  Serial.println(dt); //디버깅용으로 시간변화량출력
  Serial.print("Omega = "); //각도의변화량을 나타냄. 항상 200 나타남
  Serial.println((double)(DTHETA) / dt); //각도의 변화량 / 시간의 변화량 . 도/s 에서 rad/s 로 단위 계산해야함.
  Serial.println();
}
for(theta = 180; theta >= 1; theta -= DTHETA)
{
  myservo.write(theta);
  delay(DT);
}
}
```

결과

	1	2	3	4
40.00	Theta=0	Theta=20	Theta=40	Theta=60
	DTheta=20.00	DTheta=20.00	DTheta=20.00	DTheta=20.00
	Total time = 0.10	Total time = 0.20	Total time = 0.30	Total time = 0.40
	dt=0.10	dt=0.10	dt=0.10	dt=0.10
	Omega =200.00	Omega =200.00	Omega =200.00	Omega =200.00
5	6	7	8	9
Theta=80	Theta=100	Theta=120	Theta=140	Theta=160
DTheta=20.00	DTheta=20.00	DTheta=20.00	DTheta=20.00	DTheta=20.00
Total time = 0.50	Total time = 0.60	Total time = 0.70	Total time = 0.80	Total time = 0.90

dt=0.10	dt=0.10	dt=0.10	dt=0.10	dt=0.10
Omega =200.00	Omega =200.00	Omega =200.00	Omega =200.00	Omega =200.00

시간의 변화량을 100ms로 하고 세타(각도)의 변화량을 20도로 함.

처음 세타는 0으로 초기화, 실행한지 총 시간량 time도 0으로 초기화 함. dt라는 이름으로 DT라는 ms단위에서 초 단위로 바꾸어 0.1초로 세팅. DTHETA만큼 각도변화를 주며 각속도, 각가속도, 속도, 가속도의 값을 측정한다.

시간 변화량은 0.1초로 일정하고 각도 변화량도 20도로 일정하다. 그러므로 0부터 시작한 total time과 Theta는 각각 0.1, 20씩 증가하여 각속도 w가 일정한 20/0.1 == 200의 값이 나오게 된다. 등각속도이므로 각가속도 알파는 0이 된다.

Rand 값 생성

```
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));//srandtimeNULL 이랑 같은 것?
  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:

}
```

각가속도를 보기위한 랜덤 값 설정

```
#include<Servo.h>

#define DT   50.0

Servo myservo;

double pi=3.14159265358979323846264338462643383279502884197169399375105820 9; //무리수 파이, 길게 해야 정밀도가 높음

int theta = 0; //각도
double d_theta = 0.0; //각도의 변화량

double radian = 0.0; //pi/180
double omega = 0.0; //라디안으로 변환된 값의 각속도
double d_omega = 0.0; //각속도 변화량

double alpha = 0.0; //각 가속도
double velocity; //속도
double acceleration; //rw^2

double dt =DT / 1000.0; //1/20 == 0.05 초
double time = 0.0; //전체시간

void setup() {
Serial.begin(9600);
randomSeed(analogRead(0)); //랜덤사용하기위한
myservo.attach(9);
}
```

```
void loop() {
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);
```

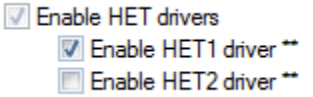
```
    Serial.print("Velocity = ");
    velocity = 0.01815 * omega; //0.01815 : 날개길이
    Serial.println(velocity);

    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);
}
}
```

결과(계산값)

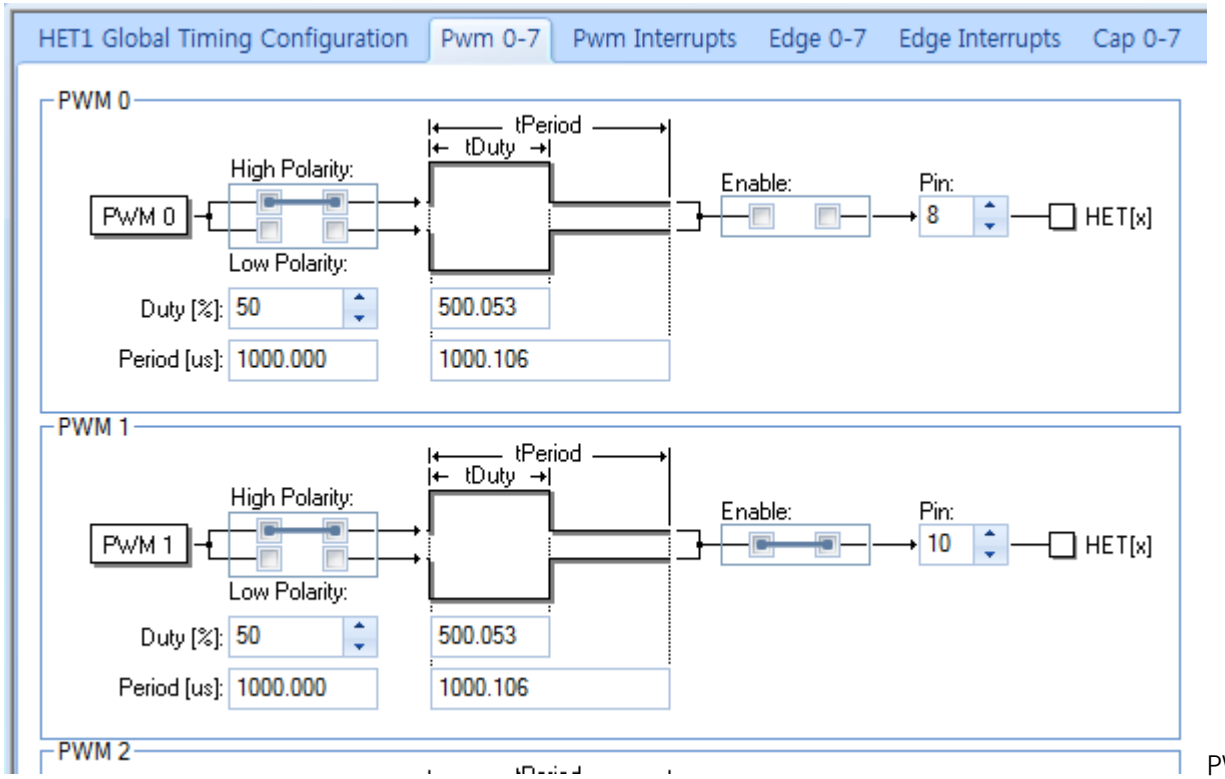
처음	랜덤 theta 가 10 일 때	랜덤 theta 가 5 일 때
Theta = 0	Theta = 10	Theta = 5
DTheta = 0	DTheta = 10	DTheta = 5
Radian = 0	Radian = 0.1745	Radian = 0.08726
Total Time = 0.05	Total Time = 0.1	Total Time = 0.15
dt = 0.05	dt = 0.05	dt = 0.05
Omega = 0	Omega = 3.49	Omega = -1.74467
DOmega = 0	DOmega = 3.49	DOmega = -1.7448
Velocity = 0	Velocity = 0.0633435	Velocity = -0.031665
Acceleration = 0	Acceleration = 0.221068815	Acceleration = 1.478365
Alpha = 0	Alpha = 69.8	Alpha = 34.896

HetPWM

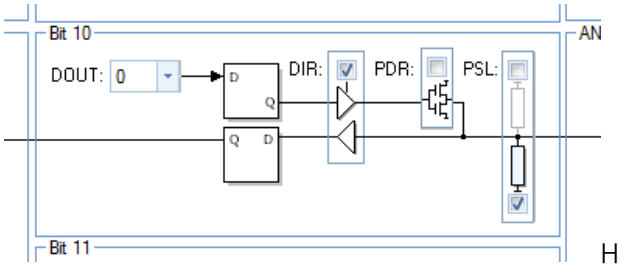
설정



driver 설정



PWM1 을 Enable 시킴

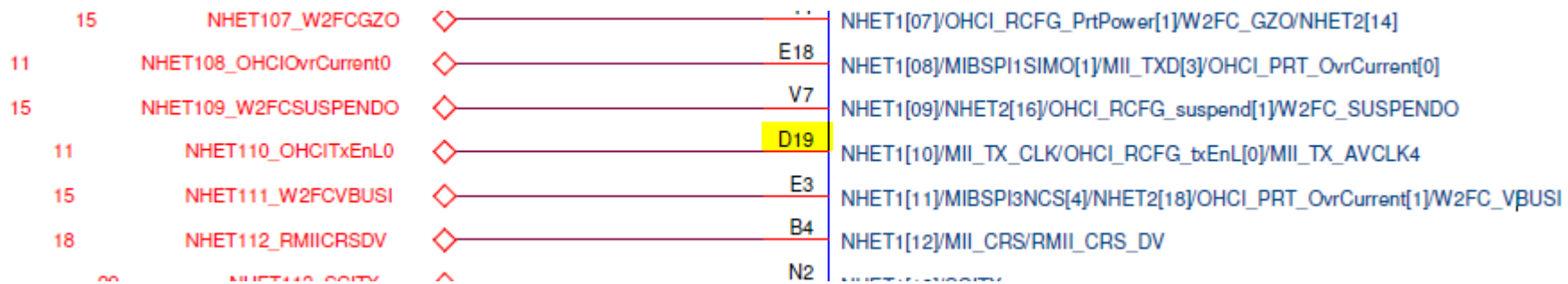


HET1 - Pin 8-15 - Bit 10 에서 DIR 클릭

Generate Code 하여 코드 생성.

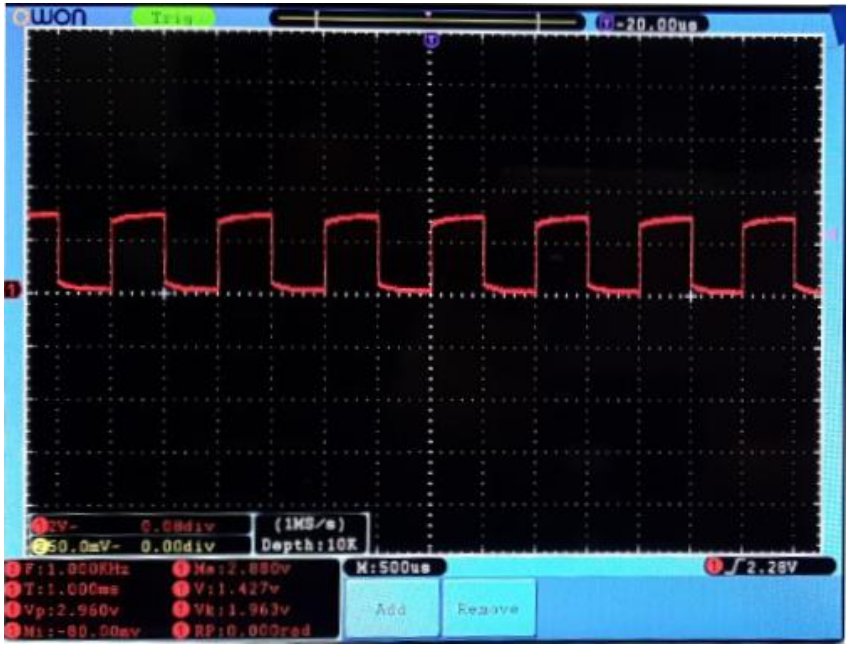
소스코드

```
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_het.h"

int main(void)
{

    hetInit();

    while(1);

    return 0;
}
```



회로 연결 할 때 D19 와 저항, LED 를 차례대로 연결한 뒤 GND 에 연결한다.

결과



위에서 설정한 대로 duty 50%인 펄스파가 측정된다.

duty rate 를 길게 조정하면 LED 의 밝기가 밝아지고, 짧게 조정하면 어두워진다.

또한 회로에 연결 된 LED 는 사람의 눈으로 측정 불가능한 주파수로 신호가 가기 때문에 그냥 켜져 있는 것처럼 보인다. 만약 period 를 500000[us]로 바꾸면 사람이 구별 가능한 주파수로 눈으로 확인 가능하다.

분석

hetInit

```
1075 void hetInit(void)
1076 {
1077     /** @b initialize @b HET */
1078
1079     /** - Set HET pins default output value */
1080     hetREG1->DOUT = (uint32)((uint32)0U << 31U)
1081                   | (uint32)((uint32)0U << 30U)
1082                   | (uint32)((uint32)0U << 29U)
1083                   | (uint32)((uint32)0U << 28U)
1084                   | (uint32)((uint32)0U << 27U)
1085                   | (uint32)((uint32)0U << 26U)
1086                   | (uint32)((uint32)0U << 25U)
1087                   | (uint32)((uint32)0U << 24U)
1088                   | (uint32)((uint32)0U << 23U)
1089                   | (uint32)((uint32)0U << 22U)
1090                   | (uint32)((uint32)0U << 21U)
1091                   | (uint32)((uint32)0U << 20U)
1092                   | (uint32)((uint32)0U << 19U)
1093                   | (uint32)((uint32)0U << 18U)
1094                   | (uint32)((uint32)0U << 17U)
1095                   | (uint32)((uint32)0U << 16U)
1096                   | (uint32)((uint32)0U << 15U)
1097                   | (uint32)((uint32)0U << 14U)
1098                   | (uint32)((uint32)0U << 13U)
1099                   | (uint32)((uint32)0U << 12U)
1100                   | (uint32)((uint32)0U << 11U)
1101                   | (uint32)((uint32)0U << 10U)
1102                   | (uint32)((uint32)0U << 9U)
1103                   | (uint32)((uint32)0U << 8U)
1104                   | (uint32)((uint32)0U << 7U)
1105                   | (uint32)((uint32)0U << 6U)
1106                   | (uint32)((uint32)0U << 5U)
1107                   | (uint32)((uint32)0U << 4U)
1108                   | (uint32)((uint32)0U << 3U)
1109                   | (uint32)((uint32)0U << 2U)
1110                   | (uint32)((uint32)0U << 1U)
1111                   | (uint32)((uint32)0U << 0U);
```

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.

:1080 ~ 1111 0~31 비트를 전부 0 으로 초기화

Pin HET[0~31]가 low 상태이다.

```
1113    /** - Set HET pins direction */
1114    hetREG1->DIR = (uint32) 0x00000000U
1115                 | (uint32) 0x00000000U
1116                 | (uint32) 0x00000000U
1117                 | (uint32) 0x00000000U
1118                 | (uint32) 0x00000000U
1119                 | (uint32) 0x00000000U
1120                 | (uint32) 0x00000000U
1121                 | (uint32) 0x00000000U
1122                 | (uint32) 0x00000000U
1123                 | (uint32) 0x00000000U
1124                 | (uint32) 0x00000000U
1125                 | (uint32) 0x00000000U
1126                 | (uint32) 0x00000000U
1127                 | (uint32) 0x00000000U
1128                 | (uint32) 0x00000000U
1129                 | (uint32) 0x00000000U
1130                 | (uint32) 0x00000000U
1131                 | (uint32) 0x00000000U
1132                 | (uint32) 0x00000000U
1133                 | (uint32) 0x00000000U
1134                 | (uint32) 0x00000000U
1135                 | (uint32) 0x00000400U
1136                 | (uint32) 0x00000000U
1137                 | (uint32) 0x00000000U
1138                 | (uint32) 0x00000000U
1139                 | (uint32) 0x00000000U
1140                 | (uint32) 0x00000000U
1141                 | (uint32) 0x00000000U
1142                 | (uint32) 0x00000000U
1143                 | (uint32) 0x00000000U
1144                 | (uint32) 0x00000000U
1145                 | (uint32) 0x00000000U;
```

: 1135 0x00000400U 이므로 10 번째 비트 1 로 셋팅

Pin HET[10]를 output 으로 설정.

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.

0 이면 입력,

1 이면 출력.

:1148 hetREG1->PDR 모든비트 0

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.

0 이면 push/pull 모드

1 이면 open drain 모드

모두 0 이므로

:1182 hetREG1->PULDIS

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].

0 이면 pull 방식 사용

1 이면 pull 방식을 사용하지 않음

: 1216 hetREG1->PSL 0 으로 셋

Table 23-41. N2HET Pull Select Register (HETPSL) Field Descriptions

Bit	Field	Value	Description
31-0	HETPSL[n]		Pull select for NHET pins
		0	The pull down functionality is enabled if corresponding bit in HETPULDIS is 0.
		1	The pull up functionality is enabled if corresponding bit in HETPULDIS is 0.

0 이면 pull down 사용

1 이면 pull up 사용

0 이니까 pull down 사용. HALcogen 설정에서도 바꾸지 않았음.

: 1250 hetREG1->HRSH

```
1249    /** - Set HET pins high resolution share */
1250    hetREG1->HRSH = (uint32) 0x00008000U
1251                  | (uint32) 0x00004000U
1252                  | (uint32) 0x00002000U
1253                  | (uint32) 0x00001000U
1254                  | (uint32) 0x00000000U
1255                  | (uint32) 0x00000000U
1256                  | (uint32) 0x00000000U
1257                  | (uint32) 0x00000000U
1258                  | (uint32) 0x00000000U
1259                  | (uint32) 0x00000000U
1260                  | (uint32) 0x00000000U
1261                  | (uint32) 0x00000000U
1262                  | (uint32) 0x00000008U
1263                  | (uint32) 0x00000004U
1264                  | (uint32) 0x00000002U
1265                  | (uint32) 0x00000001U;
```

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].

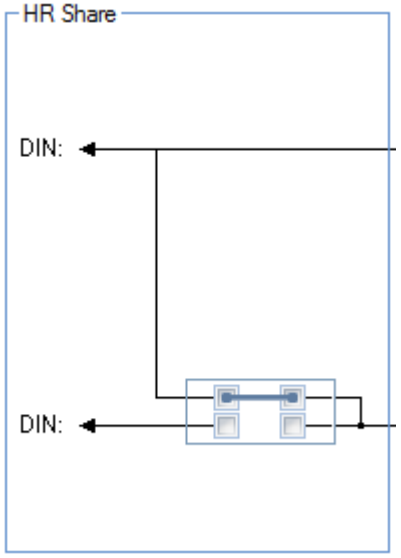
15, 14, 13, 12, 3, 2, 1, 0 비트 0 으로 셋

0 이면 HET[n+1}과 HET[n] 공유하지 않음

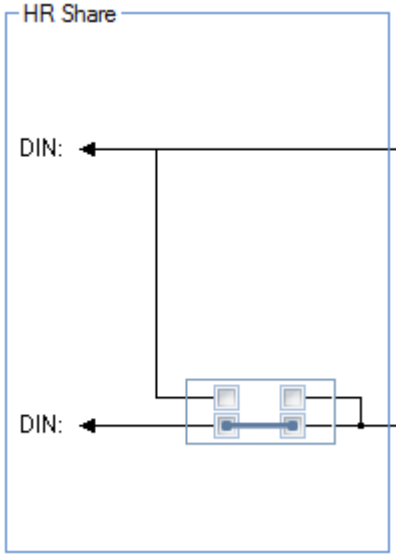
1 이면 공유함, 신호선이 공유된다.

HAL 확인 pin 0-7, pin 8-15

공유할 때



공유하지 않을 때



: 1268 hetREG1->AND

Table 23-28. AND Share Control Register (HETAND) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	ANDSHARE n+1 / n		AND Share Enable Enable the AND sharing of the same pin for two HR structures. For example, if bit ANDSHARE1/0 is set, the pin HET[0] will then be commanded by a logical AND of both HR structures 0 and 1. **Note:** If HR AND 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 Output of HET[n+1] and HET[n] are not AND shared.
		1	HR Output of HET[n+1] and HET[n] are AND shared onto pin HET[n].

0 일 때 HR 의 출력의 HET[n+1]과 HET[n]이 AND 게이트 통해 공유되지 않음

1 일때는 공유된다.

모든 비트가 0 이니 공유되지 않는다.

:1286 hetREG1->XOR

Table 23-30. XOR Share Control Register (HETXOR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	XORSHARE n+1 / n		XOR Share Enable Enable the XOR-share of the same pin for two output HR structures. For example, if bit XORSHARE1/0 is set, the pin HET[0] will then be commanded by a logical XOR of both HR structures 0 and 1. **Note:** If XOR 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 Output of HET[n+1] and HET[n] are not XOR shared.
		1	HR Output of HET[n+1] and HET[n] are XOR shared onto pin HET[n].

모든 비트가 0 으로 셋

HR 출력의 HET[n+1]과 HET[n]이 XOR 게이트로 공유되지 않음.

1 일때는 공유된다.

```
1306    /** - Setup prescaler values
1307    *      - Loop resolution prescaler
1308    *      - High resolution prescaler
1309    */
1310    hetREG1->PFR = (uint32)((uint32) 6U << 8U)
1311                 | ((uint32) 0U);
1312
1313
1314   /** - Parity control register
1315    *      - Enable/Disable Parity check
1316    */
1317    hetREG1->PCR = (uint32) 0x00000005U;
```

: 1310 8,9 번 비트 1 로 셋, 0 번 비트 0 으로 고정

:1317 2, 0 번 비트 1 로 셋

: 1310 hetREG1->PFR

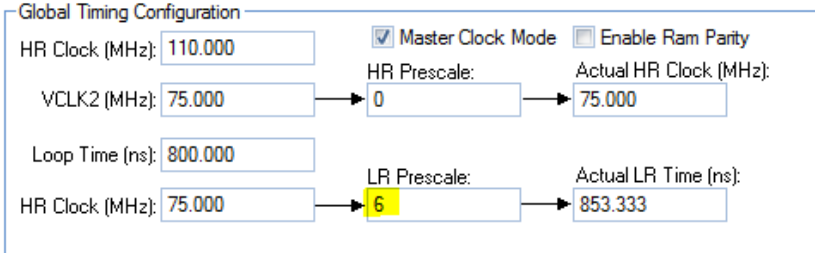
Table 23-17. Prescale Factor Register (HETPFR) Field Descriptions

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

LRPFC 는 64 분주, HRPFC 는 분주 없이.

HR prescale 이 0 이고 LR Prescale 은 6 이다.

75M / 64 = 1171875[Hz] = 853.333[ns]



:1317 hetREG1->PCR

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.

1, 2 비트 1 로 셋

5 니까 parity 체크 허용. 에러에 대한 검사 안함.

: 1323 (void)memcpy((void *)hetRAM1, (const void *)het1PROGRAM, sizeof(het1PROGRAM));

```
1323    (void)memcpy((void *)hetRAM1, (const void *)het1PROGRAM, sizeof(het1PROGRAM));
```

FPFA 칩이 PWM 파형을 만들어 내는 동작을 함. 동작 제어는 이 명령어에서 제어한다.

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

여기에 해당하는 주소 값으로 het1PROGRAM 을 복사하여 붙여넣는다.

```
73 static const hetINSTRUCTION_t het1PROGRAM[58U] =
74 {
75     /* CNT: Timebase
76     *       - Instruction                 = 0
77     *       - Next instruction            = 1
78     *       - Conditional next instruction = na
79     *       - Interrupt                   = na
80     *       - Pin                         = na
81     *       - Reg                         = T
82     */
83     {
84         /* Program */
85         0x00002C80U,
86         /* Control */
87         0x01FFFFFFU,
88         /* Data */
89         0xFFFFFF80U,
90         /* Reserved */
91         0x00000000U
92     },
```

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

het1PROGRAM[58U]는 이러한 명령어들을 나타내는 배열이다.

CNT

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

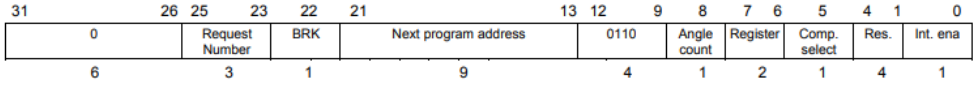


Figure 23-135. CNT Control Field (C31:C0)

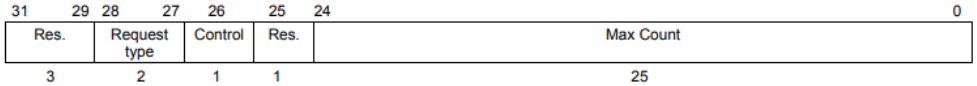
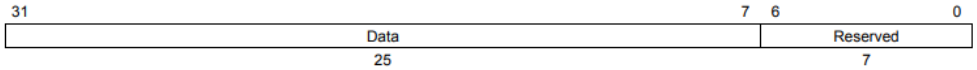


Figure 23-136. CNT Data Field (D31:D0)



R 7,10,11,13

13 : Next program address 가 1 이니 1 로가라. == 배열 1 번째로 가라

11,10 : 0110 가 opcode 임 (mov r1, r2 같은 명령어에서 mov 에 해당), 9~12 번째를 해석하면 mov 인지 cnt 인지 구별 가능

7 : Register A,B,T 의 범용 레지스터 중 하나를 받음.

8 번 0 이니 angle count OFF

C 0~24 비트까지 1

0~24 : Max Count 1111111111111111111111111 (25 개),

데이터 필드의 카운트가 max 와 같으면 필드가 0 으로 설정되어 Z flag 가 1 로 설정된다.

D 7~31 비트 1

7~31 : 카운터 역할을 하는 25 비트 정수 값을 1111111111111111111111111(25 개)로 지정