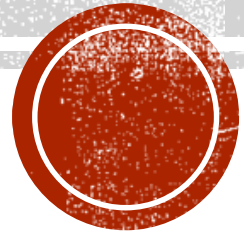
3조

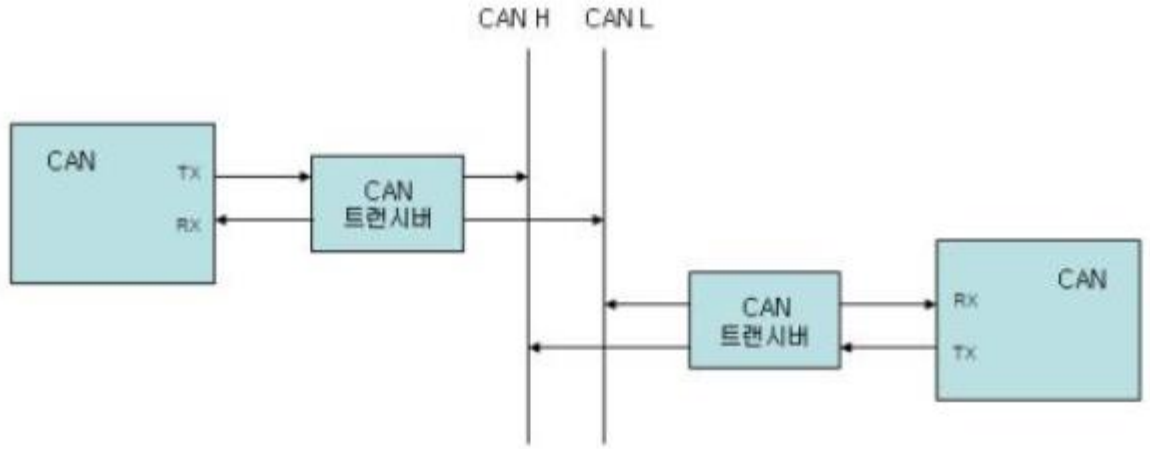
자율주행 자동차 1주차 18.07.04 ~ 18.07.11



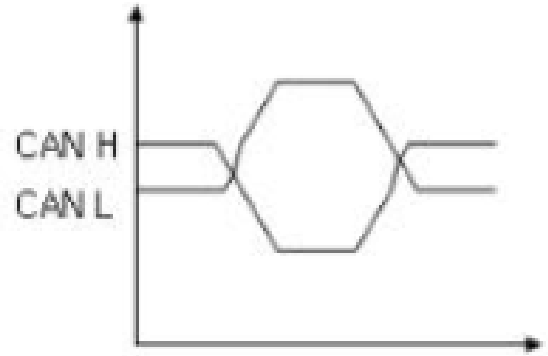
TMS570 BOARD CAN 통신 - 장성환

- **CAN** 통신

- **Multi master – multi slave** 구조가 가능하여 **N:N** 통신이 가능.
- **Differential** 신호 방식을 사용하여 노이즈에 강함.
- 주로 중장거리 보드 간 통신에 사용, 별도의 트랜시버를 필요로 한다.



- 동작 구조



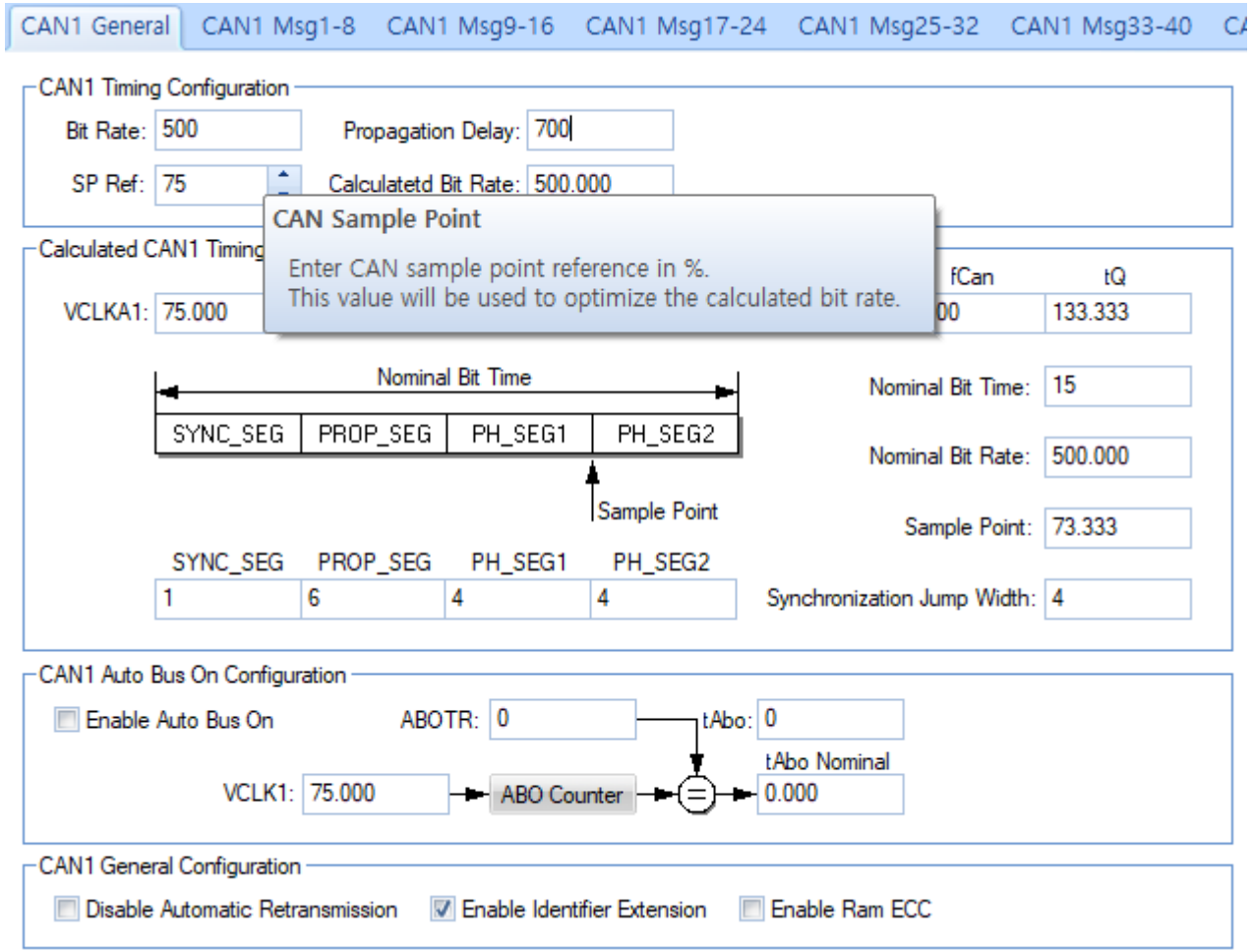
TX에서 나온 **DATA**를 1번 **invert** 한 신호를 **CAN H**에, 2번 **invert** 한 신호를 **CAN L**에 담는다.

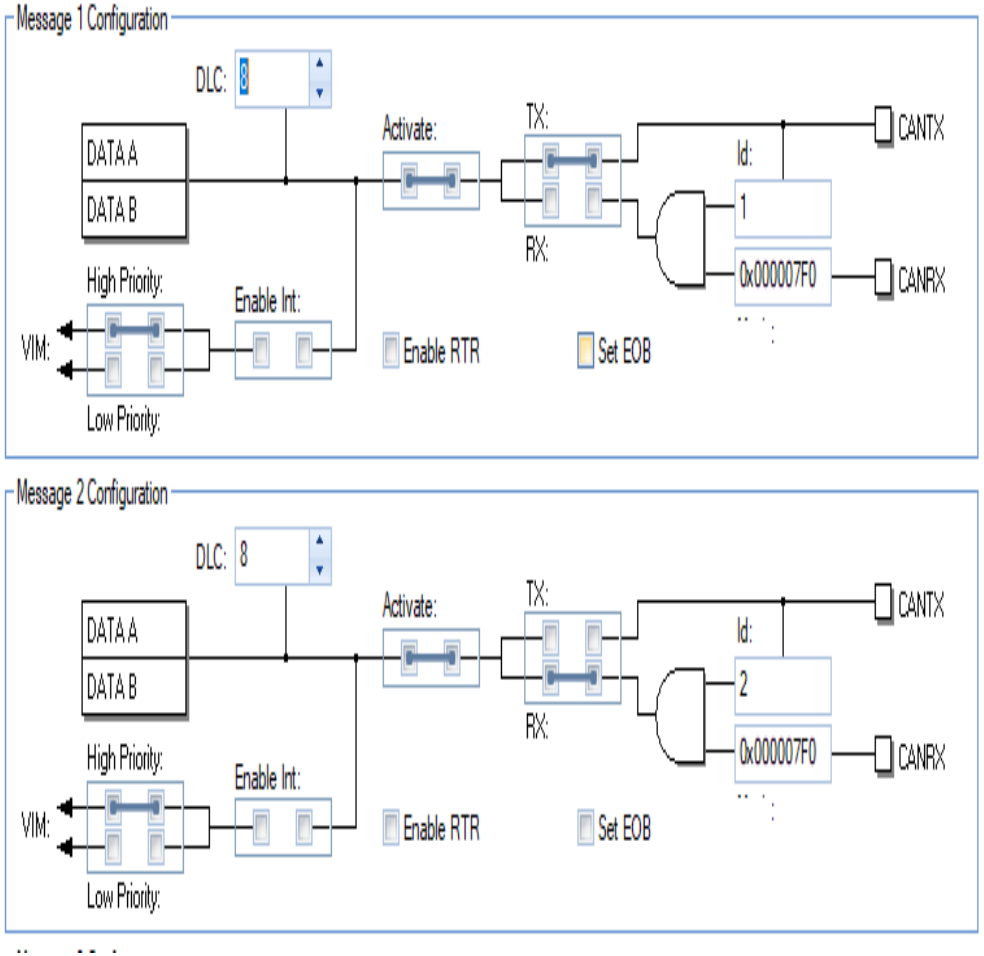
두 신호의 전위차로 오리지날 신호를 파악하게 된다.

ID	Mail Box 1
ID	Mail Box 2
ID	Mail Box 3
ID	Mail Box 4
ID	Mail Box 5

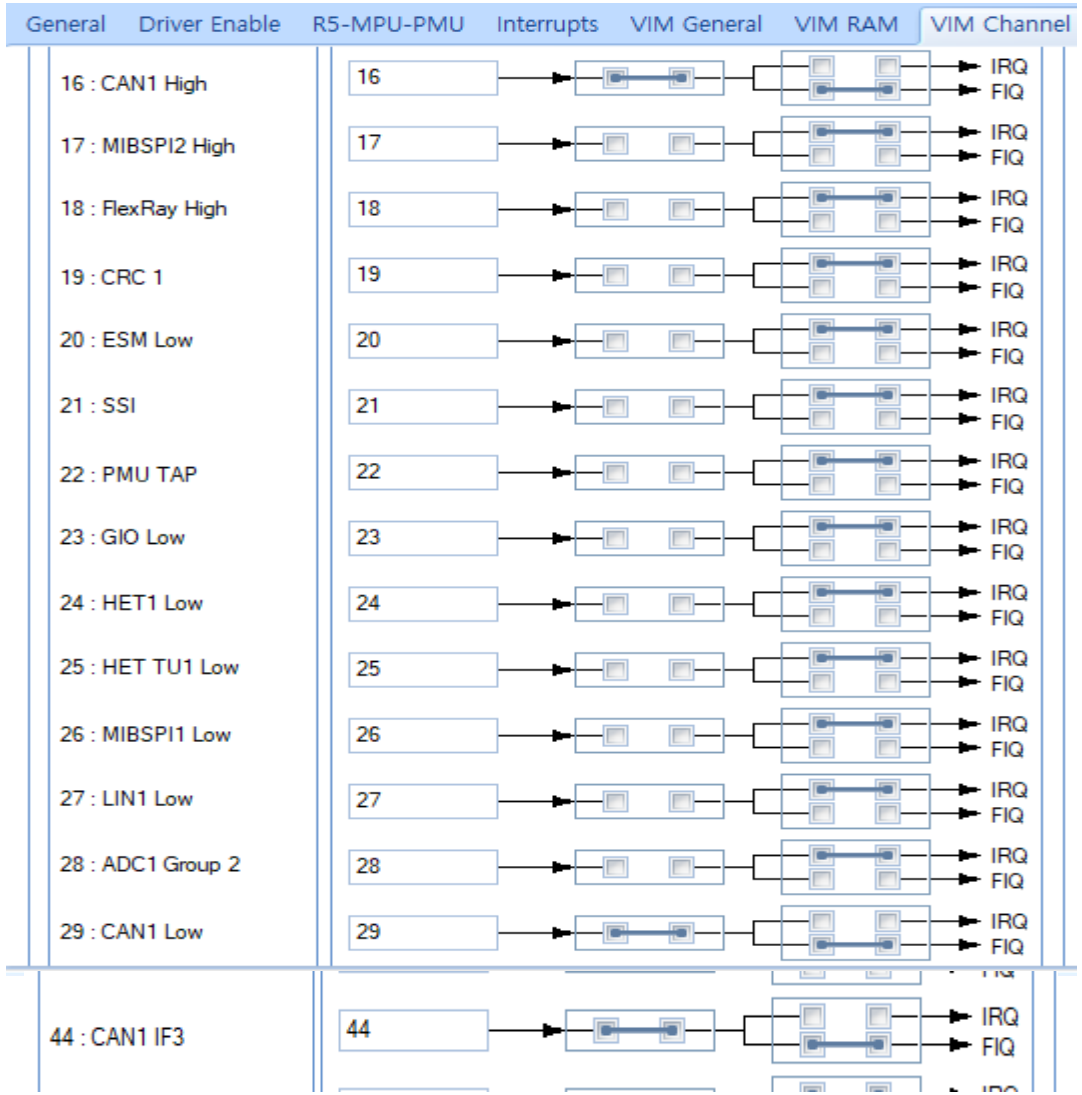
실제 데이터에는 **Mail Box**에 저장되고 보내진다.
각각의 **ID**가 존재하며 이 **ID**는 **address**의 역할을 하게 된다.

- **TMS board HALCOGEN 설정**



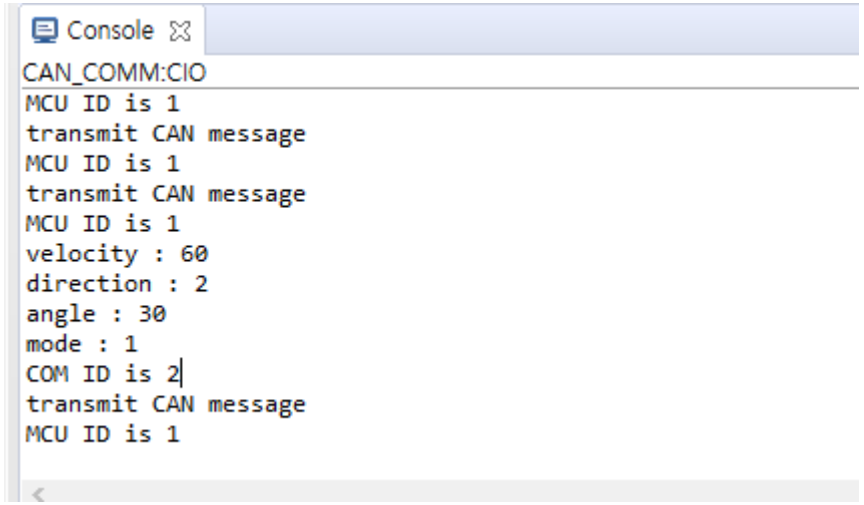


- TMS board HALCOGEN 설정

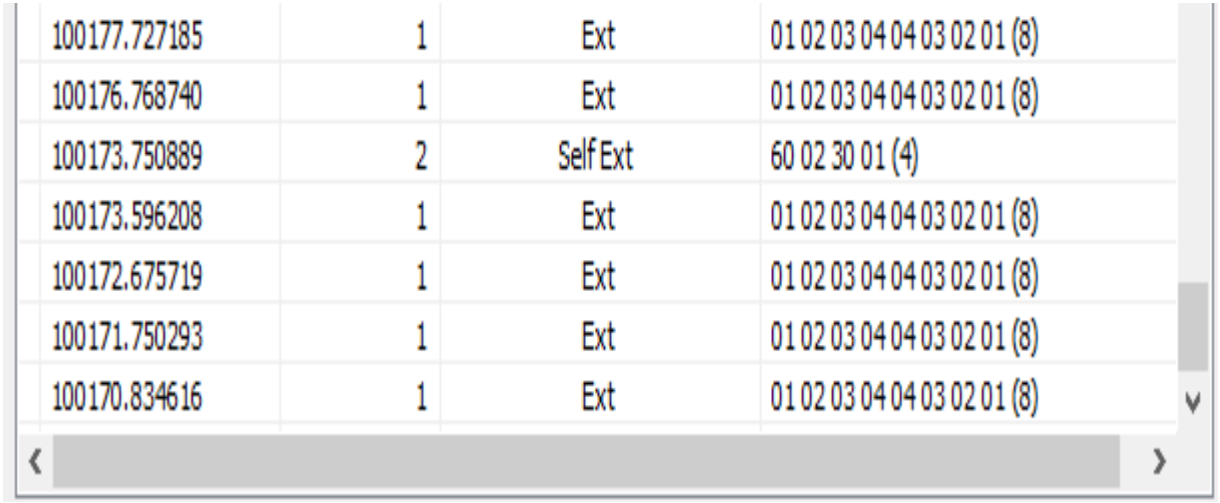


- 실험 결과

MCU Console창 출력



컴퓨터 CAN program 출력



100177.727185	1	Ext	01 02 03 04 04 03 02 01 (8)
100176.768740	1	Ext	01 02 03 04 04 03 02 01 (8)
100173.750889	2	Self Ext	60 02 30 01 (4)
100173.596208	1	Ext	01 02 03 04 04 03 02 01 (8)
100172.675719	1	Ext	01 02 03 04 04 03 02 01 (8)
100171.750293	1	Ext	01 02 03 04 04 03 02 01 (8)
100170.834616	1	Ext	01 02 03 04 04 03 02 01 (8)

- 문제점

- **Interrupt** 방식으로 **CAN** 통신 구현하였을 때, **IRQ** 방식의 인터럽트를 사용할 경우에는 작동이 정상적으로 이루어 지지 않는 경우가 발생하였다.

- 내부 코드에는 문제점이 보이지 않아 인터럽트가 들어 올 때, 처리 시간에 문제점이 있지 않을까 생각하였음.

- **IRQ** 방식 대신에 **FIQ**를 사용하여 문제를 해결.

- 더 정확한 이유를 파악해야 할 필요성이 있음.

HET (HIGH END TIMER) - 정한별

목차

1. 기본설명
2. led 켜고 끄는 기본 예제
    a. **1) HALCoGen**
    b. **2) CCS**

HET 이란? (High End Timer)
고성능의 타이머.

타이머는 축소 된 명령어 세트를 작동시키는 특수 마이크로 기계로 구성됩니다.

두 개의 **25** 비트 레지스터 및 **3** 개의 **32** 비트 레지스터를 사용하여 시간, 이벤트 카운트, 각도 값을 표현한다.

시스템 성능은 넓은 명령어 형식 **(96** 비트**)**으로 향상되어 **N2HET**을 사용하여 한 시스템 사이클에서 명령 연산 코드와 데이터를 가져와 속도를 높입니다.

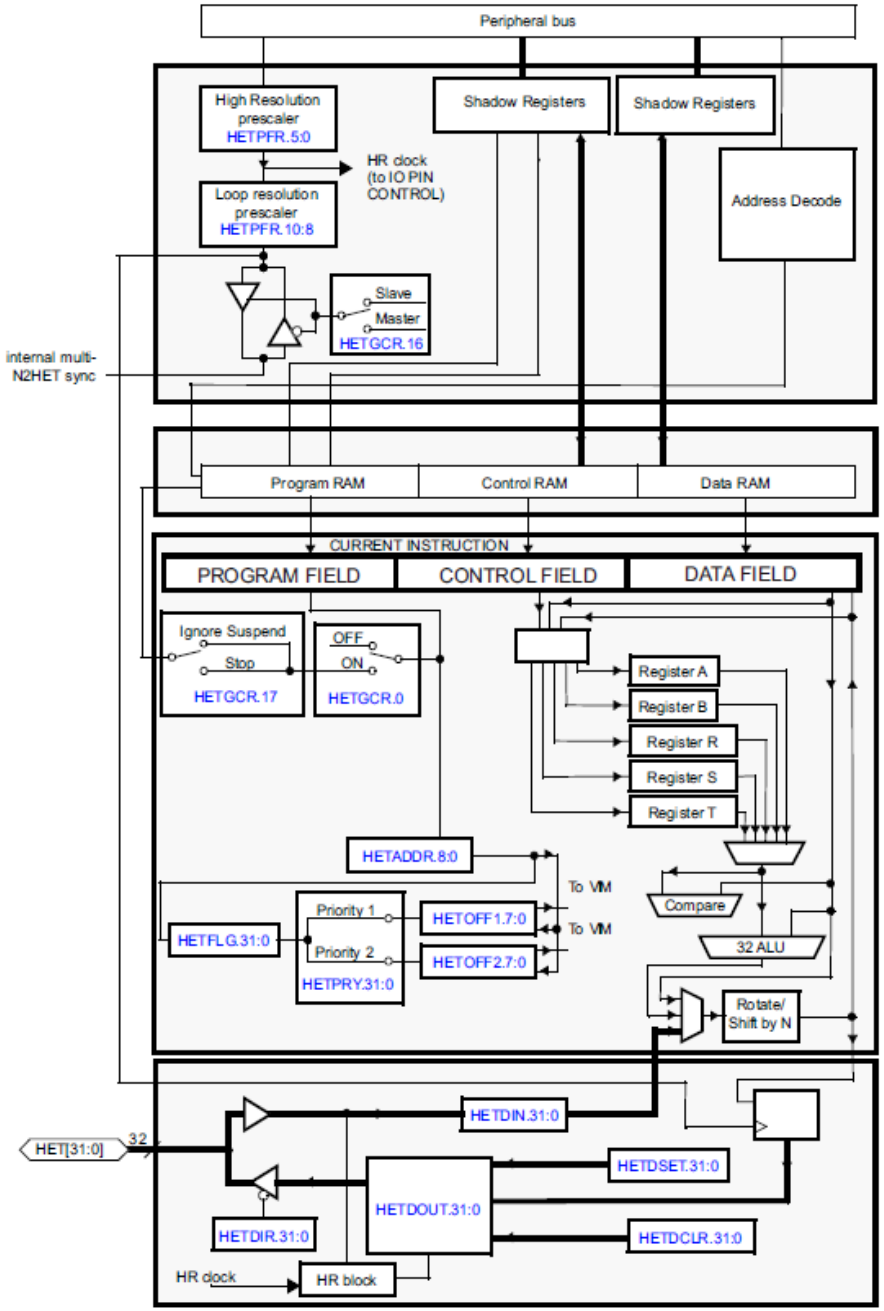
기능 설명

1. **HET PORT**만 따로 **(** 주파수 분주**)** 설정이 가능하다**. - duty ,** 등 주파수 관련한 일부 동작이 가능하다**.**
2. **PWM** 설정이 가능하다**.**
3. **VCLK2** 를 사용한다**.**
4. **VCLK2**의 파형이 결정 되면 **capture mode** 등으로 파형의 상태를 **check**가 가능한데 이것으로 동작 상태를 조절 할 수 있다**.**
5. **VCLK2**의 파형을 이용해서 **Edge Low , Edge High** 일 때**,** 인터럽트를 동작하게 하는 모드 등이 있다**.**
6. **pwm interrupts** 를 이용해서 사용이 가능하다**.**
7. **N2HET**은 소프트웨어로 제어됩니다**.**
8. 입력 및 출력 타이밍 기능을 위한 프로그래밍 가능한 타이머
9. 각 핀에 대한 **7** 비트 하드웨어 카운터는 **25** 비트 가상 카운터와 함께 최대 **32** 비트 분해능을 허용합니다
10. 타이머**,** 이벤트 카운터 및 앵글 카운터를 위한 **25** 비트 가상 카운터의 사용자 정의 구성
11. 전용 하이 엔드 타이머 전송 장치 **(HTU)** 또는 **DMA**를 사용하여 **CPU** 메모리와의 효율적인 데이터 전송

전용 타이머 마이크로 머신과 **30 가지 명령어 세트가있는 타이머**. (ADD, ADC, SUB, SBB, AND, OR, XOR, RCNT** 등**)**
N2HET 마이크로 머신은 최대 **32** 개의 입 **/** 출력 **(I / O)** 핀 포트에 연결됩니다**.**

12. 원래는 **HET -> NHET -> N2HET** 으로 발전했다**.** 그래서 고고고 성능 **timer** 이다**.**

Figure 23-1. N2HET Block Diagram



23.1.6 N2HET Compared to NHET

N2HET enhancements from NHET include:

- Eight new instructions: ADD, ADC, SUB, SBB, AND, OR, XOR, RCNT
- Full set of ALU flags Carry (C), Negative (N), Zero (Z), Overflow (V)
- Branch instruction (BR) extended to support signed and unsigned arithmetic comparison conditions
- Two additional 32-bit temporary working registers R, S.
- New HETAND register for AND-Sharing of High Resolution structure between pairs of pins
- Improved high resolution PCNT instruction

23.1.7 NHET and N2HET Compared to HET

Compared to the HET module, the N2HET contains all of the enhancements described in Section 23.1.6 plus the following additional enhancements:

- New Interrupt Enable Set and Clear registers
- Capability to generate requests to the DMA module or the HET Transfer Unit (HTU) including new Request Enable Set and Clear registers
- N2HET RAM parity error detection
- Suppression filters for each of the 32 I/O channel and control register to configure the limiting frequency and counter clock
- Enhanced edge detection hardware that does not rely on the previous bit field in the control word of the N2HET instruction.
- The next, conditional and remote addresses are extended from 8 to 9 bits
- The loop resolution data fields are extended from 20 to 25 bits
- The high resolution data fields are extended from 5 to 7 bits
- Instructions with an adequate condition are able to specify the number of the request line, which triggers either the HET Transfer Unit (HTU) or the DMA module
- The CNT instruction provides a bit, which allows to configure either an equal comparison or a greater or equal comparison when comparing the selected register value with the Max-value
- The MOV32 instruction provides a new bit. If set to one the MOV32 will only perform the move, when the Z-flag is set. If set to zero the MOV32 will perform the move whenever it is executed (independent on the state of the Z-flag)
- There is a new instruction WCAPE, which is a combination of a time stamp and an edge counter
- New Open Drain, Pull Disable, and Pull Select registers

HET에는 특수 마이크로 머신 이라는 명령어 체계가 존재 한다.

RAM영역 에 명령어들이 존재 하고 있다.

크게 **program** 영역, **control** 영역 , **data** 영역이 있다.

마이크로 머신이 이 세가지 영역에 접근해서 레지스터들을 건드려 어떻게 동작할지 설정해준다.

Figure 23-2. Specialized Timer Micromachine

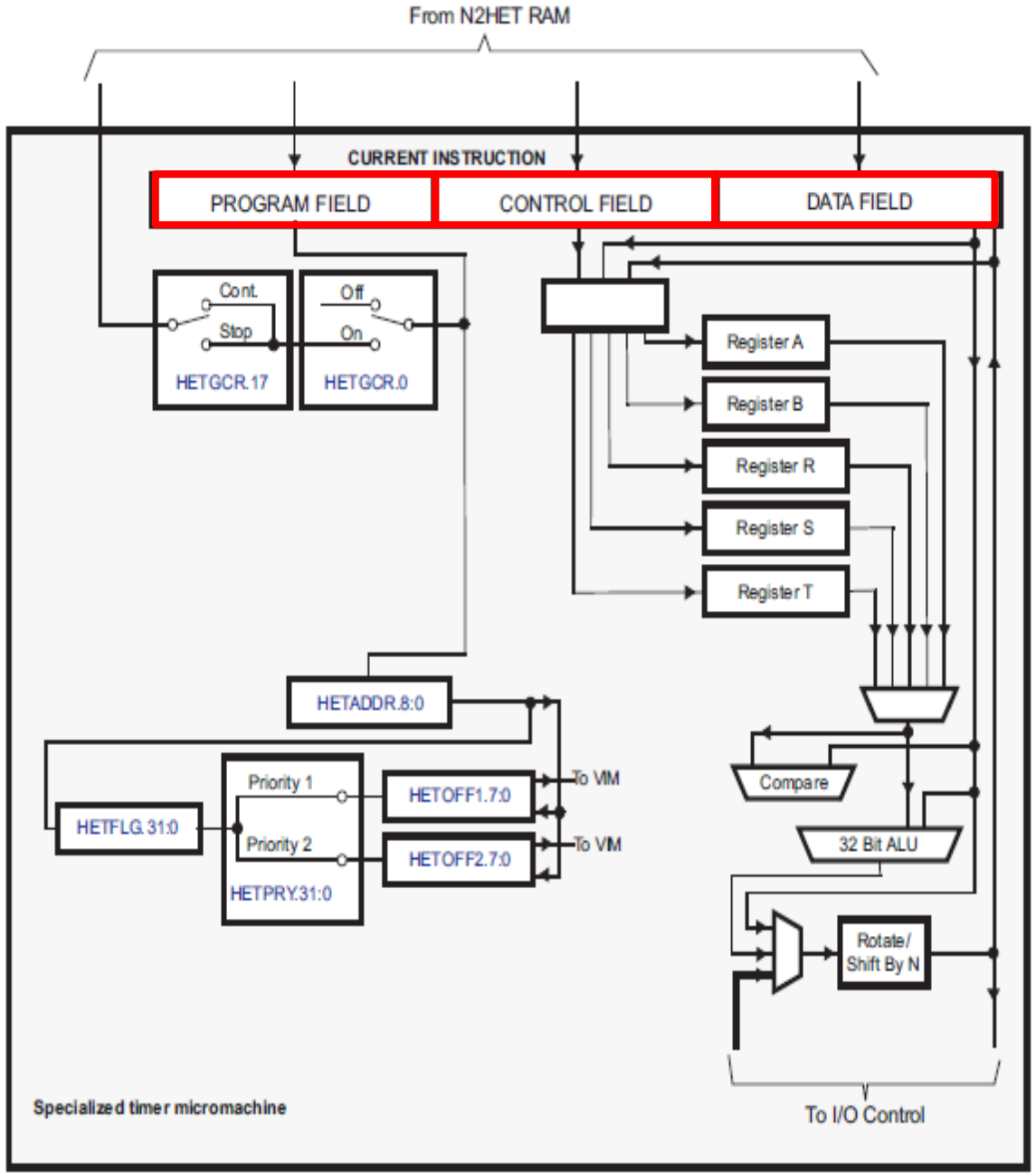
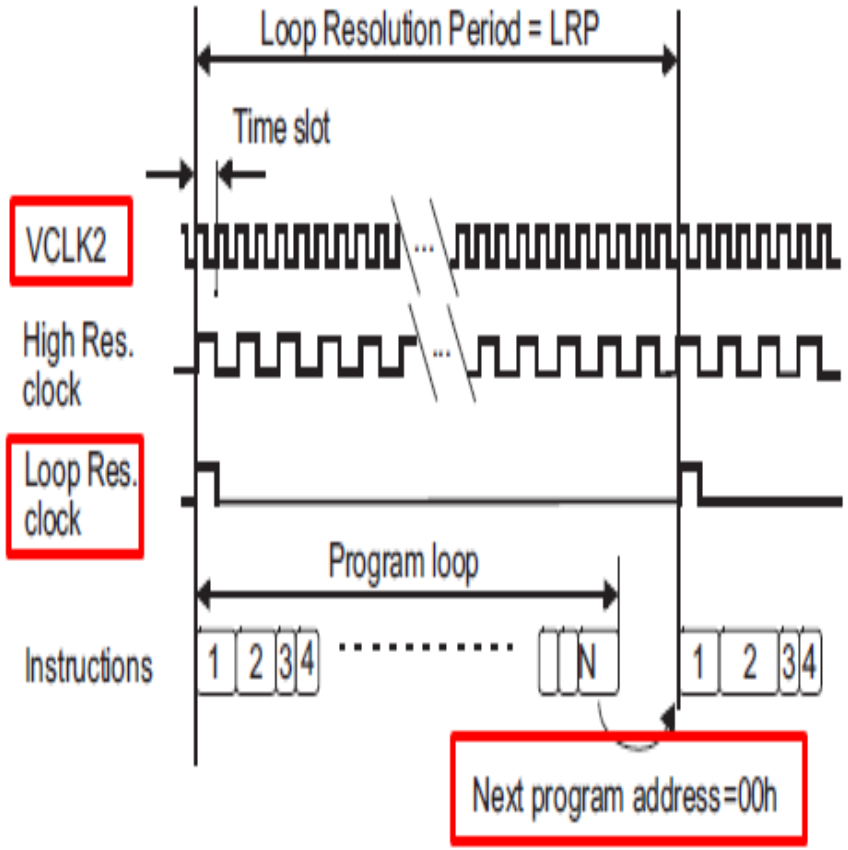


Figure 23-3. Program Flow Timings



HET을 가지는 RAM에는 2가지 SECTION이 있다.

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 23-4. N2HET Parity Bit Mapping

Address N2HET1	Address N2HET2	Bits	
		[31:1]	[0]
0xFF46_2000	0xFF44_2000	Reads 0, Writes have no effect	Instruction 0 Program Field Parity Bit
0xFF46_2004	0xFF44_2004	Reads 0, Writes have no effect	Instruction 0 Control Field Parity Bit
0xFF46_2008	0xFF44_2008	Reads 0, Writes have no effect	Instruction 0 Data Field Parity Bit
0xFF46_200C	0xFF44_200C	Reads 0, Writes have no effect	Read 0
0xFF46_2010	0xFF44_2010	Reads 0, Writes have no effect	Instruction 1 Program Field Parity Bit
....	....	...	...

1. 프로그램에 대한 **section**

2. **parity protection bits**들을 프로그램밍 하기 위한 **section**

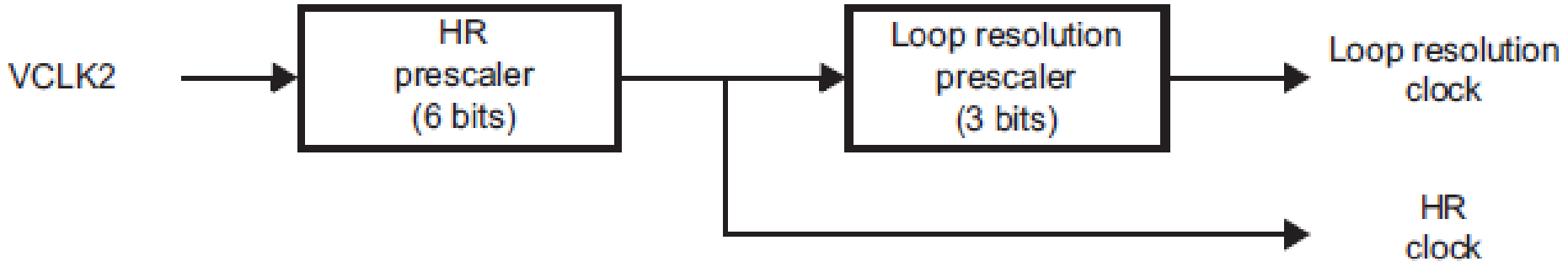
TIME BASE

N2HET은 VCLK2 의 분주를 base로 돌아 가도록 되어 있다.

Prescalers
1.high resolution (HR) clock - HR I/O counters 를 위한, (일반적으로 on 되어 있다.)
- 6bit 분주기로 이루어진 HR prescale divide rate에 VCLK2 에 따라 저장 되어 있다.

2. timer loop resolution clock - program loop 를 위한,
- 3bit 분주기로 이루어진 loop-resolution divide rate는 HR prescaler 에 따라 저장 되어 있다.

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)

I/O CONTROL

1) HET_DIN →(1) lr(Loop Resolution Clock) (선택 가능) → timer data in
→(2)HET_DOUT → HR(High Resolution) → lr(Loop Resolution Clock) (선택 가능) → timer data in

2) HET_DIR ← HET_DOUT ← HET_DSET
← HET_DCLR
← Timer data out

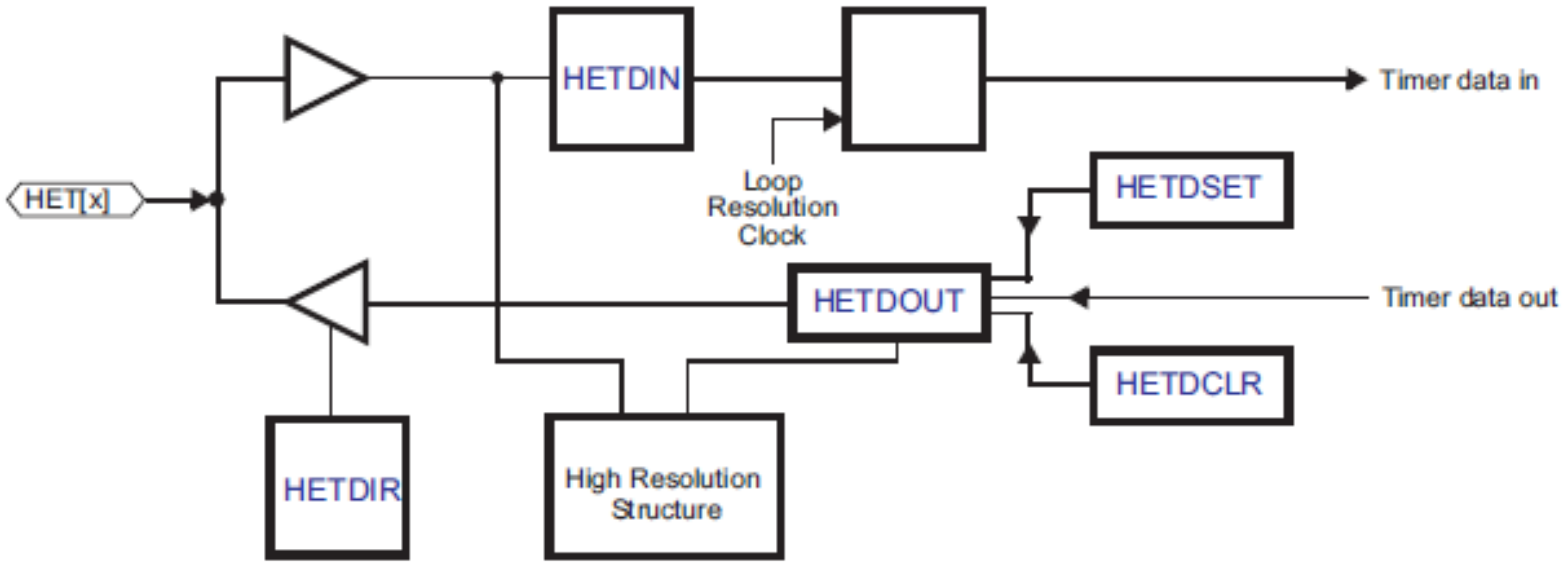
위와 같은 형식으로 **I/O PORT**를 어떻게 쓸지 설정을 할 수 있다.

23.2.5 I/O Control

The N2HET has up to 32 pins. Refer to device specific data sheets for information concerning the number of N2HETIO available. All of the N2HET pins available are programmable as either inputs or outputs.

These 32 I/Os have an identical structure connected to pins HET[31] to HET[0]. See Figure 23-8 for an illustration of the I/O control. In addition all 32 I/Os have a special HR structure based on the HR clock. This structure allows any N2HET instruction to use any of these I/Os with an accuracy of either loop resolution or high resolution accuracy.

Figure 23-8. I/O Control



Pins N2HET [31] to N2HET [0] can be used by the CPU as general-purpose inputs or outputs using the N2HET Data Input Register (HETDIN) for reading and N2HET Data Output Register (HETDOUT), N2HET Data Set Register (HETDSET) or N2HET Data Clear Register (HETDCLR) for writing, depending on the type of action to perform. The N2HET pins used as general-purpose inputs are sampled on each VCLK2 period.

LOOP RESOLUTION 구조, BLOCK DIAGRAM
(상승엣지와 관련한 프로그램의 동작을 관리하는 녀석)

- **N2HET**은 명령 세트의 방법에 따라 입력 및 / 또는 출력으로서 **N2HET [31 : 0]** 핀을 사용합니다.
- 실제적으로는 각 **pin**은 **N2HET** 프로그램을 모니터링하거나 **N2HET** 프로그램으로 모니터링 할 수 있습니다.

- **N2HET**의 **I / O** 레지스터에서 **CPU**는 **N2HET** 프로그램 흐름과 상호 작용할 수 있습니다.
- **N2HET** 프로그램에 의해 동작 (세트 또는 리셋)이 핀에서 취해지면 **N2HET**은 다음 **Loop Resolution clock** 의 상승 엣지에 따라 수정될 것이다.
- **N2HET I / O** 핀에서 이벤트가 발생하면 루프의 다음 상승 에지에서 이벤트가 고려됩니다 .

HR(HIGH RESOLUTION)구조, BLOCK DIAGRAM
(HR클럭 기반으로 입력 캡쳐/출력 비교 명령을들을 관리)

- **HR** 구조체는 각 핀당 한개씩 들어 있다.
- **HR** 을 사용 하게 되면 **HETPFR** 에서 **clock** 주파수 분주할 수 있다.
- 기본 **I/O pin** 들은 **PCNT, WCAP, HR outputcompare(ECMP,MCMP ,PWCNT** 명령어를 사용) 들을 사용한다.
- 위의 **5**개의 지시어(**PCNT, WCAP,ECMP, MCMP, PWCNT**) 는 **hr_lr bit**의 영향을 받는다. **(1** 또는 **0)**

mode (hr_lr bit를 설정)
- **HR mode(bit cleared to 0) : HR** 동작 모드 이다. 동작중에는 하나의 명령어만 수행 된다.
- **Standard mode(1) : HR** 명령어 모드 이다. 한 핀당 하나의 명령어만 가능하다.

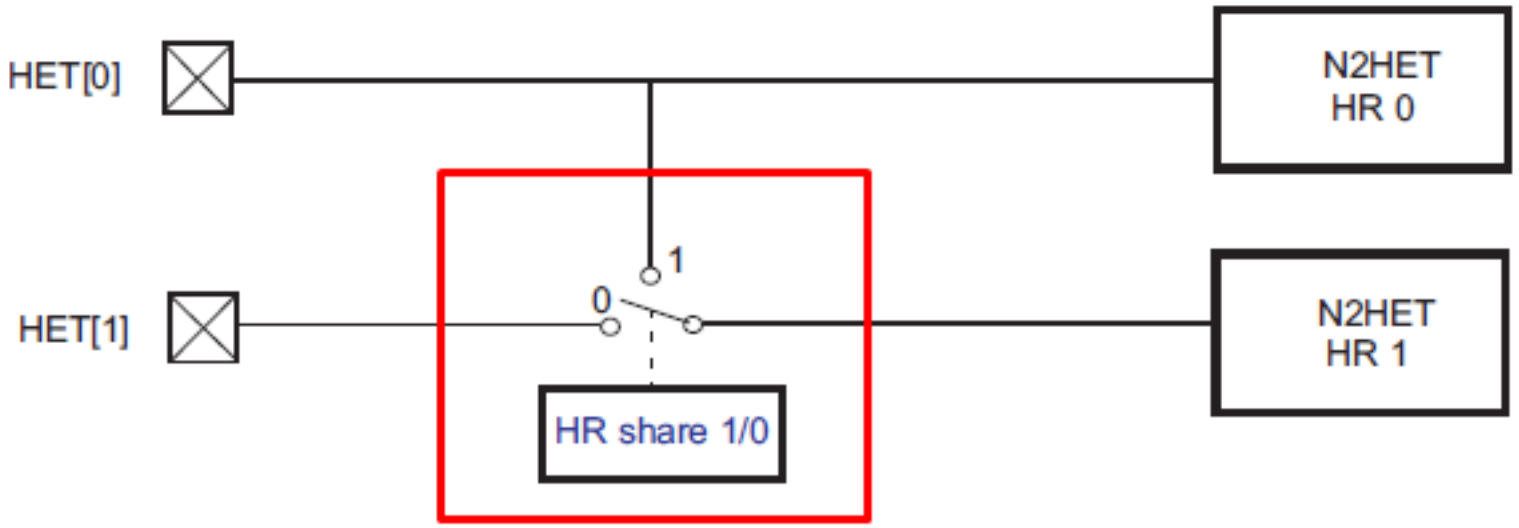
HR SHARING (INPUT) - (입력의 값을 공유해 CAPTURE용으로 사용할 때)

- **input** 의 경우에서 활용 할 수 있다.
- **HR Share Control Register (HETHRSH)**를 사용하면 두 **HR** 구조가 입력 캡처용으로 만 동일한 핀을 공유 할 수 있습니다.
- **HET[0]**의 입력 값을 받아 **N2HET HR0**와 **N2HET HR1** 로 출력 한다.

23.2.5.5 HR Structures Sharing (Input)

The HR Share Control Register (HETHRSH) allows two HR structures to share the same pin **for input capture only**. If these bits are set, the HR structures N and N+1 are connected to pin N. In this structure, pin N+1 remains available for general-purpose input/output. See Figure 23-12.

Figure 23-12. Example of HR Structure Sharing for N2HET Pins 0/1



The following program gives an example how the HR share feature (HET[0] HR structure and HET[1] HR structure shared) can be used for the PCNT instruction:

```
L00 PCNT { next=L01, type=rise2fall, pin=0 }
L01 PCNT { next=L00, type=fall2rise, pin=1 }
```

The HET[1] HR structure is also connected to the HET[0] pin. The L00 PCNT data field is able to capture a high pulse and the L01 PCNT captures a low pulse on the **same** pin (N2HET [0] pin).

LOOP BACK MODE (출력에 대한 모니터링을 할 때)

-N2HET 출력 신호를 모니터링하기 위해 응용 프로그램에서 루프백 기능을 사용할 수 있습니다.
예를 들어, **if PWM**은 **HR** 구조 **0**에 의해 생성되고, **HR** 구조 **1**에 할당 된 **PCNT** 명령은**PWM** 출력 신호의 펄스 길이를 되감습니다.

- Loopback mode는 **2**개의 **High Resolution structures**에 의해서 **setting** 할 수 있다.

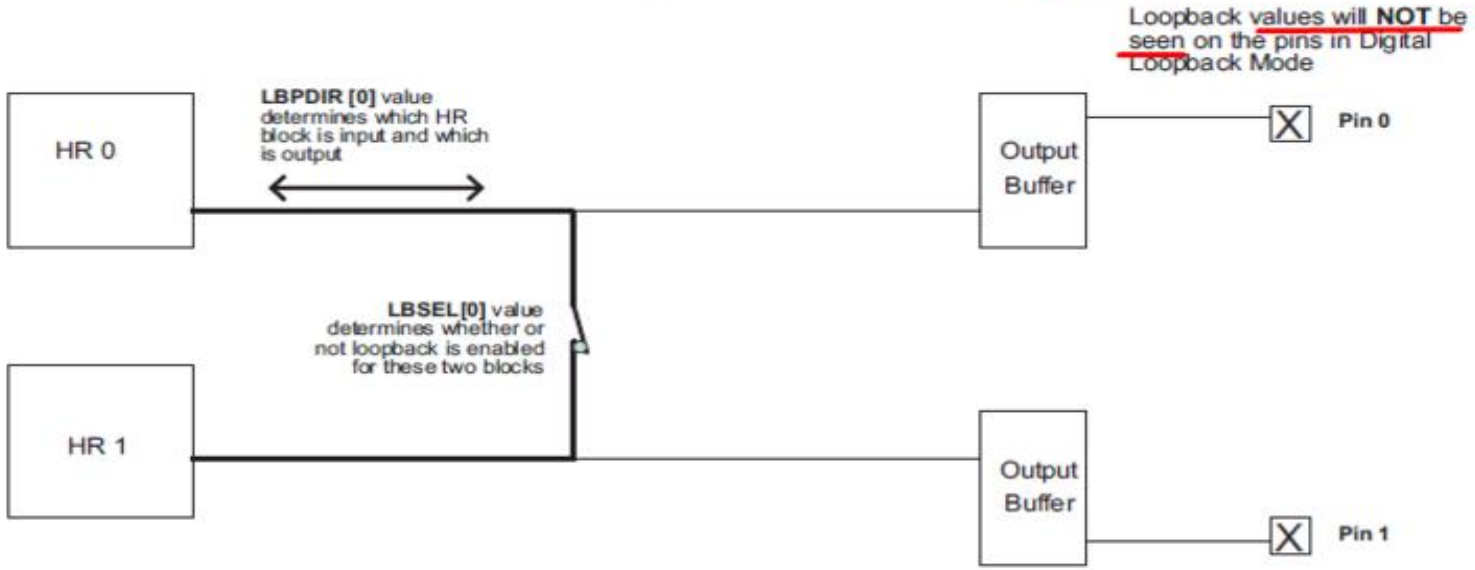
mode
1. Digital Loopback mode : LBPTYPE[x] 가 **0** 일 때 값을 안보이게 한다.
2. Analog Loopback mode : LBPTYPE[x] 가 **1** 일 때 값을 보이게 한다.

1. DIGITAL LOOPBACK MODE

Digital Loopback

Digital loopback mode is enabled by setting LBPTYPE[x] to 0 in the HETLBPSEL register for the corresponding structure pairs. In digital loopback mode, the structure pairs are connected directly and the output buffers are bypassed. Therefore, the loopback values will NOT be seen on the corresponding pins. Figure 23-16 shows an example of digital loopback between structures HR0 and HR1. LBSEL[0] has been set to 1 to enable loopback between the two structures. LBTYPE[0] has been set to 0 to select digital mode for the loopback pair. The LPBDIR[0] value will determine the direction of the loopback by selecting which of the HR blocks is output, and which is input. The bold lines show the digital loopback path.

Figure 23-16. HR0 to HR1 Digital Loopback Logic: LBTYPE[0] = 0

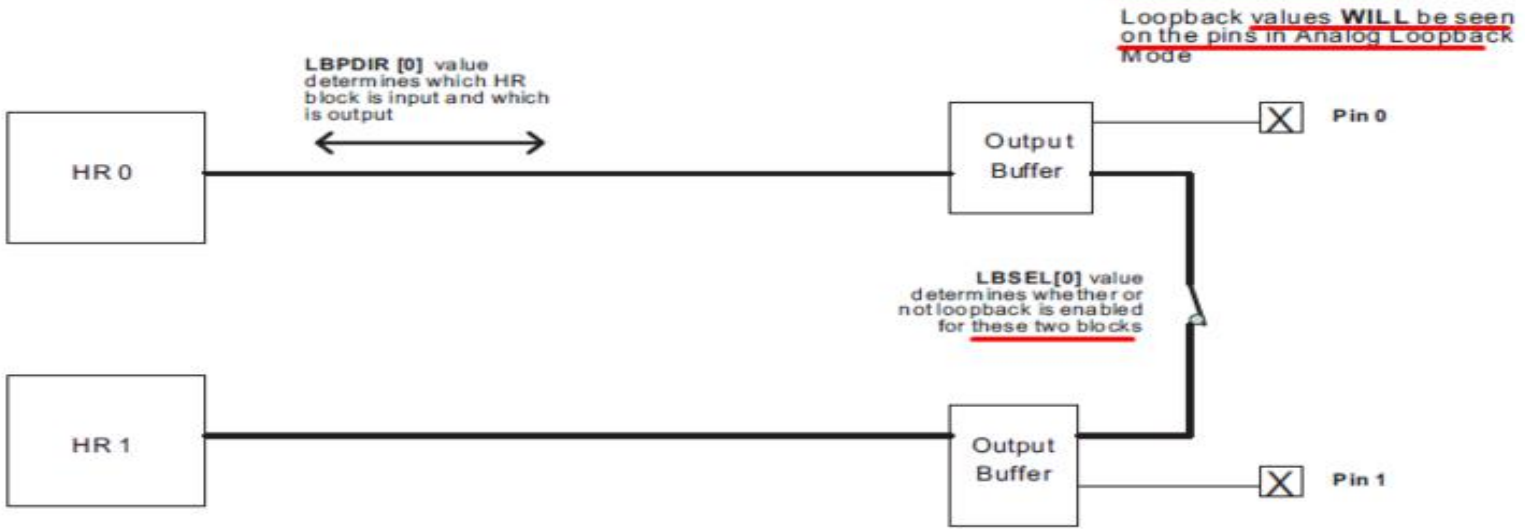


2. ANALOG LOOPBACK MODE

Analog Loopback

Analog loopback mode is enabled by setting LBPTYPE[x] to 1 in the HETLBPSEL register for the corresponding structure pairs. In analog loopback mode, the structure pairs are connected outside of the output buffers. Therefore, the loopback values WILL be seen on the corresponding pins. Figure 23-17 shows an example of analog loopback between structures HR0 and HR1. LBSEL[0] has been set to 1 to enable loopback between the two structures. LBTYPE[0] has been set to 1 to select analog mode for the loopback pair. The LPBDIR[0] value will determine the direction of the loopback by selecting which of the HR blocks is output, and which is input. The bold lines show the analog loopback path.

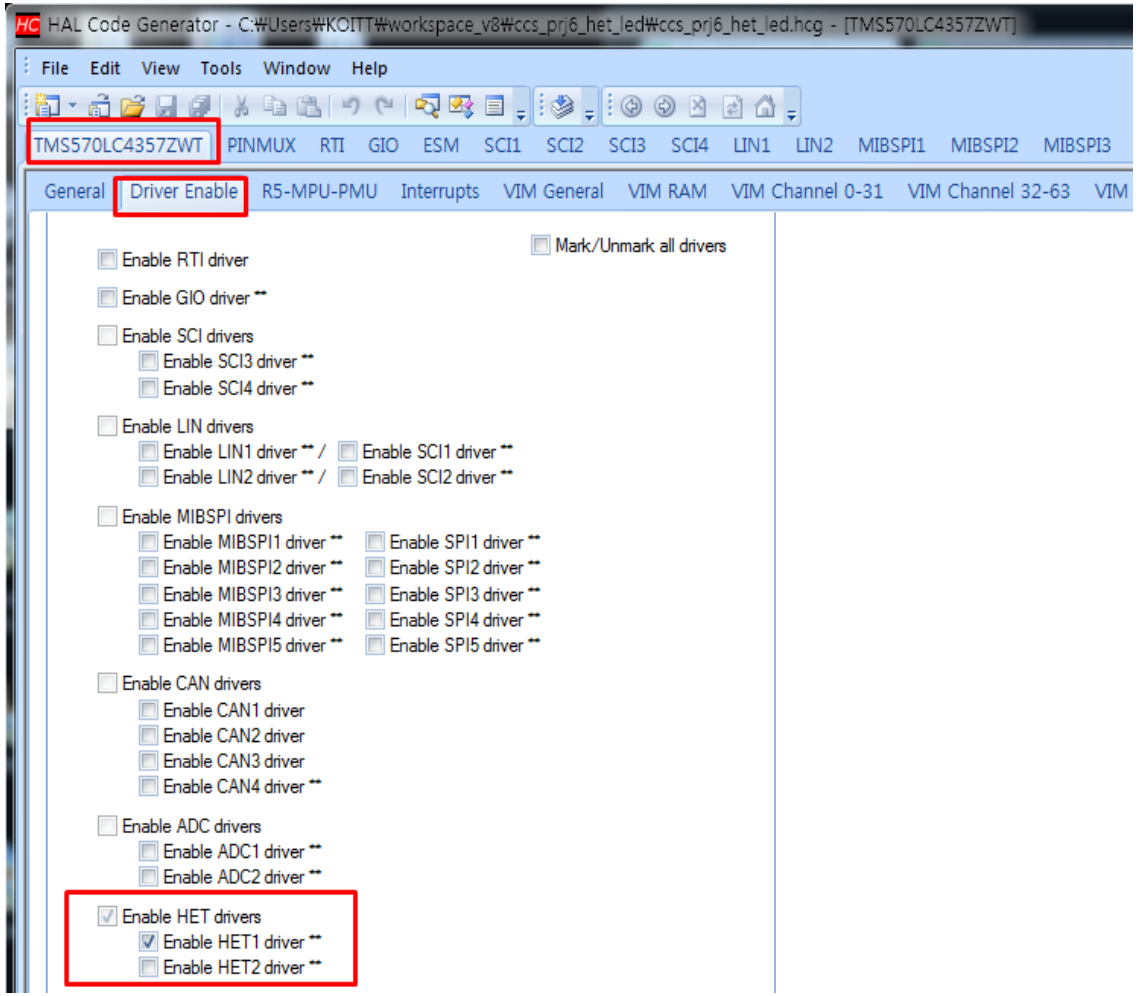
Figure 23-17. HR0 to HR1 Analog Loop Back Logic: LBTYPE[0] = 1

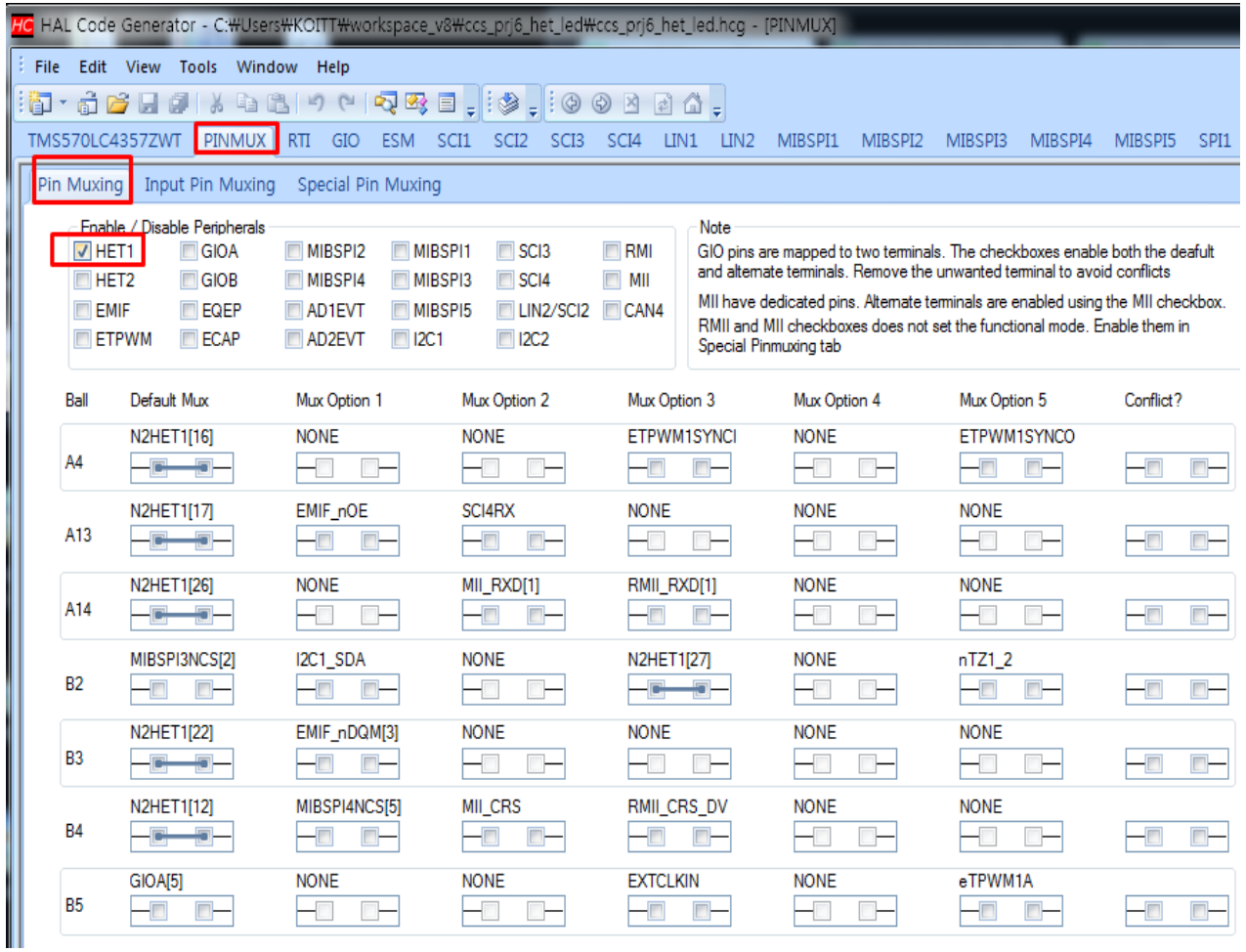


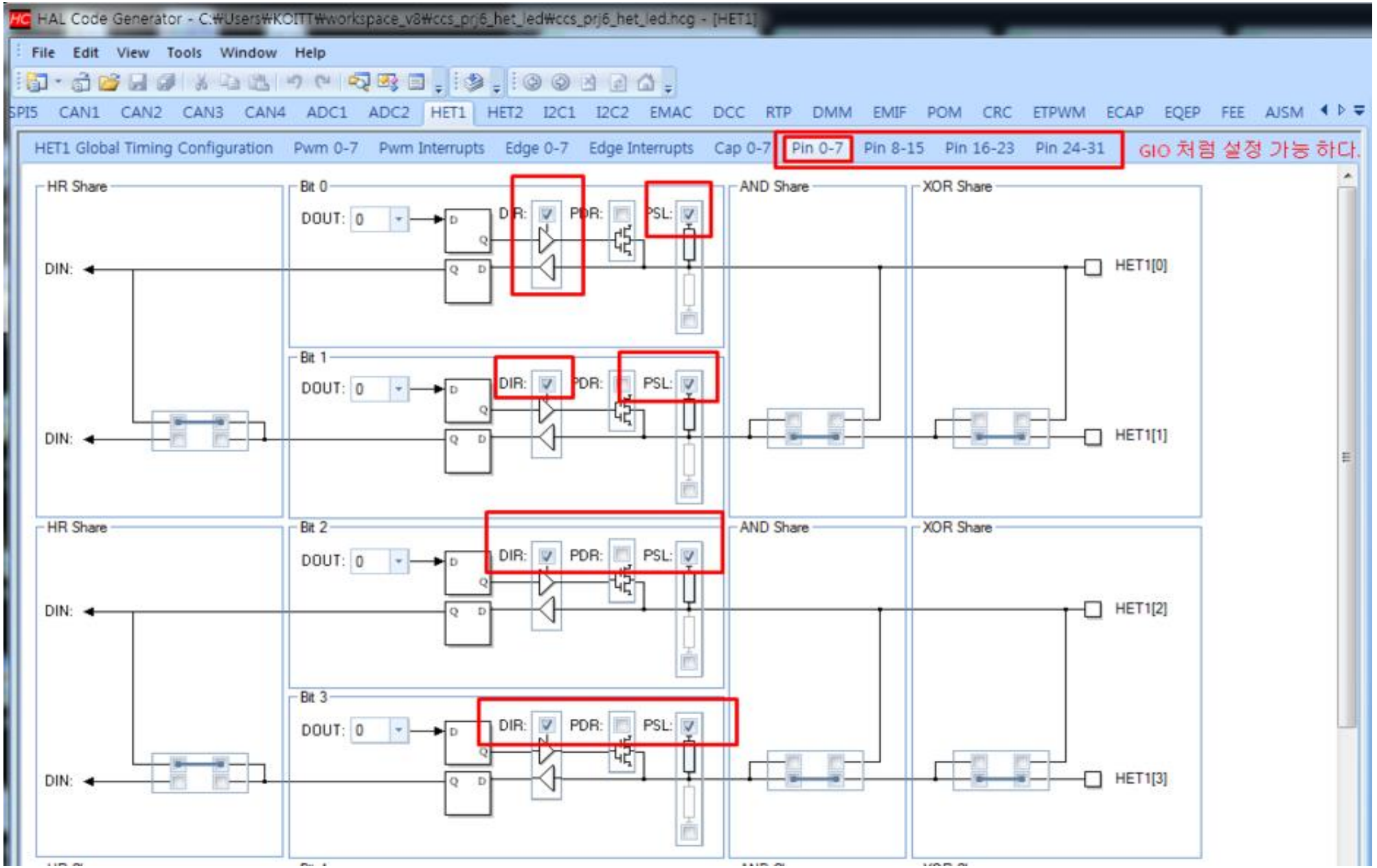
Note:

- The loop back direction can be selected independent of the HETDIR register setting.
- The pin that is not driven by the N2HET output pin actions can still be used as normal GIO pin.

HALCOGEN 설정하기






HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj6_het_led₩ccs_prj6_het_led.hcg - [HET1]
File Edit View Tools Window Help
SPI5 CAN1 CAN2 CAN3 CAN4 ADC1 ADC2 HET1 HET2 I2C1 I2C2 EMAC DCC RTP DMM EMIF POM CRC ETPWM ECAP EQEP FEE AJSM
HET1 Global Timing Configuration
Pwm 0-7
Pwm Interrupts
Edge 0-7
Edge Interrupts
Cap 0-7
Pin 0-7
Pin 8-15
Pin 16-23
Pin 24-31
GIO 처럼 설정 가능 하다.
HR Share
Bit 0
DOUT: 0
DIR:
PDR:
PSL:
AND Share
XOR Share
DIN:
HET1[0]
Bit 1
HET1[1]
Bit 2
HET1[2]
Bit 3
HET1[3]

CCS

- **hetREG(het** 레지스터)에 직접 접근한 구현.
- 여기서 알아야 할 점은 **gio**구조 형식이 같다고 **gio**에 있던 함수를 쓰려고 하면 안되어 있다. 쓰려고 한다면 **gio.c** 파일에서 형식만 가져오고 타입을 바꾸주면 쉽게 쓸 수 있다.

```
#INCLUDE <INCLUDE/HL_HAL_STDTYPES.H>
#INCLUDE <INCLUDE/HL_HET.H>
#INCLUDE <INCLUDE/HL_REG_HET.H>


/* USER CODE BEGIN (2) */
VOID WAIT(UINT32 TIME)
{
    INT I =0;
    WHILE(I++ < TIME)
        ;
}
/* USER CODE END */

INT MAIN(VOID)
{
/* USER CODE BEGIN (3) */

   HETINIT();
   // HET1 레지스터의 출력을  값을설정해 주는 것이다.
   HETREG1->DOUT = 0X0000000F;
   // HET1 레지스터의  입출력 방향을 설정해 준다.
   HETREG1->DIR = 0X0000000F;

   WHILE(1)
   {
      HETREG1->DOUT = HETREG1->DOUT ^ 0X0000000F;
      WAIT(1000000);
   }

/* USER CODE END */

    RETURN 0;
}
```

GIO RTI 분석 - 정상용

첫번째 주 : 7.4(화) ~ 7.10(수)

MCU Peripheral(GIO, RTI) Interface Functions와 각 register 분석.

GIO INTERFACE FUNCTIONS

```
153 /* GIO Interface Functions */
154 void gioInit(void);
155 void gioSetDirection(gioPORT_t *port, uint32 dir);
156 void gioSetBit(gioPORT_t *port, uint32 bit, uint32 value);
157 void gioSetPort(gioPORT_t *port, uint32 value);
158 uint32 gioGetBit(gioPORT_t *port, uint32 bit);
159 uint32 gioGetPort(gioPORT_t *port);
160 void gioToggleBit(gioPORT_t *port, uint32 bit);
161 void gioEnableNotification(gioPORT_t *port, uint32 bit);
162 void gioDisableNotification(gioPORT_t *port, uint32 bit);
163 void gioNotification(gioPORT_t *port, uint32 bit);
164 void gioGetConfigValue(gio_config_reg_t *config_reg, config_value_type_t type);
```

RTI INTERFACE FUNCTIONS

```
280 /* RTI Interface Functions */
281
282 void rtiInit(void);
283 void rtiStartCounter(rtiBASE_t *rtiREG,uint32 counter);
284 void rtiStopCounter(rtiBASE_t *rtiREG,uint32 counter);
285 uint32 rtiResetCounter(rtiBASE_t *rtiREG,uint32 counter);
286 void rtiSetPeriod(rtiBASE_t *rtiREG,uint32 compare, uint32 period);
287 uint32 rtiGetPeriod(rtiBASE_t *rtiREG,uint32 compare);
288 uint32 rtiGetCurrentTick(rtiBASE_t *rtiREG,uint32 compare);
289 void rtiEnableNotification(rtiBASE_t *rtiREG,uint32 notification);
290 void rtiDisableNotification(rtiBASE_t *rtiREG,uint32 notification);
291 void dwdInit(rtiBASE_t *rtiREG,uint16 dwdPreload);
292 void dwwdInit(rtiBASE_t *rtiREG,dwwdReaction_t Reaction, uint16 dwdPreload, dwwdWindowSize_t Window_Size);
293 uint32 dwwdGetCurrentDownCounter(rtiBASE_t *rtiREG);
294 void dwdCounterEnable(rtiBASE_t *rtiREG);
295 void dwdSetPreload(rtiBASE_t *rtiREG,uint16 dwdPreload);
296 void dwdReset(rtiBASE_t *rtiREG);
297 void dwdGenerateSysReset(rtiBASE_t *rtiREG);
298 boolean IsdwdKeySequenceCorrect(rtiBASE_t *rtiREG);
299 dwdResetStatus_t dwdGetStatus(rtiBASE_t *rtiREG);
300 dwdViolation_t dwdGetViolationStatus(rtiBASE_t *rtiREG);
301 void dwdClearFlag(rtiBASE_t *rtiREG);
302 void rtiGetConfigValue(rti_config_reg_t *config_reg, config_value_type_t type);
```