EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv2 과정

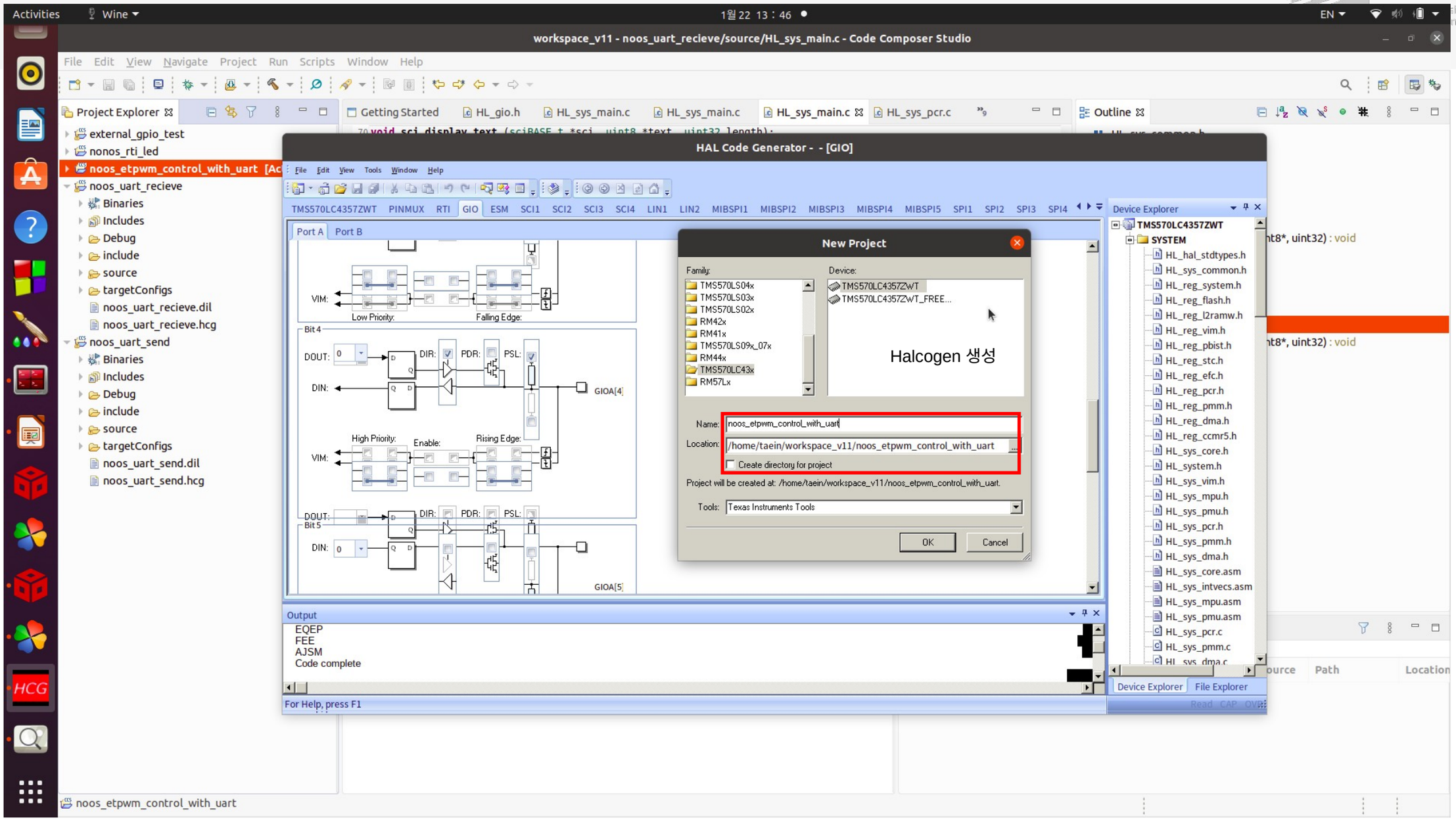
[TMS 570_etPWM(w/UART, 서보모터)]

제 1기

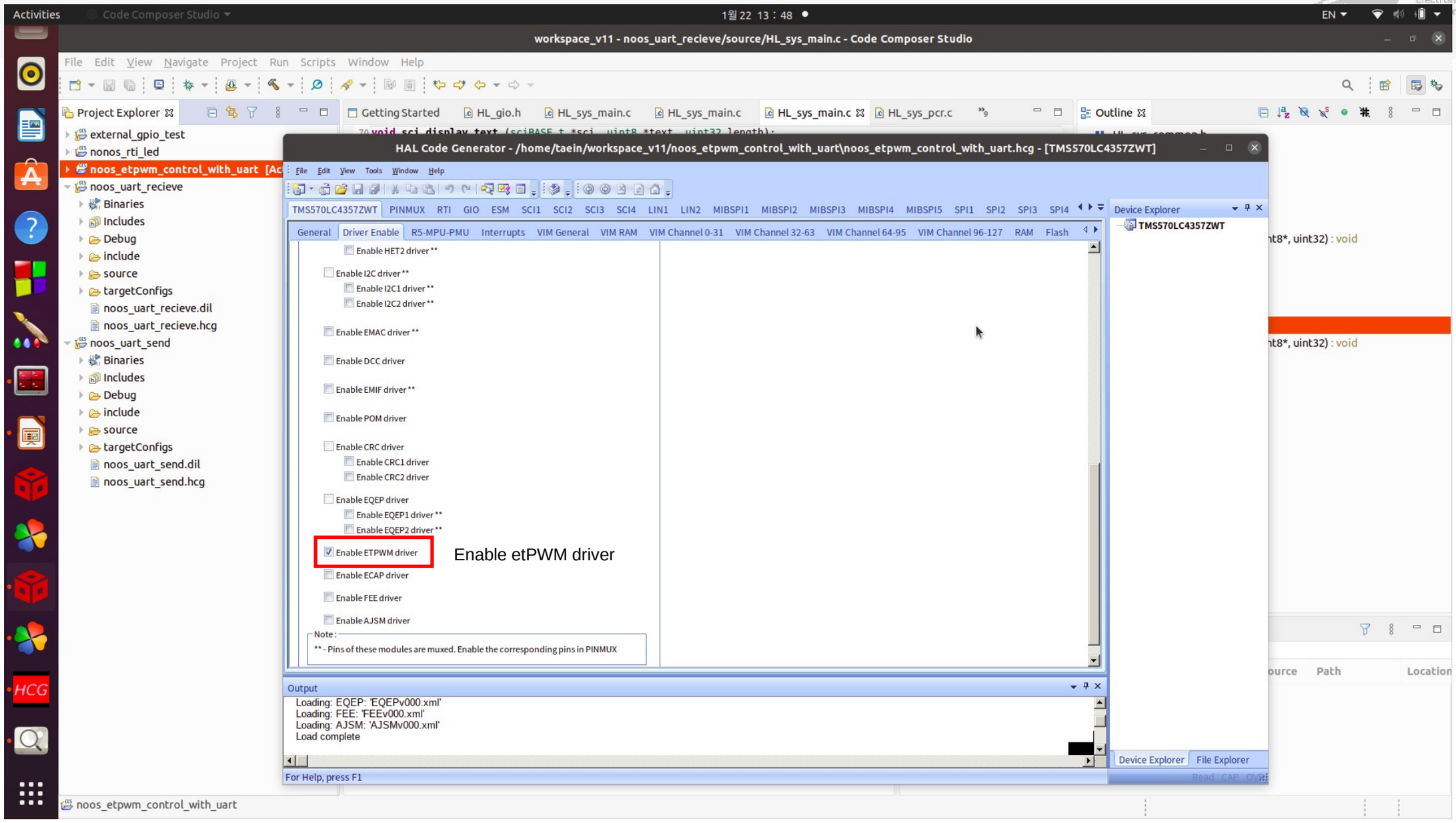
2022. 02. 25

박태인

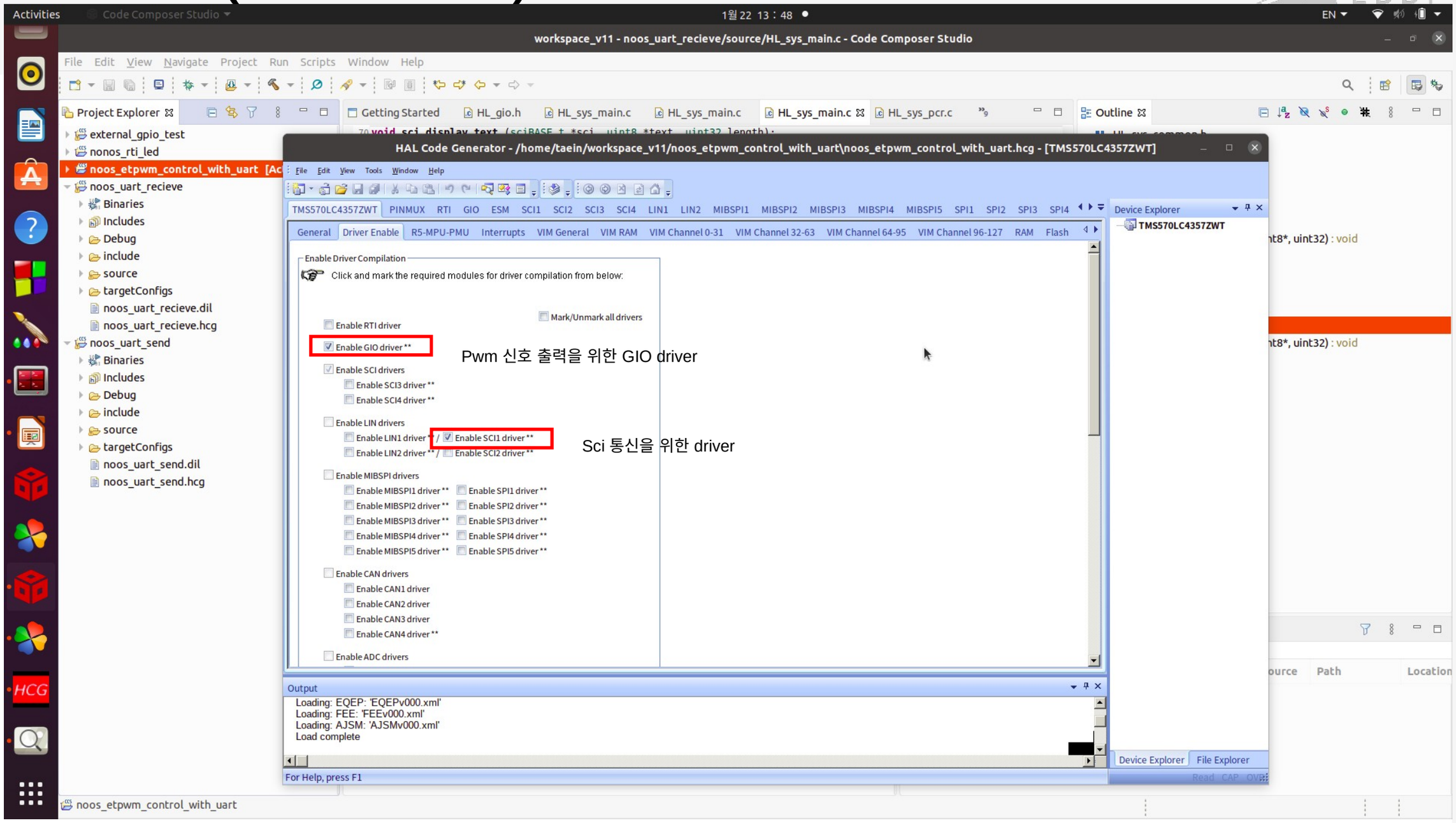
etPWM(w/UART)


Activities
Wine
1월 22 13 : 46
EN
workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Project Explorer
external_gpio_test
nonos_rti_led
noos_etpwm_control_with_uart [Ac
noos_uart_recieve
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_recieve.dil
noos_uart_recieve.hcg
noos_uart_send
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_send.dil
noos_uart_send.hcg
Getting Started
HL_gio.h
HL_sys_main.c
HL_sys_main.c
HL_sys_main.c
HL_sys_pcr.c
Outline
HAL Code Generator - - [GIO]
TMS570LC4357ZWT PINMUX RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1 SPI2 SPI3 SPI4
Port A
Port B
VIM:
Low Priority:
Falling Edge:
Bit 4
DOUT:
DIR:
PDR:
PSL:
DIN:
GIOA[4]
High Priority:
Enable:
Rising Edge:
Bit 5
GIOA[5]
New Project
Family:
TMS570LS04x
TMS570LS03x
TMS570LS02x
RM42x
RM41x
TMS570LS09x_07x
RM44x
TMS570LC43x
RM57Lx
Device:
TMS570LC4357ZWT
TMS570LC4357ZWT_FREE...
Halcogen 생성
Name:
noos_etpwm_control_with_uart
Location:
/home/taein/workspace_v11/noos_etpwm_control_with_uart
Create directory for project
Project will be created at: /home/taein/workspace_v11/noos_etpwm_control_with_uart.
Tools:
Texas Instruments Tools
OK
Cancel
Device Explorer
TMS570LC4357ZWT
SYSTEM
HL_hal_stdtypes.h
HL_sys_common.h
HL_reg_system.h
HL_reg_flash.h
HL_reg_l2ramw.h
HL_reg_vim.h
HL_reg_pbist.h
HL_reg_stc.h
HL_reg_efc.h
HL_reg_pcr.h
HL_reg_pmm.h
HL_reg_dma.h
HL_reg_ccmr5.h
HL_sys_core.h
HL_system.h
HL_sys_vim.h
HL_sys_mpu.h
HL_sys_pmu.h
HL_sys_pcr.h
HL_sys_pmm.h
HL_sys_dma.h
HL_sys_core.asm
HL_sys_intvecs.asm
HL_sys_mpu.asm
HL_sys_pmu.asm
HL_sys_pcr.c
HL_sys_pmm.c
HL_sys_dma.c
Output
EQEP
FEE
AJSM
Code complete
For Help, press F1
Device Explorer
File Explorer
Path
Location
noos_etpwm_control_with_uart

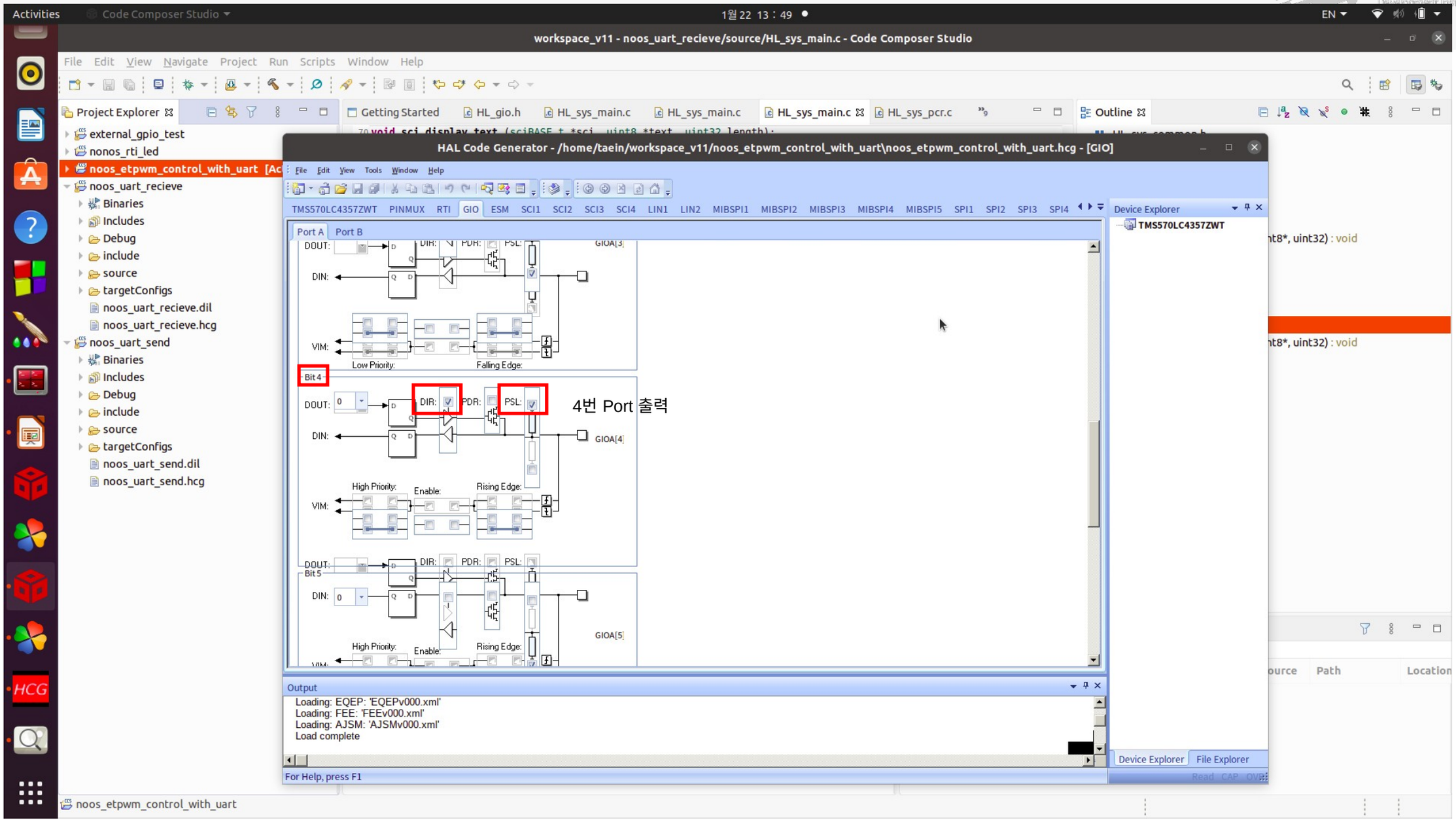
etPWM(w/UART)


Activities
Code Composer Studio
1월 22 13 : 48
EN
workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Project Explorer
external_gpio_test
nonos_rti_led
noos_etpwm_control_with_uart
noos_uart_recieve
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_recieve.dil
noos_uart_recieve.hcg
noos_uart_send
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_send.dil
noos_uart_send.hcg
Getting Started
HL_gio.h
HL_sys_main.c
HL_sys_main.c
HL_sys_main.c
HL_sys_pcr.c
Outline
HAL Code Generator - /home/taein/workspace_v11/noos_etpwm_control_with_uart\noos_etpwm_control_with_uart.hcg - [TMS570LC4357ZWT]
File Edit View Tools Window Help
TMS570LC4357ZWT PINMUX RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1 SPI2 SPI3 SPI4
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31 VIM Channel 32-63 VIM Channel 64-95 VIM Channel 96-127 RAM Flash
Enable HET2 driver **
Enable I2C driver **
Enable I2C1 driver **
Enable I2C2 driver **
Enable EMAC driver **
Enable DCC driver
Enable EMIF driver **
Enable POM driver
Enable CRC driver
Enable CRC1 driver
Enable CRC2 driver
Enable EQEP driver
Enable EQEP1 driver **
Enable EQEP2 driver **
Enable ETPWM driver
Enable ECAP driver
Enable FEE driver
Enable AJSM driver
Note :
** - Pins of these modules are muxed. Enable the corresponding pins in PINMUX
Enable etPWM driver
Device Explorer
TMS570LC4357ZWT
Output
Loading: EQEP: 'EQEPv000.xml'
Loading: FEE: 'FEEv000.xml'
Loading: AJSM: 'AJSMv000.xml'
Load complete
For Help, press F1
Device Explorer File Explorer
Path
Location
noos_etpwm_control_with_uart

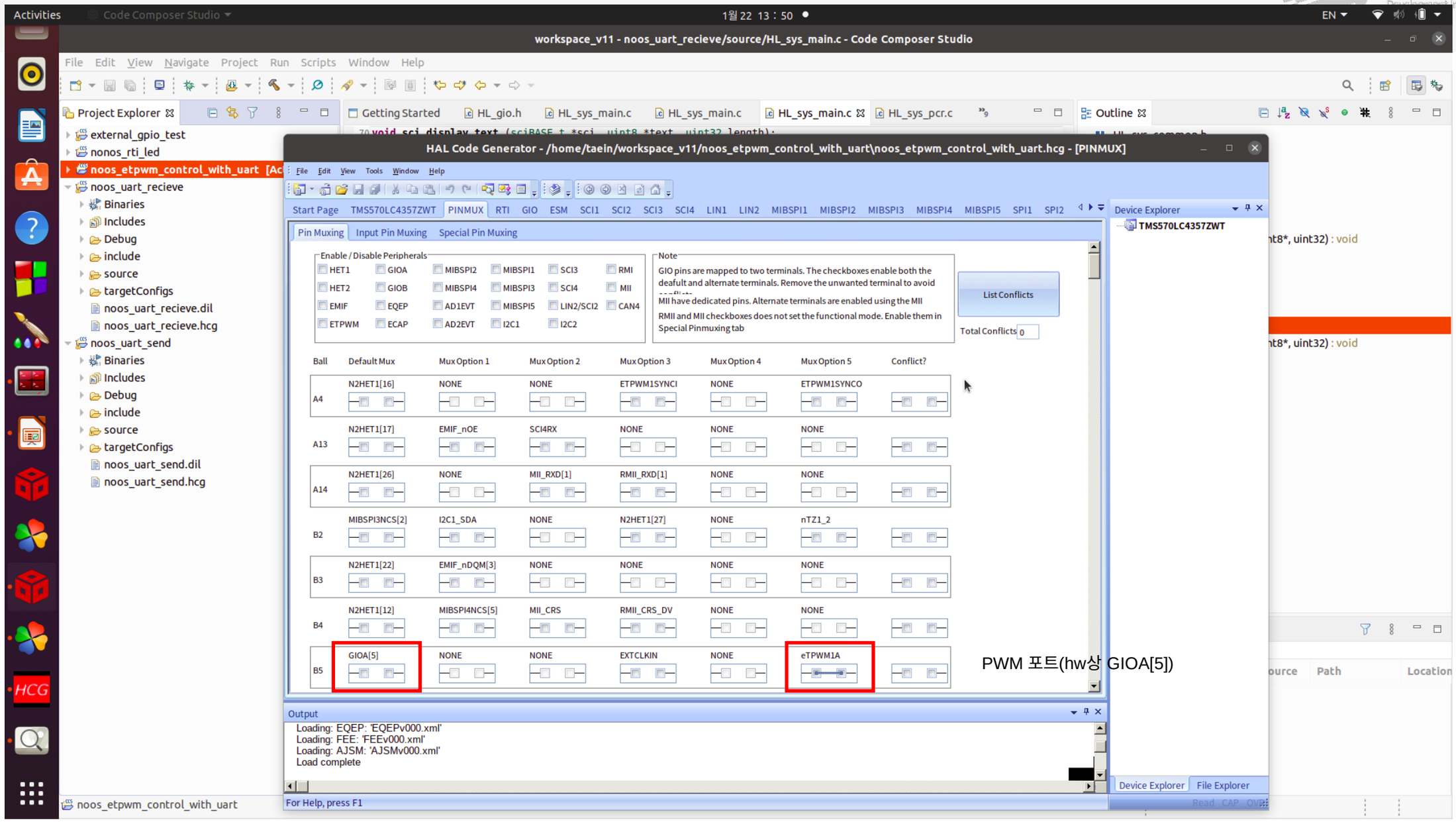
etPWM(w/UART)


workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
HAL Code Generator - /home/taein/workspace_v11/noos_etpwm_control_with_uart\noos_etpwm_control_with_uart.hcg - [TMS570LC4357ZWT]
Enable GIO driver **
Pwm 신호 출력을 위한 GIO driver
Enable SCI1 driver **
Sci 통신을 위한 driver

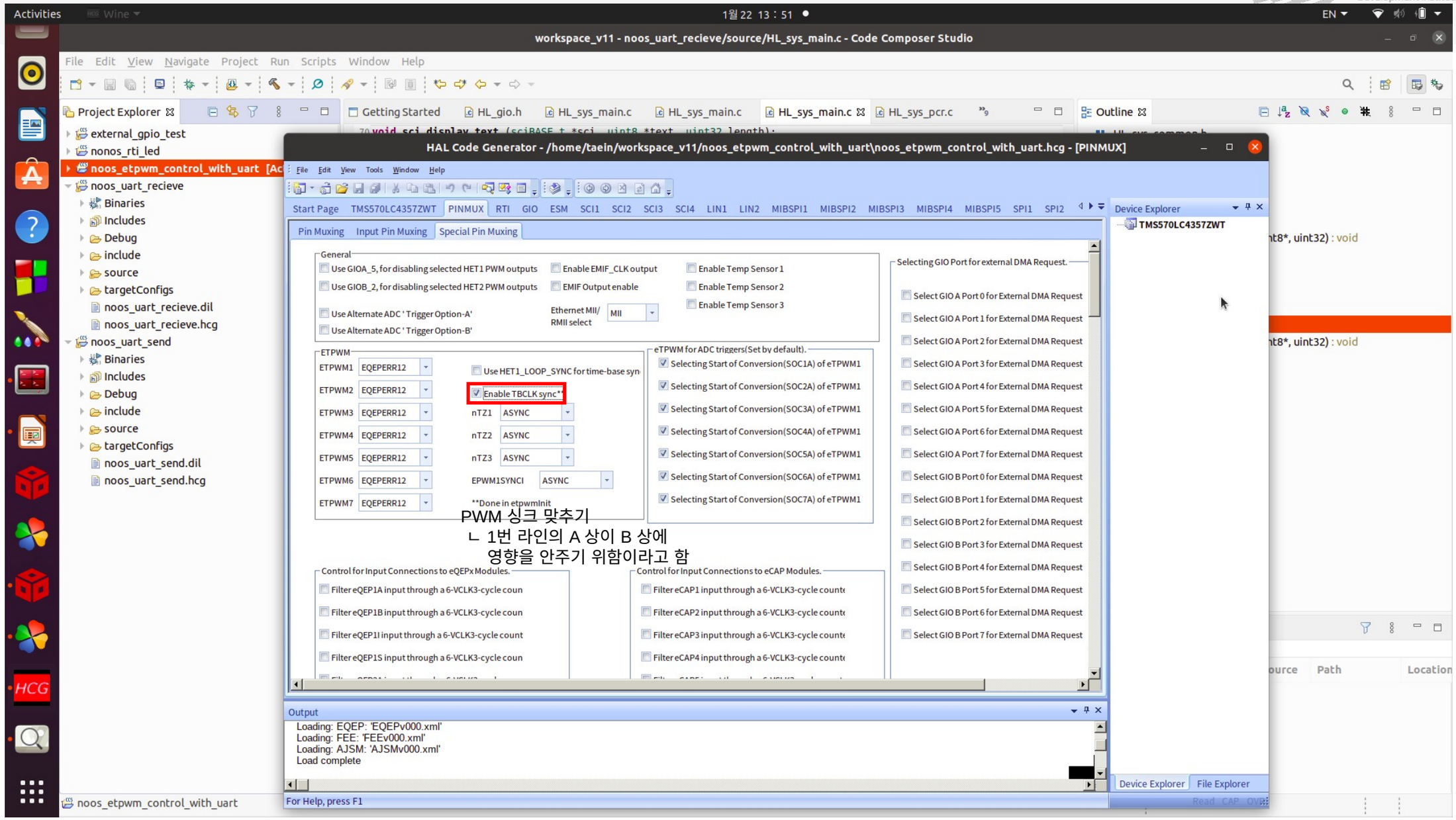
etPWM(w/UART)


Activities
Code Composer Studio
1월 22 13 : 49
EN
workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Project Explorer
external_gpio_test
nonos_rti_led
noos_etpwm_control_with_uart [Ac
noos_uart_recieve
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_recieve.dil
noos_uart_recieve.hcg
noos_uart_send
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_send.dil
noos_uart_send.hcg
Getting Started
HL_gio.h
HL_sys_main.c
HL_sys_main.c
HL_sys_main.c
HL_sys_pcr.c
Outline
HAL Code Generator - /home/taein/workspace_v11/noos_etpwm_control_with_uart\noos_etpwm_control_with_uart.hcg - [GIO]
File Edit View Tools Window Help
TMS570LC4357ZWT PINMUX RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1 SPI2 SPI3 SPI4
Port A Port B
DOUT:
DIR:
PDR:
PSL:
DIN:
VIM:
Low Priority:
Falling Edge:
Bit 4
4번 Port 출력
GIOA[4]
High Priority:
Enable:
Rising Edge:
Bit 5
GIOA[5]
Device Explorer
TMS570LC4357ZWT
Output
Loading: EQEP: 'EQEPv000.xml'
Loading: FEE: 'FEEv000.xml'
Loading: AJSM: 'AJSMv000.xml'
Load complete
For Help, press F1
Device Explorer
File Explorer
Path
Location
noos_etpwm_control_with_uart

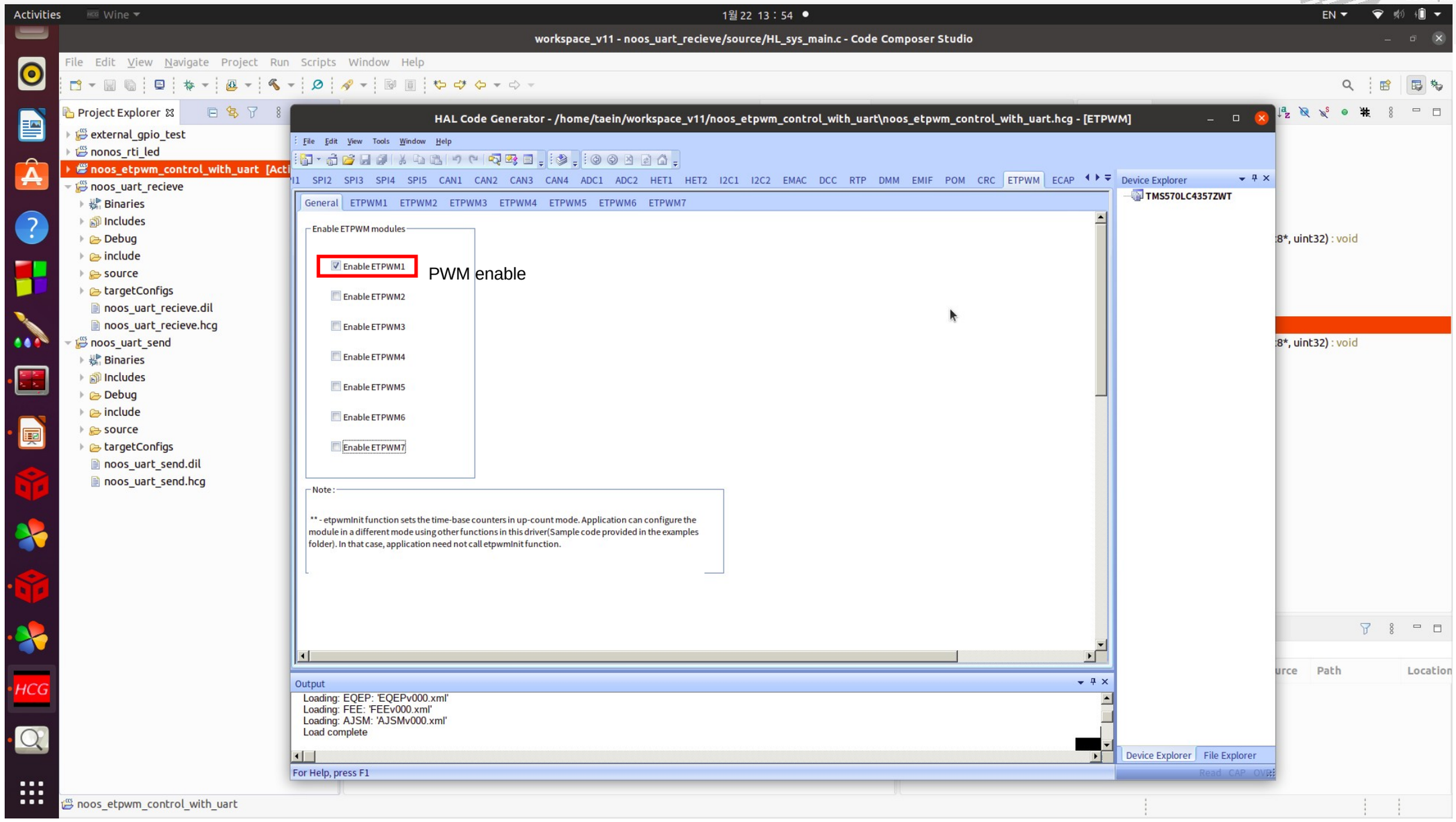
etPWM(w/UART)


workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
HAL Code Generator - /home/taein/workspace_v11/noos_etpwm_control_with_uart\noos_etpwm_control_with_uart.hcg - [PINMUX]
PWM 포트(hw상 GIOA[5])

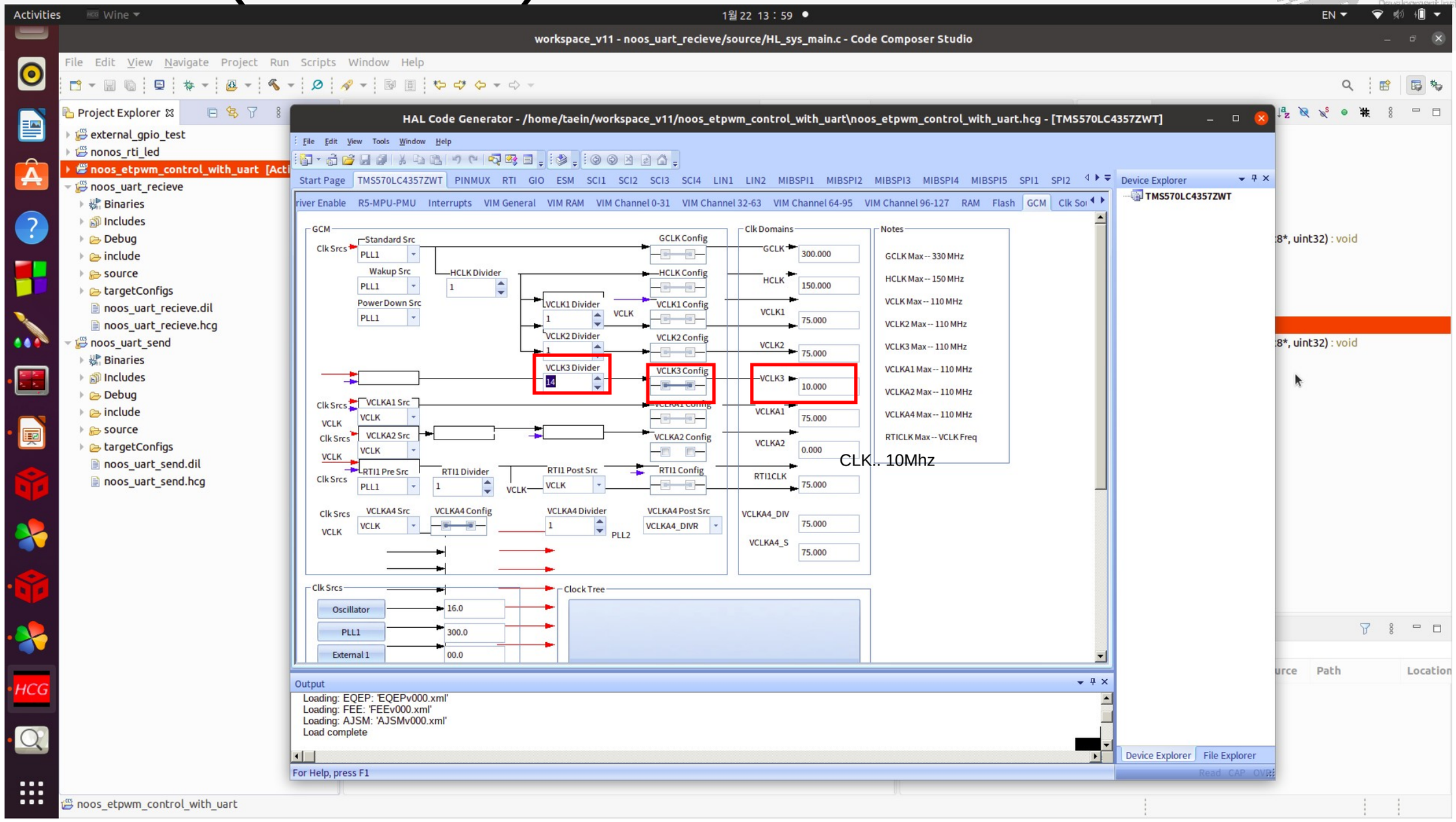
etPWM(w/UART)


workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
HAL Code Generator - /home/taein/workspace_v11/noos_etpwm_control_with_uart\noos_etpwm_control_with_uart.hcg - [PINMUX]
Enable TBCLK sync**
PWM 싱크 맞추기
ㄴ 1번 라인의 A 상이 B 상에
영향을 안주기 위함이라고 함

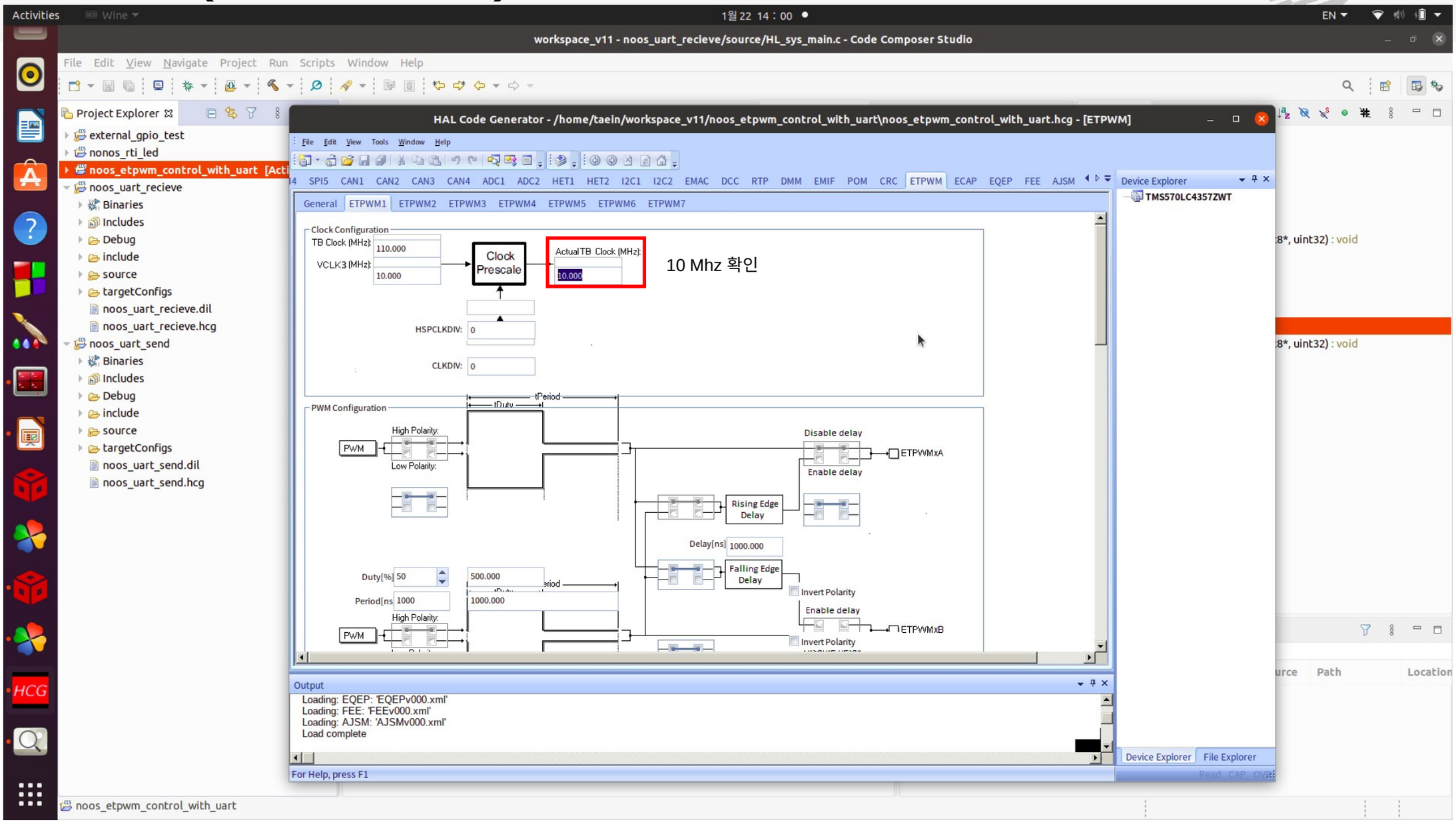
etPWM(w/UART)


Activities
Wine
1월 22 13 : 54
EN
workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Project Explorer
external_gpio_test
nonos_rti_led
noos_etpwm_control_with_uart
noos_uart_recieve
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_recieve.dil
noos_uart_recieve.hcg
noos_uart_send
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_send.dil
noos_uart_send.hcg
HAL Code Generator - /home/taein/workspace_v11/noos_etpwm_control_with_uart\noos_etpwm_control_with_uart.hcg - [ETPWM]
File Edit View Tools Window Help
SPI2 SPI3 SPI4 SPI5 CAN1 CAN2 CAN3 CAN4 ADC1 ADC2 HET1 HET2 I2C1 I2C2 EMAC DCC RTP DMM EMIF POM CRC ETPWM ECAP
General ETPWM1 ETPWM2 ETPWM3 ETPWM4 ETPWM5 ETPWM6 ETPWM7
Enable ETPWM modules
Enable ETPWM1
PWM enable
Enable ETPWM2
Enable ETPWM3
Enable ETPWM4
Enable ETPWM5
Enable ETPWM6
Enable ETPWM7
Note :
** - etpwmInit function sets the time-base counters in up-count mode. Application can configure the module in a different mode using other functions in this driver(Sample code provided in the examples folder). In that case, application need not call etpwmInit function.
Device Explorer
TMS570LC4357ZWT
Output
Loading: EQEP: 'EQEPv000.xml'
Loading: FEE: 'FEEv000.xml'
Loading: AJSM: 'AJSMv000.xml'
Load complete
For Help, press F1
Device Explorer
File Explorer
Path
Location
noos_etpwm_control_with_uart

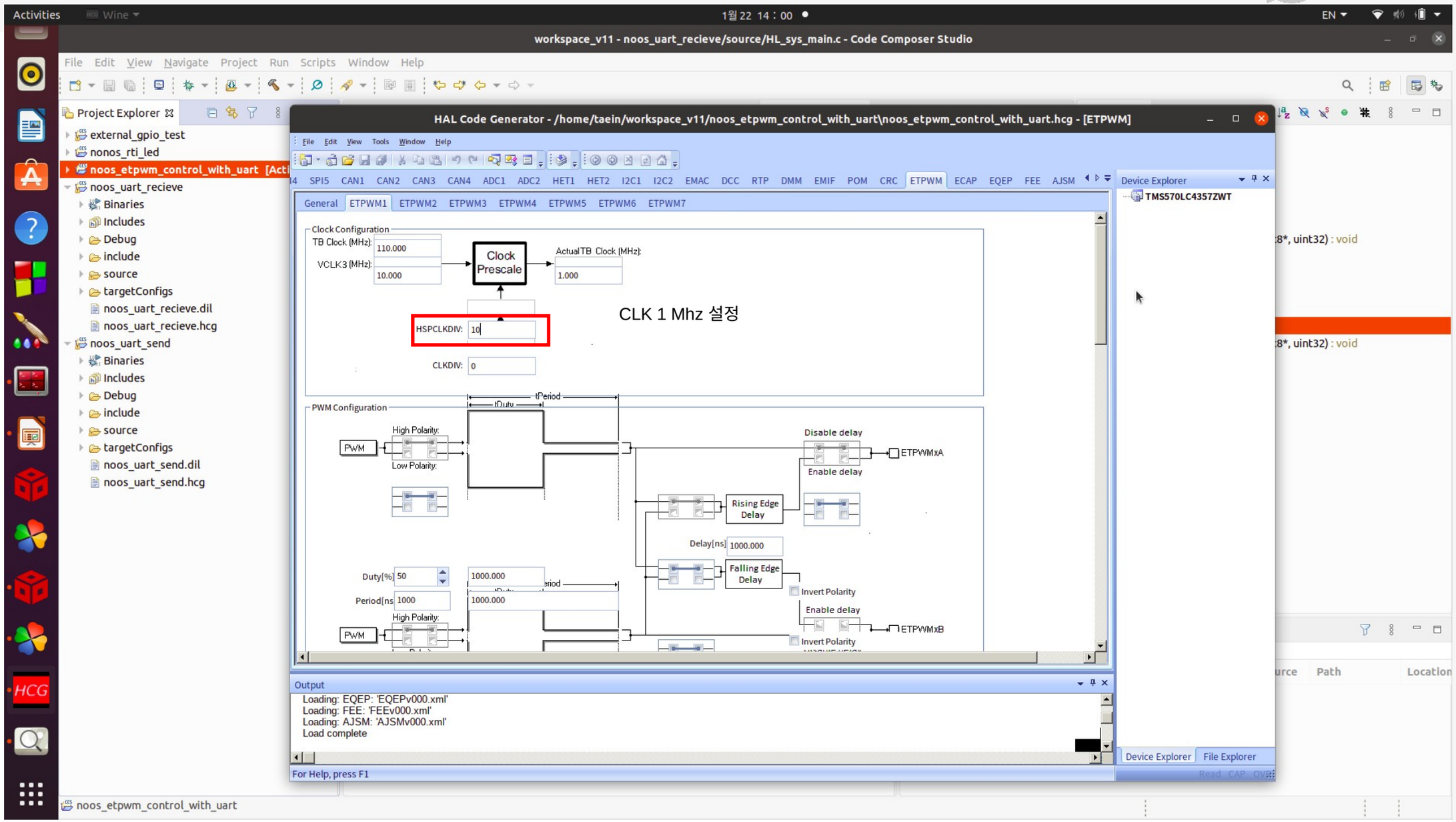
etPWM(w/UART)


workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
HAL Code Generator - /home/taein/workspace_v11/noos_etpwm_control_with_uart\noos_etpwm_control_with_uart.hcg - [TMS570LC4357ZWT]
CLK.. 10Mhz

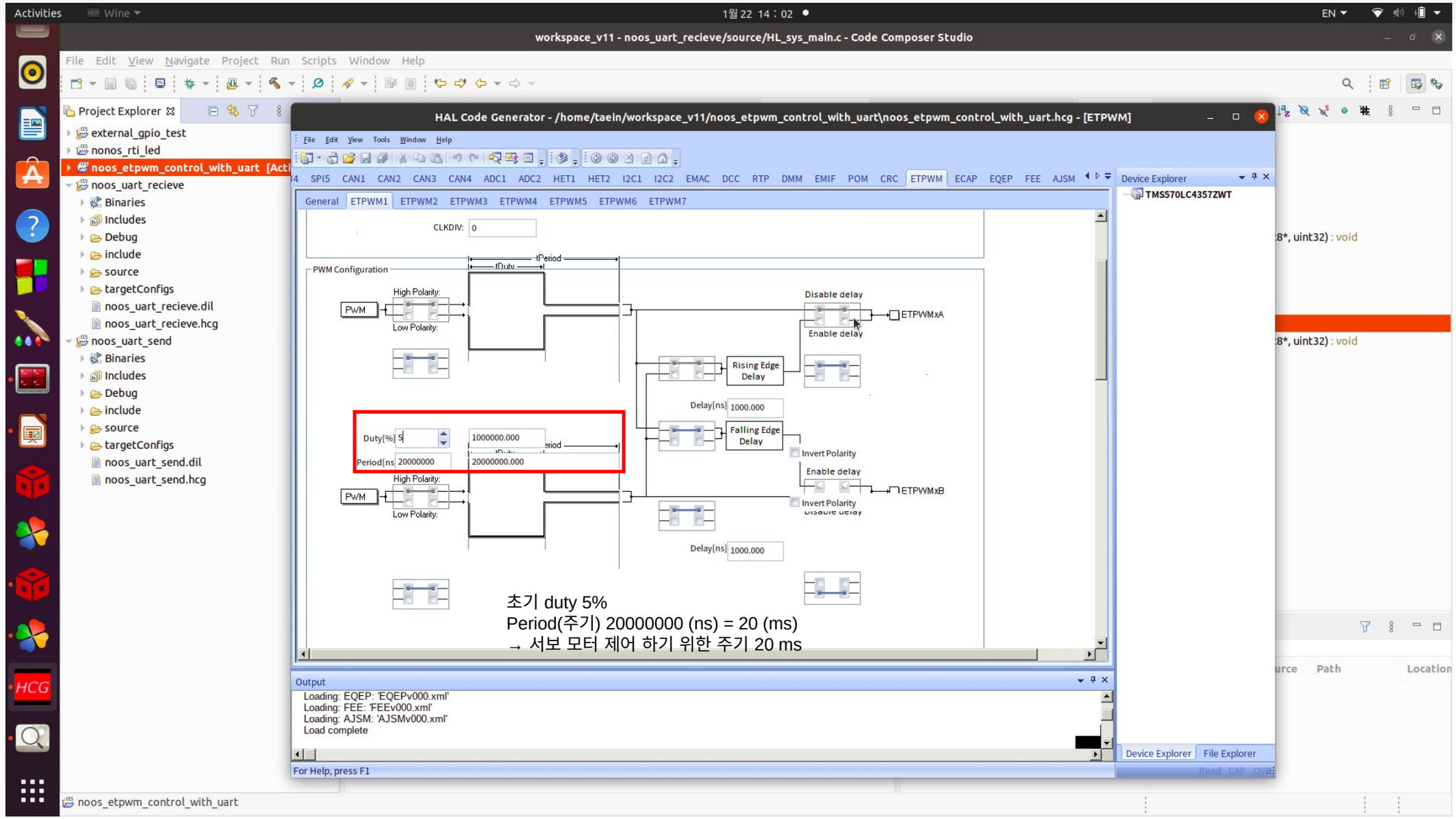
etPWM(w/UART)


workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
HAL Code Generator - /home/taein/workspace_v11/noos_etpwm_control_with_uart\noos_etpwm_control_with_uart.hcg - [ETPWM]
Clock Configuration
TB Clock (MHz): 110.000
VCLK3 (MHz): 10.000
Clock Prescale
Actual TB Clock (MHz): 10.000
10 Mhz 확인
HSPCLKDIV: 0
CLKDIV: 0
PWM Configuration
High Polarity:
Low Polarity:
Disable delay
Enable delay
ETPWMxA
Rising Edge Delay
Delay[ns] 1000.000
Falling Edge Delay
Invert Polarity
Duty[%] 50
Period[ns] 1000
500.000
1000.000
ETPWMxB
Output
Loading: EQEP: 'EQEPv000.xml'
Loading: FEE: 'FEEv000.xml'
Loading: AJSM: 'AJSMv000.xml'
Load complete
For Help, press F1
Device Explorer
TMS570LC4357ZWT
Project Explorer
external_gpio_test
nonos_rti_led
noos_etpwm_control_with_uart
noos_uart_recieve
noos_uart_send

etPWM(w/UART)


Activities
Wine
1월 22 14 : 00
EN
workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Project Explorer
external_gpio_test
nonos_rti_led
noos_etpwm_control_with_uart [Act
noos_uart_recieve
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_recieve.dil
noos_uart_recieve.hcg
noos_uart_send
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_send.dil
noos_uart_send.hcg
HAL Code Generator - /home/taein/workspace_v11/noos_etpwm_control_with_uart\noos_etpwm_control_with_uart.hcg - [ETPWM]
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
General ETPWM1 ETPWM2 ETPWM3 ETPWM4 ETPWM5 ETPWM6 ETPWM7
Clock Configuration
TB Clock (MHz): 110.000
VCLK3 (MHz): 10.000
Clock Prescale
Actual TB Clock (MHz): 1.000
HSPCLKDIV: 10
CLKDIV: 0
CLK 1 Mhz 설정
PWM Configuration
tPeriod
tDuty
PWM
High Polarity:
Low Polarity:
Disable delay
ETPWMxA
Enable delay
Rising Edge Delay
Delay[ns] 1000.000
Falling Edge Delay
Invert Polarity
Enable delay
ETPWMxB
Invert Polarity
Duty[%] 50
Period[ns] 1000
1000.000
1000.000
Output
Loading: EQEP: 'EQEPv000.xml'
Loading: FEE: 'FEEv000.xml'
Loading: AJSM: 'AJSMv000.xml'
Load complete
For Help, press F1
Device Explorer
TMS570LC4357ZWT
Device Explorer
File Explorer
Path
Location
noos_etpwm_control_with_uart

etPWM(w/UART)


workspace_v11 - noos_uart_recieve/source/HL_sys_main.c - Code Composer Studio
HAL Code Generator - /home/taein/workspace_v11/noos_etpwm_control_with_uart\noos_etpwm_control_with_uart.hcg - [ETPWM]
Project Explorer
external_gpio_test
nonos_rti_led
noos_etpwm_control_with_uart
noos_uart_recieve
Binaries
Includes
Debug
include
source
targetConfigs
noos_uart_recieve.dil
noos_uart_recieve.hcg
noos_uart_send
noos_uart_send.dil
noos_uart_send.hcg
ETPWM
General
ETPWM1
ETPWM2
ETPWM3
ETPWM4
ETPWM5
ETPWM6
ETPWM7
CLKDIV: 0
PWM Configuration
tPeriod
tDuty
High Polarity:
Low Polarity:
PWM
Disable delay
Enable delay
ETPWMxA
Rising Edge Delay
Delay[ns] 1000.000
Falling Edge Delay
Duty[%] 5
1000000.000
Period[ns] 20000000
20000000.000
Invert Polarity
ETPWMxB
Device Explorer
TMS570LC4357ZWT
Output
Loading: EQEP: 'EQEPv000.xml'
Loading: FEE: 'FEEv000.xml'
Loading: AJSM: 'AJSMv000.xml'
Load complete
For Help, press F1
초기 duty 5%
Period(주기) 20000000 (ns) = 20 (ms)
→ 서보 모터 제어 하기 위한 주기 20 ms

etPWM(w/UART) – code 및 내용 분석


```
50 #include "HL_sys_common.h"
51
52 /* USER CODE BEGIN (1) */
53 #include "HL_gio.h"
54 #include "HL_sci.h"
55 #include "HL_etpwm.h"
56 #include "HL_system.h"
57
58 #include <string.h>
59 #include <stdio.h>
```

* 헤더 선언
  ㄴ gio, sci, etpwm, system
  sci에 활용할 string.h, stdio.h

```
86 void sci_display_text(sciBASE_t *sci, uint8 *text, uint32 length)
87 {
88     while(length--)
89     {
90         while ((UART->FLR & 0x4) == 4)       // 조건문에 해당되면 busy 하다는 얘기
91         ;
92
93          sciSendByte(UART, *text++);        // 1 바이트씩 움직일 거니까 포인터로
94     }
95 }
96
97 void wait(uint32 time)
98 {
99     int i;
00
01     for(i=0; i<time; i++)
02     ;
03 }
```

앞 서 있었던 내용 이므로
설명은 생략

```
70 /* USER CODE BEGIN (2) */
71 void sci_display_text (sciBASE_t *sci, uint8 *text, uint32 length);
72 void wait (uint32 time);
73 void set_pwm();
74
75 #define UART    sciREG1
76
77 uint32 receive_data;
78 uint32 pwm_value;
79 uint32 tmp;
80
81 uint32 pwm_duty[] = { 500, 750, 1000, 1250, 1500, 1750, 2000, 2250, 2500 }; // 20000000ns(20ms) 에서 500이
82                                                                              // 쉽게 생각해서 전체 pwm 이 20이라고 봤을 때
83                                                                              //500이 0.5 정도
```

Putty 창에 uart 통신 나타내기 위한 함수

UART는 sciREG1

Duty 비 제어에 활용 예정

etPWM(w/UART) – code 및 내용 분석

EDDI
Electronic Design Development Institute

```c
int main(void)
{
/* USER CODE BEGIN (3) */
    char buf[128];
    unsigned int buf_len;

    sciInit();

    sprintf(buf, "*SCI Init Success!\n\r\0");
    buf_len = strlen(buf);
    sci_display_text(UART, (uint8 *)buf, buf_len);

    gioInit();

    sprintf(buf, "*GIO Init Success!\n\r\0");
    buf_len = strlen(buf);
    sci_display_text(UART, (uint8 *)buf, buf_len);

    etpwmInit();

    sprintf(buf, "*EPWM Init Success!\n\r\0");
    buf_len = strlen(buf);
    sci_display_text(UART, (uint8 *)buf, buf_len);

    sprintf(buf, "Press Number!\n\r\0");
    buf_len = strlen(buf);
    sci_display_text(UART, (uint8 *)buf, buf_len);
```

Sci , gio, etpwm init 과
숫자 입력 받도록 만듦 → 입력 하는 숫자에 따라 duty 비 설정하게 됨

이번에 중점적으로 알아 봐야 할 부분은 etpwmInit() !!

35.1 Introduction

An effective PWM peripheral must be able to generate complex pulse width waveforms with minimal CPU overhead or intervention. It needs to be highly programmable and very flexible while being easy to understand and use. The ePWM unit described here addresses these requirements by allocating all needed timing and control resources on a per PWM channel basis. Cross coupling or sharing of resources has been avoided; instead, the ePWM is built up from smaller single channel modules with separate resources that can operate together as required to form a system. This modular approach results in an orthogonal architecture and provides a more transparent view of the peripheral structure, helping users to understand its operation quickly.

In this document the letter x within a signal or module name is used to indicate a generic ePWM instance on a device. For example output signals EPWMxA and EPWMxB refer to the output signals from the ePWMx instance. Thus, EPWM1A and EPWM1B belong to ePWM1 and likewise EPWM4A and EPWM4B belong to ePWM4.

이 문서에서 신호 또는 모듈 이름 내의 문자 x는 일반 ePWM 장치에 인스턴스를 나타내는 데 사용됩니다.

예를 들어 출력 신호 EPWMxA 및 EPWMxB는 ePWMx 의 인스턴스.
따라서 EPWM1A 및 EPWM1B는 ePWM1에 속하고 마찬가지로
EPWM4A 및 EPWM4B는 ePWM4에 속합니다.

etPWM(w/UART) – code 및 내용 분석

ePWM 모듈은 2개의 PWM 출력(EPWMxA 및 EPWMxB)으로 구성된 하나의 완전한 PWM 채널을 나타냅니다. 여러 ePWM 모듈은 그림 35-1과 같이 장치 내에서 인스턴스화됩니다. 각 ePWM 인스턴스는 동일하며 1부터 시작하는 숫자 값으로 표시됩니다. 예를 들어, ePWM1은 시스템의 첫 번째 인스턴스이고 ePWM3은 세 번째 인스턴스이며 ePWMx는 임의의 인스턴스를 나타냅니다.

ePWM 모듈은 클록 동기화 방식을 통해 함께 연결되어 필요할 때 단일 시스템으로 작동할 수 있습니다. 또한 이 동기화 체계는 eCAP(Capture Peripheral Module)로 확장될 수 있습니다. 모듈은 독립 실행형으로도 작동할 수 있습니다.

35.1.1 Submodule Overview

The ePWM module represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB. Multiple ePWM modules are instanced within a device as shown in Figure 35-1. Each ePWM instance is identical and is indicated by a numerical value starting with 1. For example, ePWM1 is the first instance and ePWM3 is the third instance in the system, and ePWMx indicates any instance.

The ePWM modules are chained together via a clock synchronization scheme that allows them to operate as a single system when required. Additionally, this synchronization scheme can be extended to the capture peripheral modules (eCAP). Modules can also operate stand-alone.

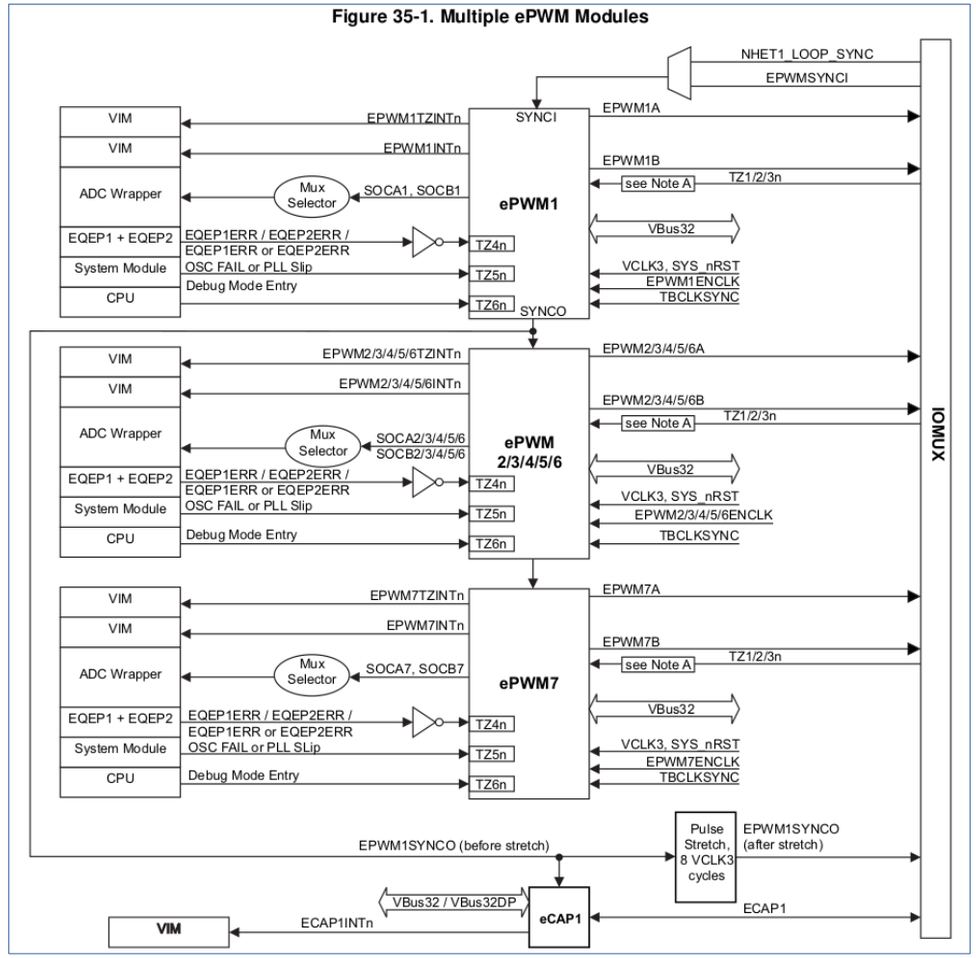
Each ePWM module supports the following features:

- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations:
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software.
- Programmable phase-control support for lag or lead operation relative to other ePWM modules.
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis.
- Dead-band generation with independent rising and falling edge delay control.
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions.
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs.
- All events can trigger both CPU interrupts and ADC start of conversion (SOC)
- Programmable event prescaling minimizes CPU overhead on interrupts.
- PWM chopping by high-frequency carrier signal, useful for pulse transformer gate drives.

Each ePWM module is connected to the input/output signals shown in Figure 35-1. The signals are described in detail in subsequent sections.

Each ePWM module consists of eight submodules and is connected within a system via the signals shown in Figure 35-2.

Figure 35-1. Multiple ePWM Modules



etPWM(w/UART) – code 및 내용 분석

EDDI
Electronic Design Development Institute

35.1.1 Submodule Overview

The ePWM module represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB. Multiple ePWM modules are instanced within a device as shown in Figure 35-1. Each ePWM instance is identical and is indicated by a numerical value starting with 1. For example, ePWM1 is the first instance and ePWM3 is the third instance in the system, and ePWMx indicates any instance.

The ePWM modules are chained together via a clock synchronization scheme that allows them to operate as a single system when required. Additionally, this synchronization scheme can be extended to the capture peripheral modules (eCAP). Modules can also operate stand-alone.

Each ePWM module supports the following features:

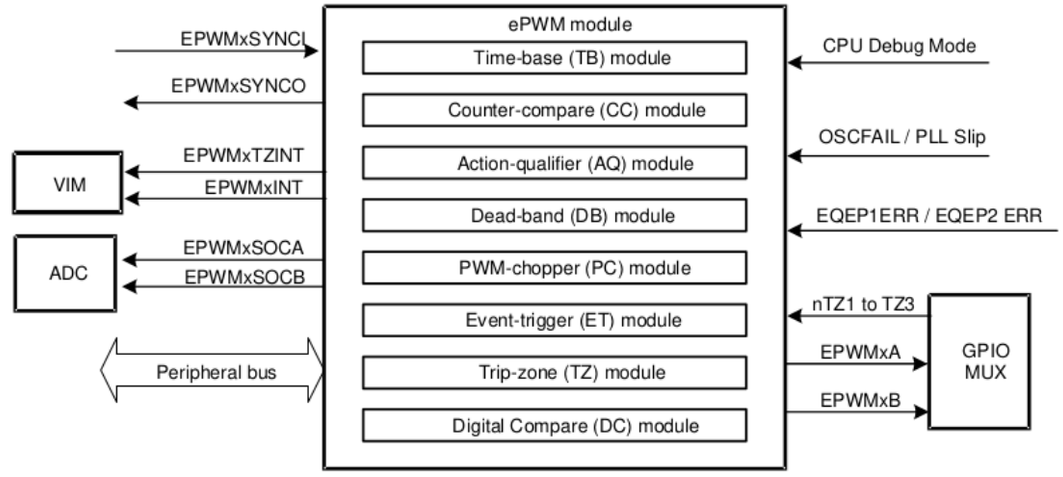
- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations:
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software.
- Programmable phase-control support for lag or lead operation relative to other ePWM modules.
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis.
- Dead-band generation with independent rising and falling edge delay control.
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions.
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs.
- All events can trigger both CPU interrupts and ADC start of conversion (SOC)
- Programmable event prescaling minimizes CPU overhead on interrupts.
- PWM chopping by high-frequency carrier signal, useful for pulse transformer gate drives.

Each ePWM module is connected to the input/output signals shown in Figure 35-1. The signals are described in detail in subsequent sections.

Each ePWM module consists of eight submodules and is connected within a system via the signals shown in Figure 35-2.

각 ePWM 모듈은 다음 기능을 지원합니다. • 주기 및 주파수 제어 기능이 있는 전용 16비트 타임베이스 카운터 • 다음 구성에서 사용할 수 있는 2개의 PWM 출력 (EPWMxA 및 EPWMxB): – 단일 에지 작동이 가능한 2개의 독립적인 PWM 출력 – 듀얼 에지 대칭 작동이 가능한 2개의 독립적인 PWM 출력 – 듀얼 에지 비대칭 작동을 지원하는 1개의 독립적인 PWM 출력 • 소프트웨어를 통한 PWM 신호의 비동기식 오버라이드 제어. • 다른 ePWM 모듈에 비해 지연 또는 선행 작동을 위한 프로그래밍 가능한 위상 제어 지원. • 주기별로 하드웨어 잠금(동기화) 단계 관계. • 독립적인 상승 및 하강 에지 지연 제어를 통한 불감대 생성. • 사이클별 트립 및 오류 조건에 대한 원샷 트립 모두의 프로그래밍 가능한 트립 영역 할당. • 트립 조건은 PWM 출력에서 하이, 로우 또는 하이 임피던스 상태 로직 레벨을 강제할 수 있습니다. • 모든 이벤트는 CPU 인터럽트와 ADC 변환 시작(SOC)을 모두 트리거할 수 있습니다. • 프로그래밍 가능한 이벤트 프리스케일링은 인터럽트 시 CPU 오버헤드를 최소화합니다. • 고주파수 캐리어 신호에 의한 PWM 초핑, 펄스 트랜스포머 게이트 드라이브에 유용합니다. 각 ePWM 모듈은 그림 35-1과 같은 입력/출력 신호에 연결됩니다. 신호는 다음 섹션에서 자세히 설명합니다. 각 ePWM 모듈은 8개의 하위 모듈로 구성되며 그림 35-2에 표시된 신호를 통해 시스템 내에서 연결됩니다.

Figure 35-2. Submodules and Signal Connections for an ePWM Module



etPWM(w/UART) – code 및 내용 분석

EDDI
Electronic Design Development Institute

```
void etpwmInit(void)
{
/* USER CODE BEGIN (1) */
/* USER CODE END */

    /** @b initialize @b ETPWM1 */

    /** - Sets high speed time-base clock prescale bits */
    etpwmREG1->TBCTL = (uint16)5U << 7U;

    /** - Sets time-base clock prescale bits */
    etpwmREG1->TBCTL |= (uint16)((uint16)0U << 10U);
```

```
uint16 TBCTL;        /**<  0x0002 Time-Base Control Register*/
```

7을 왼쪽으로 5bit shift → 0111 → 1110 0000

10을 왼쪽으로 0 bit shift → 1010

따라서 TBCTL은 1110 1010

Table 35-1. ePWM Module Control and Status Register Set Grouped by Submodule

Name	Address Offset[1]	Size (x16)	Shadow	Privileged Mode Write Only?	Description
Time-Base Submodule Registers					
TBSTS	00h	1	No	No	Time-Base Status Register
TBCTL	02h	1	No	No	Time-Base Control Register

Figure 35-64. Time-Base Control Register (TBCTL) [offset = 02h]

15	14	13	12	10	9	8
FREE	SOFT	PHSDIR	CLKDIV		HSPCLKDIV	
R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	

7	6	5	4	3	2	1	0
HSPCLKDIV	SWFSYNC	SYNCOSEL		PRDLD	PHSEN	CTRMODE	
R/W-1	R/W-0	R/W-0		R/W-0	R/W-0	R/W-3h	

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

9-7	HSPCLKDIV		High Speed Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value: TBCLK = VCLK3 / (HSPCLKDIV × CLKDIV)
		0	/1
		1h	/2 (default on reset)

Bit	Field	Value	Description
6	SWFSYNC		Software Forced Synchronization Pulse.
		0	Writing a 0 has no effect and reads always return a 0.
		1	Writing a 1 forces a one-time synchronization pulse to be generated.
			This event is ORed with the EPWMxSYNCI input of the ePWM module.
			SWFSYNC is valid (operates) only when EPWMxSYNCI is selected by SYNCOSEL = 00.

1을 쓰면 일회성 동기화 펄스가 생성됩니다. 이 이벤트는 ePWM 모듈의 EPWMxSYNCI 입력과 OR 연결됩니다. SWFSYNC는 EPWMxSYNCI가 SYNCOSEL = 00으로 선택된 경우에만 유효(작동)합니다.

5-4	SYNCOSEL		Synchronization Output Select. These bits select the source of the EPWMxSYNCO signal.
		0	EPWMxSYNC
		1h	CTR = zero: Time-base counter equal to zero (TBCTR = 0x0000)
		2h	CTR = CMPB : Time-base counter equal to counter-compare B (TBCTR = CMPB)
		3h	Disable EPWMxSYNCO signal

CTR = CMPB : 카운터 비교 B와 동일한 시간 기반 카운터(TBCTR = CMPB)

etPWM(w/UART) – code 및 내용 분석

따라서 TBCTL은 1110 1010

Figure 35-64. Time-Base Control Register (TBCTL) [offset = 02h]

15	14	13	12	10	9	8
FREE	SOFT	PHSDIR	CLKDIV		HSPCLKDIV	
R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	

7	6	5	4	3	2	1	0
HSPCLKDIV	SWFSYNC	SYNCOSEL		PRDLD	PHSEN	CTRMODE	
R/W-1	R/W-0	R/W-0		R/W-0	R/W-0	R/W-3h	

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

3	PRDLD		Active Period Register Load From Shadow Register Select.
		0	The period register (TBPRD) is loaded from its shadow register when the time-base counter, TBCTR, is equal to zero. A write or read to the TBPRD register accesses the shadow register.
		1	Load the TBPRD register immediately without using a shadow register. A write or read to the TBPRD register directly accesses the active register.

섀도우 레지스터를 사용하지 않고 즉시 TBPRD 레지스터를 로드합니다. TBPRD 레지스터에 대한 쓰기 또는 읽기는 활성 레지스터에 직접 액세스합니다.

1-0	CTRMODE		Counter Mode. The time-base counter mode is normally configured once and not changed during normal operation. If you change the mode of the counter, the change will take effect at the next TBCLK edge and the current counter value shall increment or decrement from the value before the mode change. These bits set the time-base counter mode of operation as follows:
		0	Up-count mode
		1h	Down-count mode
		2h	Up-down-count mode
		3h	Stop-freeze counter operation (default on reset)

카운터 모드.
시간 기반 카운터 모드는 일반적으로 한 번 구성되며 정상 작동 중에는 변경되지 않습니다. 카운터 모드를 변경하면 다음 TBCLK 에지에서 변경 사항이 적용되고 현재 카운터 값은 모드 변경 전 값에서 증가하거나 감소합니다.

etPWM(w/UART) – code 및 내용 분석

```
void etpwmInit(void)
{

    /** - Sets time period or frequency for ETPWM block both PWMA and PWMB*/
    etpwmREG1->TBPRD = 19999U;
```

```
    uint16 TBPRD;           /**<  0x0008 Time-Base Period Register*/
```

08h	TBPRD	Time-Base Period Register

Table 35-26. Time-Base Period Register (TBPRD) Field Descriptions

Bits	Name	Description
15-0	TBPRD	These bits determine the period of the time-base counter. This sets the PWM frequency. Shadowing of this register is enabled and disabled by the TBCTL[PRDLD] bit. By default this register is shadowed. • If TBCTL[PRDLD] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the active register will be loaded from the shadow register when the time-base counter equals 0. • If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware. • The active and shadow registers share the same memory map address.

이 비트는 시간축 카운터의 주기를 결정합니다. PWM 주파수를 설정합니다. 이 레지스터의 섀도잉은 TBCTL[PRDLD] 비트에 의해 활성화 및 비활성화됩니다. 기본적으로 이 레지스터는 숨겨져 있습니다.

• TBCTL[PRDLD] = 0 이면 섀도우가 활성화되고 쓰기 또는 읽기가 자동으로 섀도우 레지스터로 이동합니다. 이 경우 활성 레지스터는 시간이 흐를 때 섀도우 레지스터에서 로드됩니다. 기본 카운터는 0입니다.

• TBCTL[PRDLD] = 1 이면 섀도우가 비활성화되고 모든 쓰기 또는 읽기가 활성 레지스터, 즉 하드웨어를 능동적으로 제어하는 레지스터로 직접 이동합니다.

• 활성 및 섀도우 레지스터는 동일한 메모리 맵 주소를 공유합니다.

etPWM(w/UART) – code 및 내용 분석

```
void etpwmInit(void)
{
    /** - Setup the duty cycle for PWMA */
    etpwmREG1->CMPA = 1000U;
```

```
uint16 CMPA;            /**<  0x0010 Counter-Compare A Register*/
```

CMPB 도 유사하다.

10h	CMPA	Counter-Compare A Register

활성 CMPA 레지스터의 값은 TBCTR(타임 베이스 카운터)과 지속적으로 비교됩니다. 값이 같을 때 카운터 비교 모듈은 "시간 기반 카운터가 카운터 비교 A와 같음" 이벤트를 생성합니다. 이 이벤트는 자격이 부여되고 하나 이상의 작업으로 변환되는 action-qualifier로 전송됩니다. 이러한 작업은 AQCTLA 및 AQCTLB 레지스터의 구성에 따라 EPWMxA 또는 EPWMxB 출력에 적용될 수 있습니다. AQCTLA 및 AQCTLB 레지스터에서 정의할 수 있는 작업은 다음과 같습니다.

Table 35-29. Counter-Compare A Register (CMPA) Field Descriptions

Bits	Name	Description
15-0	CMPA	The value in the active CMPA register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare A" event. This event is sent to the action-qualifier where it is qualified and converted it into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include: • Do nothing; the event is ignored. • Clear: Pull the EPWMxA and/or EPWMxB signal low. • Set: Pull the EPWMxA and/or EPWMxB signal high. • Toggle the EPWMxA and/or EPWMxB signal. Shadowing of this register is enabled and disabled by the CMPCTL[SHDWAMODE] bit. By default this register is shadowed. • If CMPCTL[SHDWAMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register. • Before a write, the CMPCTL[SHDWAFULL] bit can be read to determine if the shadow register is currently full. • If CMPCTL[SHDWAMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware. • In either mode, the active and shadow registers share the same memory map address.

Do nothing ; 이벤트는 무시됩니다.
- 지우기: EPWMxA 및/또는 EPWMxB 신호를 로우로 당깁니다.
- 설정: EPWMxA 및/또는 EPWMxB 신호를 하이로 당깁니다.
EPWMxA 및/또는 EPWMxB 신호를 전환합니다.

이 레지스터의 섀도잉은 CMPCTL[SHDWAMODE] 비트에 의해 활성화 및 비활성화됩니다. 기본적으로 이 레지스터는 숨겨져 있습니다.

etPWM(w/UART) – code 및 내용 분석

```
void etpwmInit(void)
{
    /** - Force EPWMxA output high when counter reaches zero and low when counter reaches Compare A value */
    etpwmREG1->AQCTLA = ((uint16)((uint16)ActionQual_Set   << 0U)
                       | (uint16)((uint16)ActionQual_Clear << 4U));
```

14h	AQCTLA	Action-Qualifier Control Register for Output A (EPWMxA)

카운터가 0 에 도달하면 강제 EPWMxA 출력을 높이고 카운터가 비교 A 값에 도달하면 로우

Set 은 '2' Clear 는 '1'

```
uint16 AQCTLA;        /**<  0x0014 Action-Qualifier Control Register for Output A (ETPWMxA)*/
```

결론은 1000

Figure 35-71. Action-Qualifier Output A Control Register (AQCTLA) [offset = 14h]

15–12	11–10	9–8	7–6	5–4	3–2	1–0
Reserved	CBD	CBU	CAD	CAU	PRD	ZRO
R-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

3-2	PRD		Action when the counter equals the period. 주기와 동일시 Action
			Note: By definition, in count up-down mode when the counter equals period the direction is defined as 0 or counting down.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
1-0	ZRO		Action when counter equals zero.
			Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.

etPWM(w/UART) – code 및 내용 분석

```
void etpwmInit(void)
{
    /** - Mode setting for Dead Band Module
    *     -Select the input mode for Dead Band Module
    *     -Select the output mode for Dead Band Module
    *     -Select Polarity of the output PWMs
    */
    etpwmREG1->DBCTL = ((uint16)((uint16)0U << 5U)    /* Source for Falling edge delay(0-PWMA, 1-PWMB) */
                      | (uint16)((uint16)0u << 4U)    /* Source for Rising edge delay(0-PWMA, 1-PWMB)  */
                      | (uint16)((uint16)0U << 3U)    /* Enable/Disable EPWMxB invert       */
                      | (uint16)((uint16)0U << 2U)    /* Enable/Disable EPWMxA invert       */
                      | (uint16)((uint16)0U << 1U)    /* Enable/Disable Rising Edge Delay   */
                      | (uint16)((uint16)0U << 0U));  /* Enable/Disable Falling Edge Delay  */
```

Dead-Band Generator Submodule Registers

1Ch	DBCTL	Dead-Band Generator Control Register	Section 35.4.4.1

```
uint16 DBCTL;        /**<  0x001C Dead-Band Generator Control Register*/
```

각 설정을 모두 마치면 0111 h

35.4.4.1 Dead-Band Generator Control Register (DBCTL)

Figure 35-75. Dead-Band Generator Control Register (DBCTL) [offset = 1Ch]

15	14	8
HALFCYCLE	Reserved	
R/W-0	R-0	

7	6	5	4	3	2	1	0
Reserved		IN_MODE		POLSEL		OUT_MODE	
R-0		R/W-0		R/W-0		R/W-0	
				0	1	1	1

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

etPWM(w/UART) – code 및 내용 분석

EDDI
Electronic Design Development Institute

35.4.4.1 Dead-Band Generator Control Register (DBCTL)

Figure 35-75. Dead-Band Generator Control Register (DBCTL) [offset = 1Ch]

15	14						8
HALFCYCLE	Reserved						
R/W-0	R-0						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		IN_MODE		POLSEL		OUT_MODE	
R-0		R/W-0		R/W-0		R/W-0	
				0	1	1	1

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

Table 35-35. Dead-Band Generator Control Register (DBCTL) Field Descriptions (continued)

Bits	Name	Value	Description
1-0	OUT_MODE		Dead-band Output Mode Control. Bit 1 controls the S1 switch and bit 0 controls the S0 switch shown in Figure 35-28. This allows you to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.
		0	Dead-band generation is bypassed for both output signals. In this mode, both the EPWMxA and EPWMxB output signals from the action-qualifier are passed directly to the PWM-chopper submodule. In this mode, the POLSEL and IN_MODE bits have no effect.
		1h	Disable rising-edge delay. The EPWMxA signal from the action-qualifier is passed straight through to the EPWMxA input of the PWM-chopper submodule. The falling-edge delayed signal is seen on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].
		2h	The rising-edge delayed signal is seen on output EPWMxA. The input signal for the delay is determined by DBCTL[IN_MODE]. Disable falling-edge delay. The EPWMxB signal from the action-qualifier is passed straight through to the EPWMxB input of the PWM-chopper submodule.
		3h	Dead-band is fully enabled for both rising-edge delay on output EPWMxA and falling-edge delay on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].

3-2	POLSEL		Polarity Select Control. Bit 3 controls the S3 switch and bit 2 controls the S2 switch shown in Figure 35-28. This allows you to selectively invert one of the delayed signals before it is sent out of the dead-band submodule. The following descriptions correspond to classical upper/lower switch control as found in one leg of a digital motor control inverter. These assume that DBCTL[OUT_MODE] = 1,1 and DBCTL[IN_MODE] = 0,0. Other enhanced modes are also possible, but not regarded as typical usage modes.
		0	Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).
		1h	Active low complementary (ALC) mode. EPWMxA is inverted.
		2h	Active high complementary (AHC). EPWMxB is inverted.
		3h	Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.

```
/** - Set the rising edge delay  */
etpwmREG1->DBRED = 110U;

/** - Set the falling edge delay  */
etpwmREG1->DBFED = 110U ;
```

→ 각 Delay time 설정

Dead-band 라는게 상승 엣지와 하강 엣지에 각각 enable

위상 inverted 을 했다는 의미

(질문) Dead-band 가 뭘까요...? PWM 파형 상승 or 하강 엣지 동작 전 delay?

etPWM(w/UART) – code 및 내용 분석

```
for(;;)
{
    tmp = sciReceiveByte(UART);

    receive_data = tmp - 48;    // uart 값은 아스키 이므로 48을 빼서 숫자화

    sprintf(buf, "recv_data = %d\n\r\0", receive_data);
    buf_len = strlen(buf);
    sci_display_text(UART, (uint8 *)buf, buf_len);

    gioToggleBit(gioPORTA, 4);
    set_pwm();

    sprintf(buf, "PWM Duty = %d \n\r\0", pwm_value);
    buf_len = strlen(buf);
    sci_display_text(UART, (uint8 *)buf, buf_len);

    wait(5000000);
}
```

```
sprintf(buf, "Press Number!\n\r\0");
buf_len = strlen(buf);
sci_display_text(UART, (uint8 *)buf, buf_len);
```

sci_display_text 에서 입력한 UART 를

SciReceiveByte 의 인자로 받아서 tmp 에 담는다.

이 tmp 의 값은 아스키 문자 값으로 받은 것이므로 -48 을 해서 숫자 데이터화 해주어야 한다.

그렇게 숫자 데이터화 된 값을 buf 에 담고
Uart display 해준다.

GioToggleBit 는 해당 Port의 bit 값을 toggle 시켜 준다.

```
uint32 pwm_duty[] = { 500, 750, 1000, 1250, 1500, 1750, 2000, 2250, 2500 };
```

```
void set_pwm(void)
{
    pwm_value = pwm_duty[receive_data];
    etpwmSetCmpA(etpwmREG1, pwm_value);
    wait(100000);
}
```

```
void etpwmSetCmpA(etpwmBASE_t *etpwm, uint16 value)
{
    etpwm->CMPA = value;
}
```

Value 값 도착시 interrupt

Receive_data를 0~8의 값을 받을 건데,
그 배열의 값이 pwm_value로 들어가게 되고

EtpwmSetCmpA 를 통해 compare 방식 etpwm 동작을 하게 한다.

etPWM(w/UART) – code 및 내용 분석

HL_system.h

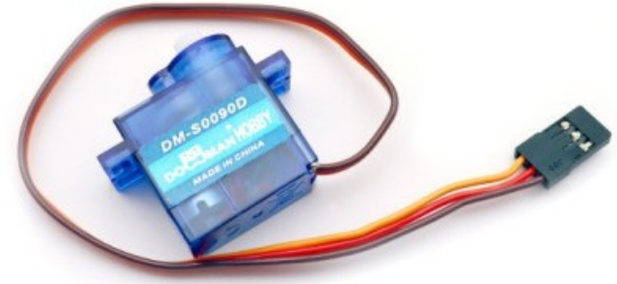
```
/dev/ttyUSB0 - PuTTY
PWM Duty = 2000
recv_data = 4
PWM Duty = 1500
recv_data = 5
PWM Duty = 1750
recv_data = 3
PWM Duty = 1250
recv_data = 4
PWM Duty = 1500
recv_data = 4
PWM Duty = 1500
recv_data = 1
PWM Duty = 750
recv_data = 1
PWM Duty = 750
recv_data = 1
PWM Duty = 750
recv_data = 3
PWM Duty = 1250
recv_data = 1
PWM Duty = 750
recv_data = 3
PWM Duty = 1250
```

좌측은 결과 창이며

입력하는 숫자에 따라

Duty 비를 다르게 되었고,

우측과 같은 서버모터 제어시 활용 되었다.



오실로스코프 파형

주기 18ms
Duty 값 확인

(주기 20ms로 설저하긴 했었는데,
18Ms 가 측정되는 이유는 모르겠다.)