


Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

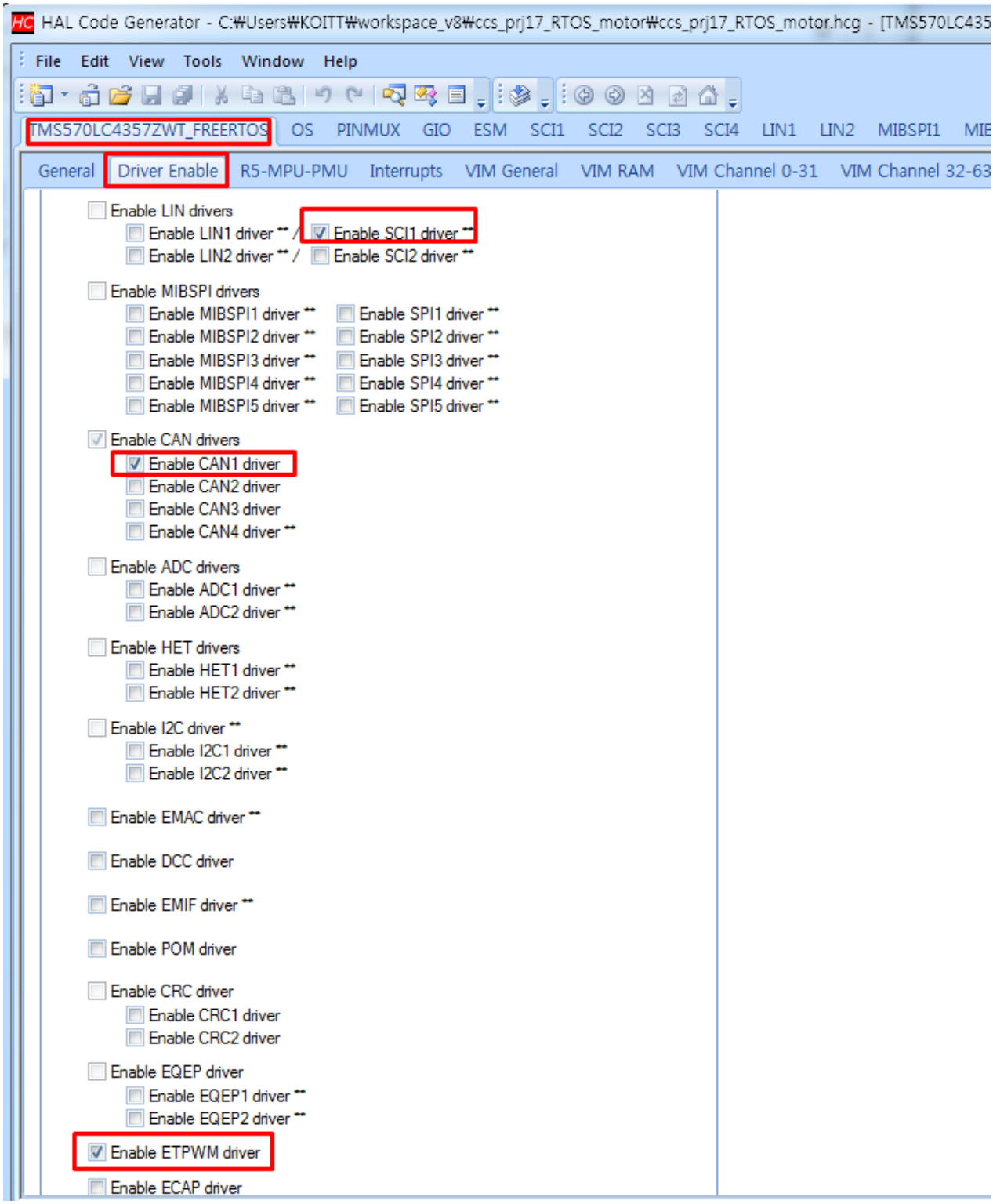
학생 – 정한별

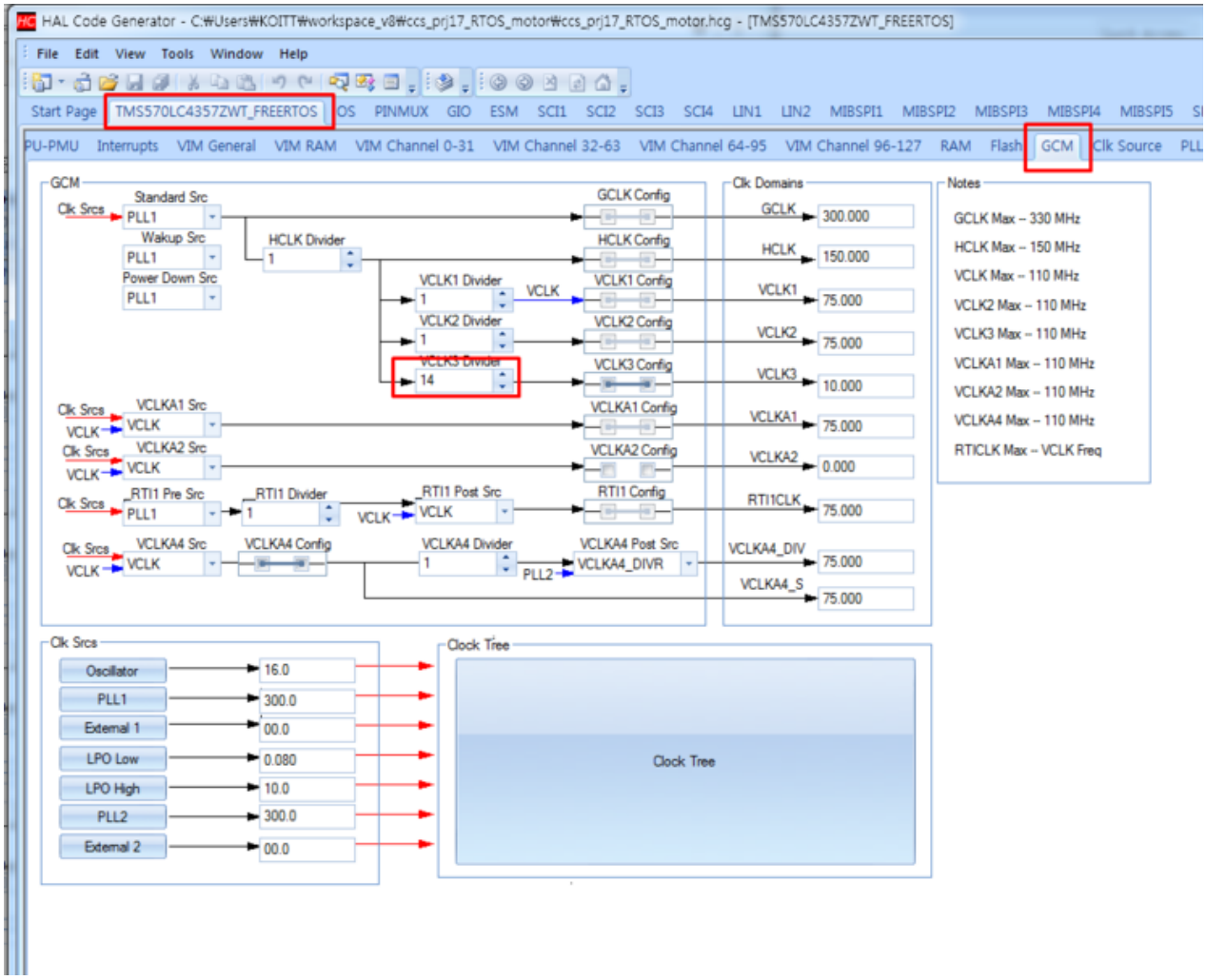
hanbulkr@gmail.com

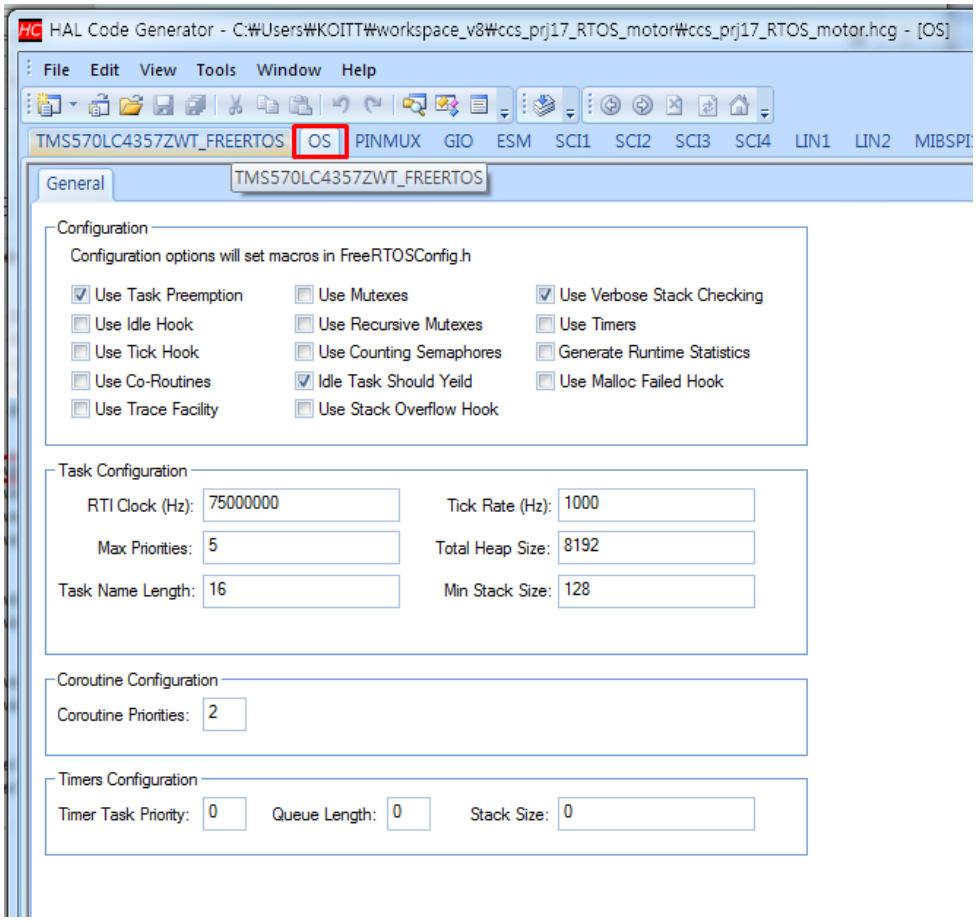
FREE_RTOS_CAN을 이용한 모터 제어 프로그래밍 2018. 8. 3. 12:08

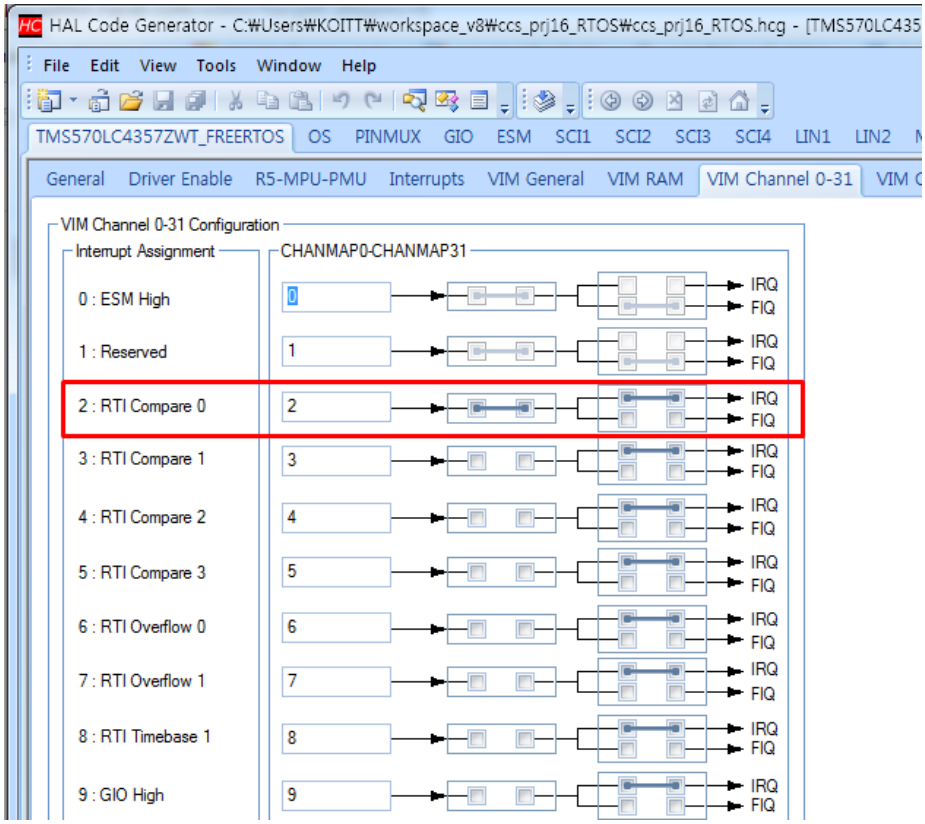
https://blog.naver.com/hanbulkr/221331871577

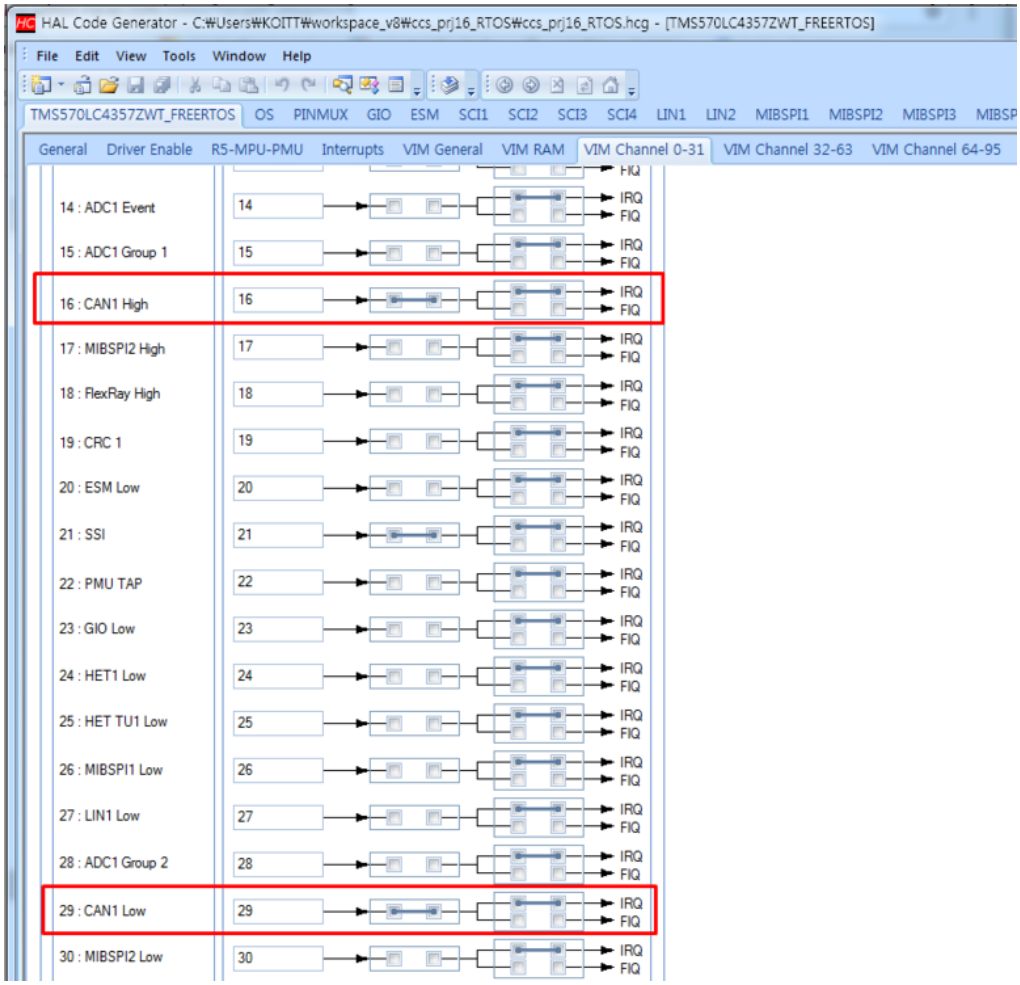
기본적인 내용은 위에서 설명 했으므로 바로 예제로 들어 가겠습니다.

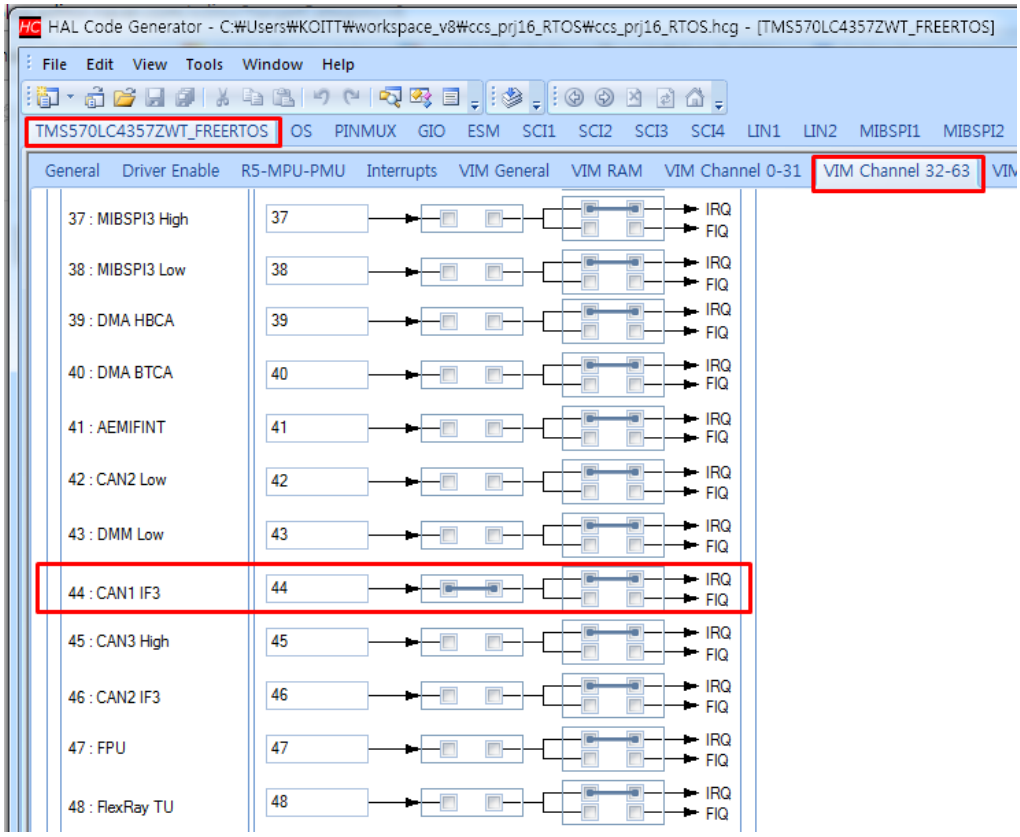


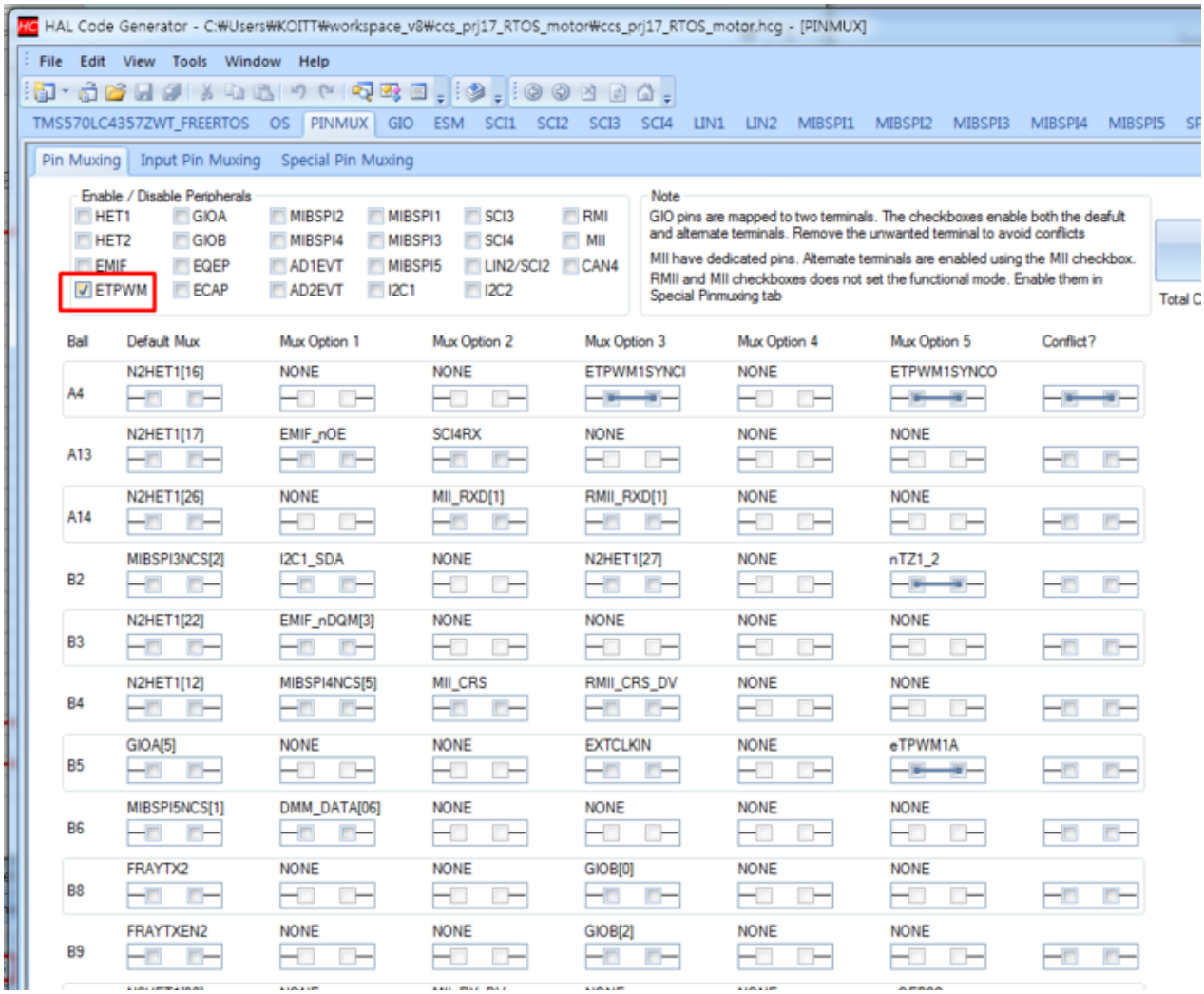

HAL Code Generator - C:#Users#KOITT#workspace_v8#ccs_prj17_RTOS_motor#ccs_prj17_RTOS_motor.hcg - [TMS570LC4357ZWT_FREERTOS]
File Edit View Tools Window Help
Start Page
TMS570LC4357ZWT_FREERTOS
OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5
PU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31 VIM Channel 32-63 VIM Channel 64-95 VIM Channel 96-127 RAM Flash GCM Clk Source PLL
GCM
Clk Srcs
Standard Src
PLL1
Wakup Src
PLL1
Power Down Src
PLL1
HCLK Divider
1
VCLK1 Divider
VCLK
VCLK2 Divider
VCLK3 Divider
14
GCLK Config
HCLK Config
VCLK1 Config
VCLK2 Config
VCLK3 Config
VCLKA1 Src
VCLKA1 Config
VCLKA2 Src
VCLKA2 Config
RTI1 Pre Src
RTI1 Divider
RTI1 Post Src
RTI1 Config
VCLKA4 Src
VCLKA4 Config
VCLKA4 Divider
VCLKA4 Post Src
VCLKA4_DIVR
PLL2
Clk Domains
GCLK 300.000
HCLK 150.000
VCLK1 75.000
VCLK2 75.000
VCLK3 10.000
VCLKA1 75.000
VCLKA2 0.000
RTI1CLK 75.000
VCLKA4_DIV 75.000
VCLKA4_S 75.000
Notes
GCLK Max -- 330 MHz
HCLK Max -- 150 MHz
VCLK Max -- 110 MHz
VCLK2 Max -- 110 MHz
VCLK3 Max -- 110 MHz
VCLKA1 Max -- 110 MHz
VCLKA2 Max -- 110 MHz
VCLKA4 Max -- 110 MHz
RTICLK Max -- VCLK Freq
Clk Srcs
Oscillator 16.0
PLL1 300.0
External 1 00.0
LPO Low 0.080
LPO High 10.0
PLL2 300.0
External 2 00.0
Clock Tree
Clock Tree

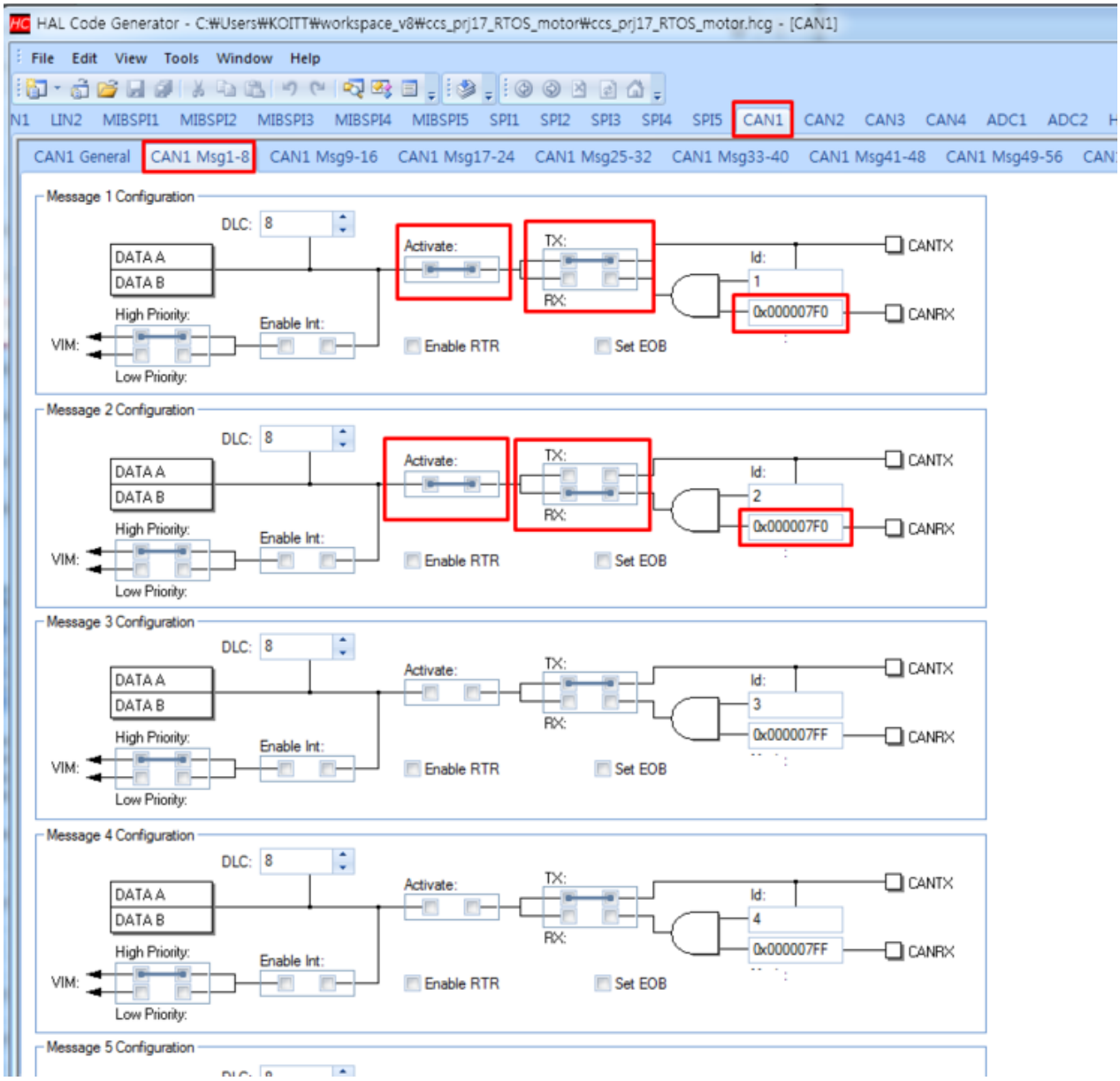

HAL Code Generator - C:#Users#KOITT#workspace_v8#ccs_prj17_RTOS_motor#ccs_prj17_RTOS_motor.hcg - [OS]
File Edit View Tools Window Help
TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI
General
TMS570LC4357ZWT_FREERTOS
Configuration
Configuration options will set macros in FreeRTOSConfig.h
Use Task Preemption
Use Idle Hook
Use Tick Hook
Use Co-Routines
Use Trace Facility
Use Mutexes
Use Recursive Mutexes
Use Counting Semaphores
Idle Task Should Yeild
Use Stack Overflow Hook
Use Verbose Stack Checking
Use Timers
Generate Runtime Statistics
Use Malloc Failed Hook
Task Configuration
RTI Clock (Hz): 75000000
Tick Rate (Hz): 1000
Max Priorities: 5
Total Heap Size: 8192
Task Name Length: 16
Min Stack Size: 128
Coroutine Configuration
Coroutine Priorities: 2
Timers Configuration
Timer Task Priority: 0
Queue Length: 0
Stack Size: 0

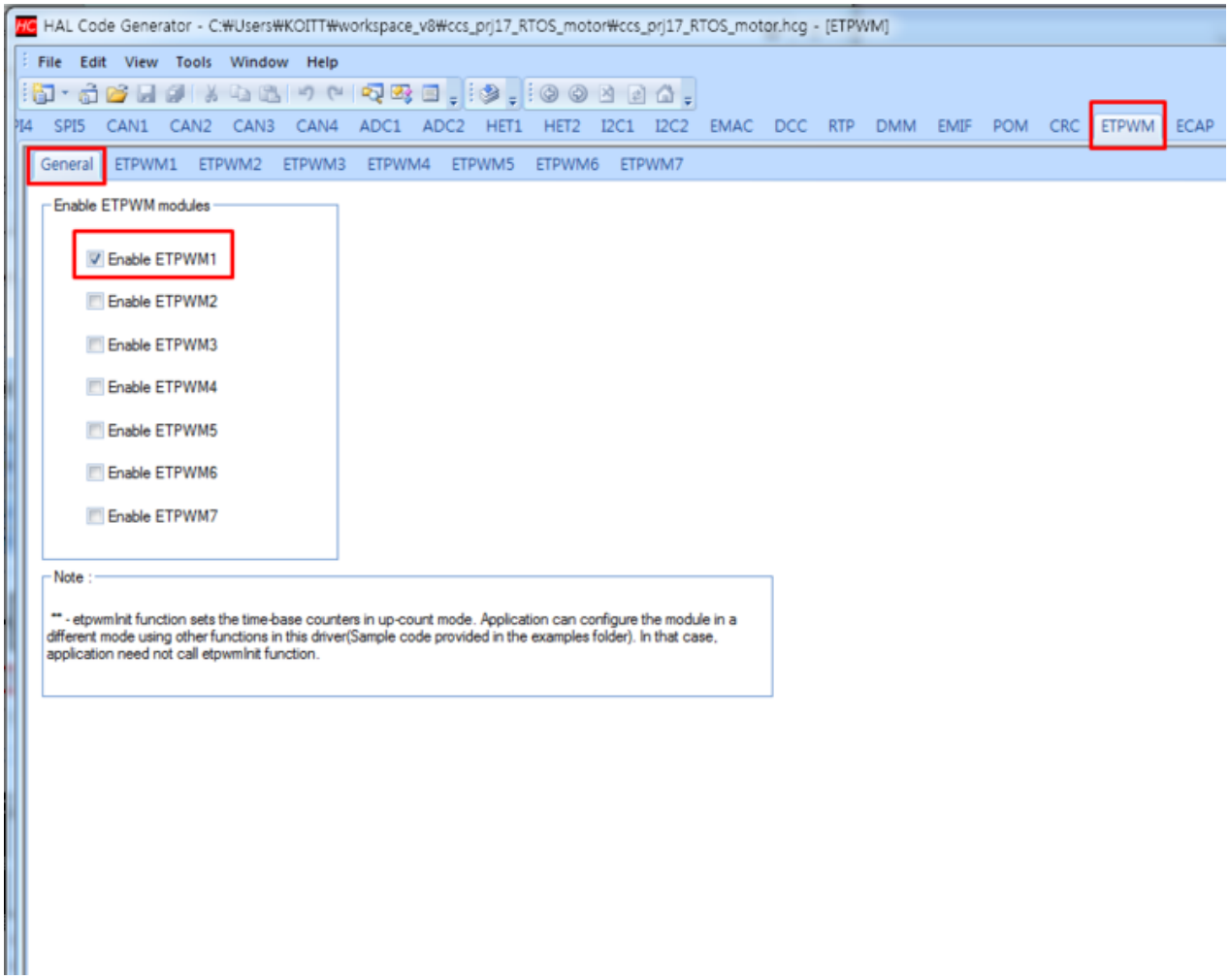

HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj16_RTOS₩ccs_prj16_RTOS.hcg - [TMS570LC435
File Edit View Tools Window Help
TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31
VIM Channel 0-31 Configuration
Interrupt Assignment
CHANMAP0-CHANMAP31
0 : ESM High
1 : Reserved
2 : RTI Compare 0
3 : RTI Compare 1
4 : RTI Compare 2
5 : RTI Compare 3
6 : RTI Overflow 0
7 : RTI Overflow 1
8 : RTI Timebase 1
9 : GIO High
IRQ
FIQ

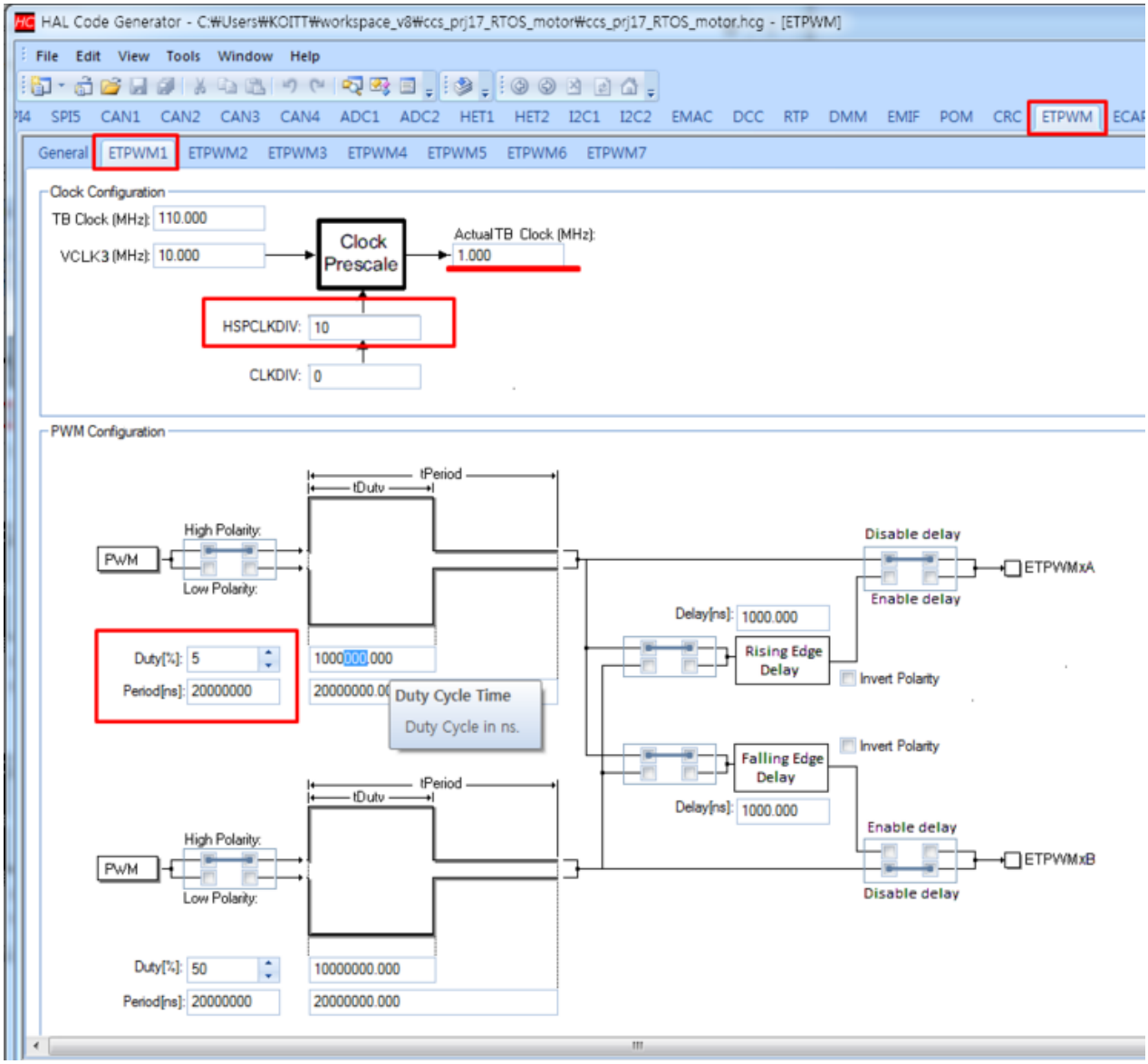

HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj16_RTOS₩ccs_prj16_RTOS.hcg - [TMS570LC4357ZWT_FREERTOS]
File Edit View Tools Window Help
TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31 VIM Channel 32-63 VIM Channel 64-95
14 : ADC1 Event
15 : ADC1 Group 1
16 : CAN1 High
17 : MIBSPI2 High
18 : FlexRay High
19 : CRC 1
20 : ESM Low
21 : SSI
22 : PMU TAP
23 : GIO Low
24 : HET1 Low
25 : HET TU1 Low
26 : MIBSPI1 Low
27 : LIN1 Low
28 : ADC1 Group 2
29 : CAN1 Low
30 : MIBSPI2 Low
IRQ
FIQ


HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj16_RTOS₩ccs_prj16_RTOS.hcg - [TMS570LC4357ZWT_FREERTOS]
File Edit View Tools Window Help
TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31 VIM Channel 32-63
37 : MIBSPI3 High
38 : MIBSPI3 Low
39 : DMA HBCA
40 : DMA BTCA
41 : AEMIFINT
42 : CAN2 Low
43 : DMM Low
44 : CAN1 IF3
45 : CAN3 High
46 : CAN2 IF3
47 : FPU
48 : FlexRay TU
IRQ
FIQ


HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj17_RTOS_motor₩ccs_prj17_RTOS_motor.hcg - [PINMUX]
File Edit View Tools Window Help
TMS570LC4357ZWT_FREERTOS OS PINMUX GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5
Pin Muxing Input Pin Muxing Special Pin Muxing
Enable / Disable Peripherals
HET1 HET2 EMIF ETPWM
GIOA GIOB EQEP ECAP
MIBSPI2 MIBSPI4 AD1EVT AD2EVT
MIBSPI1 MIBSPI3 MIBSPI5 I2C1
SCI3 SCI4 LIN2/SCI2 I2C2
RMI MII CAN4
Note
GIO pins are mapped to two terminals. The checkboxes enable both the deafult and alternate terminals. Remove the unwanted terminal to avoid conflicts
MII have dedicated pins. Alternate terminals are enabled using the MII checkbox.
RMII and MII checkboxes does not set the functional mode. Enable them in Special Pinmuxing tab
Ball Default Mux Mux Option 1 Mux Option 2 Mux Option 3 Mux Option 4 Mux Option 5 Conflict?
A4 N2HET1[16] NONE NONE ETPWM1SYNCI NONE ETPWM1SYNCO
A13 N2HET1[17] EMIF_nOE SCI4RX NONE NONE NONE
A14 N2HET1[26] NONE MII_RXD[1] RMII_RXD[1] NONE NONE
B2 MIBSPI3NCS[2] I2C1_SDA NONE N2HET1[27] NONE nTZ1_2
B3 N2HET1[22] EMIF_nDQM[3] NONE NONE NONE NONE
B4 N2HET1[12] MIBSPI4NCS[5] MII_CRS RMII_CRS_DV NONE NONE
B5 GIOA[5] NONE NONE EXTCLKIN NONE eTPWM1A
B6 MIBSPI5NCS[1] DMM_DATA[06] NONE NONE NONE NONE
B8 FRAYTX2 NONE NONE GIOB[0] NONE NONE
B9 FRAYTXEN2 NONE NONE GIOB[2] NONE NONE


HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj17_RTOS_motor₩ccs_prj17_RTOS_motor.hcg - [CAN1]
File Edit View Tools Window Help
N1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1 SPI2 SPI3 SPI4 SPI5 CAN1 CAN2 CAN3 CAN4 ADC1 ADC2
CAN1 General CAN1 Msg1-8 CAN1 Msg9-16 CAN1 Msg17-24 CAN1 Msg25-32 CAN1 Msg33-40 CAN1 Msg41-48 CAN1 Msg49-56
Message 1 Configuration
DLC: 8
DATA A
DATA B
High Priority:
VIM:
Low Priority:
Enable Int:
Activate:
TX:
RX:
Id:
1
0x000007F0
CANTX
CANRX
Enable RTR
Set EOB
Message 2 Configuration
DLC: 8
DATA A
DATA B
High Priority:
VIM:
Low Priority:
Enable Int:
Activate:
TX:
RX:
Id:
2
0x000007F0
CANTX
CANRX
Enable RTR
Set EOB
Message 3 Configuration
DLC: 8
DATA A
DATA B
High Priority:
VIM:
Low Priority:
Enable Int:
Activate:
TX:
RX:
Id:
3
0x000007FF
CANTX
CANRX
Enable RTR
Set EOB
Message 4 Configuration
DLC: 8
DATA A
DATA B
High Priority:
VIM:
Low Priority:
Enable Int:
Activate:
TX:
RX:
Id:
4
0x000007FF
CANTX
CANRX
Enable RTR
Set EOB
Message 5 Configuration


HAL Code Generator - C:₩Users₩KOITT₩workspace_v8₩ccs_prj17_RTOS_motor₩ccs_prj17_RTOS_motor.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
General ETPWM1 ETPWM2 ETPWM3 ETPWM4 ETPWM5 ETPWM6 ETPWM7
Enable ETPWM modules
Enable ETPWM1
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.



<code>

servo 모터 255단계(0~FF) 까지 가능하다.)

```
#include <FreeRTOS.h>
#include <FreeRTOSConfig.h>
#include <HL_hal_stdtypes.h>
#include <HL_reg_sci.h>
#include <HL_sci.h>
#include <os_mpu_wrappers.h>
#include <os_projdefs.h>
#include <os_semphr.h>
#include <os_task.h>
#include "string.h"
#include "stdio.h"
#include "stdlib.h"
#include "HL_can.h"
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_esm.h"
#include "HL_sys_core.h"
#include "HL_etpwm.h"
#include <string.h>
```

```
#define D_COUNT 8
#define D_SIZE 8

xTaskHandle xTask1Handle;
xTaskHandle xTask2Handle;
xTaskHandle xTask3Handle;

QueueHandle_t mutex = NULL;
QueueHandle_t mutex1 = NULL;

char flag = 0;
uint32 cnt = 0;
uint32 error = 0;
uint32 tx_done = 0;

uint8 msg[] = { 0 };
uint8 msg1[] = { 0 };
uint8 msg2[32] = { 'T', 'a', 's', 'k', '2', '\r', '\n' };
uint8 msg3[] = { 0 };
uint8 tx_data[D_COUNT] = { 1, 2, 3, 4, 4, 3, 2, 1 };
uint8 rx_data[D_COUNT] = { 0 };

void send_data(sciBASE_t* sci, uint8* msg, int length);
void delay(uint32 time);
void active_motor(int verocity);
void convert_rx_data(void);
void print_rx_data(void);
void vTask1(void *pbParameters);
void vTask2(void *pbParameters);
void vTask3(void *pbParameters);
void task_create(void);

void delay(uint32 time) {
    while (time--)
        ;
}

void active_motor(int verocity) {
    sciSend(sciREG1, 17, " convert data \r\n");

    if (rx_data[0] + 48 == '1') {
        sciSend(sciREG1, 25, "\r\n Motor On Forward \r\n\r\n");
        etpwmREG1->CMPA = verocity;
    }
    else if (rx_data[0] == 2) {

        sciSend(sciREG1, 25, "\r\n Motor On Backward \r\n\r\n");

        // 중간에 StartTBCLK를 키고 끄면 멈춘다.
        // etpwmStartTBCLK();
        etpwmREG1->CMPA = verocity;
        delay(5);
    }
    else if (rx_data[0] + 48 == '0') {
```

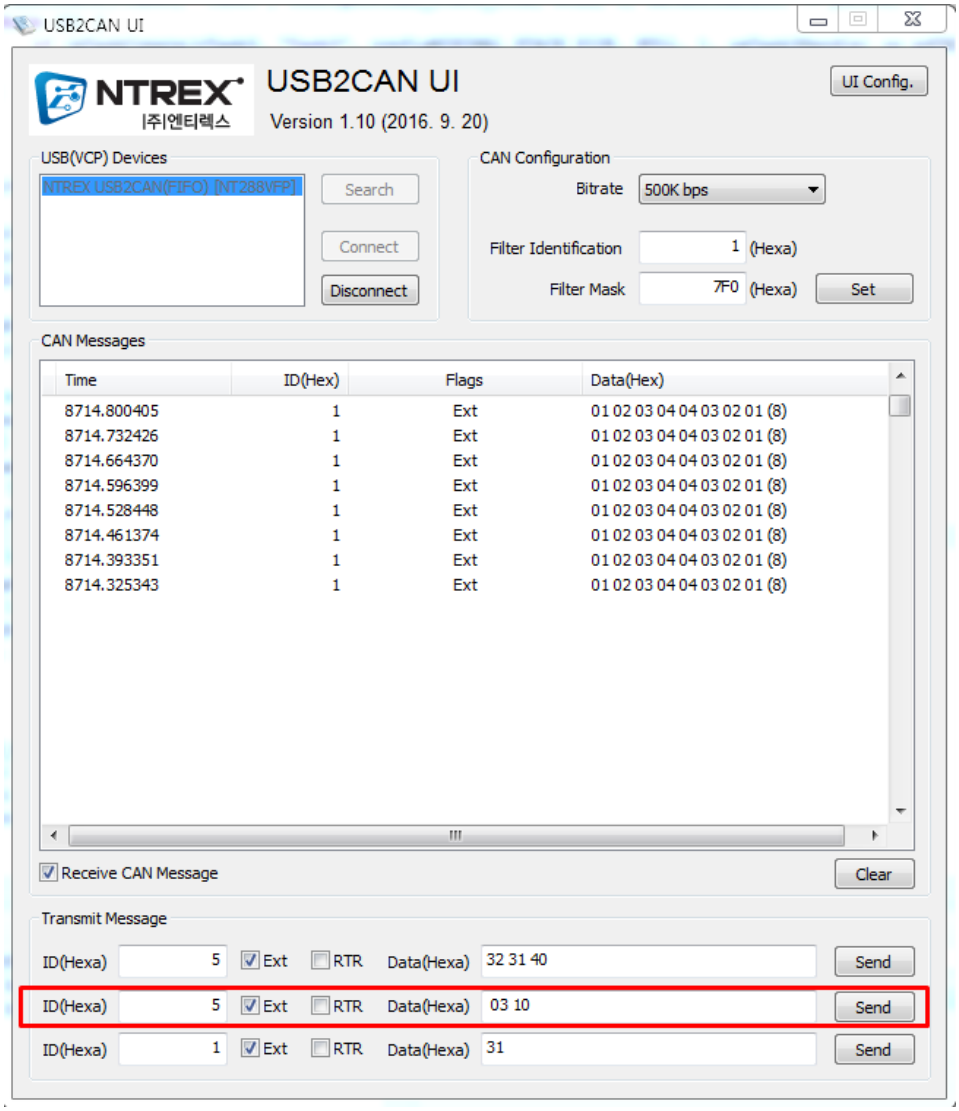
```
        sciSend(sciREG1, 25, "\r\n Motor Off \r\n\r\n");
        etpwmREG1->CMPA = 0;
        delay(5);
    }
    else
        ;
}

void convert_rx_data(void) {
//  char x[10]={0};
    int i;
    int convert_verocity = 0U;
    // 여기서 컨버팅 할 때 값이 atol을 쓰면 멈춰 버린다.
    convert_verocity = (rx_data[1] * 4) + 1000;
    // debug
    //  ltoa(convert_verocity, x);
    //  sciSend(sciREG1, strlen((const char *) x),(uint8 *) x);
    etpwmREG1->CMPA = convert_verocity;

    // 들어온 값에 따라 모터의 동작을 한다.
    active_motor(convert_verocity);
    delay(5);

    for (i = 0; i < sizeof(rx_data); i++) {
        rx_data[i] = '\0';
    }
}

void print_rx_data(void) {
    //(0 bit)DIR(0:멈춤, 1:앞으로, 2:뒤로), (1 bit)속도 조절 (단계:HEX(0~FF))

    char ID1[2] = {0};
    char ID2[2] = {0};
    char speed[10]={0};
    int convert_verocity = 0U;

    convert_verocity = (rx_data[1] * 4) + 1000;
    ltoa(convert_verocity, speed);

    ltoa(canGetID(canREG1,canMESSAGE_BOX2), ID1);
    ltoa(canGetID(canREG1,canMESSAGE_BOX1), ID2);

    sciSend(sciREG1, strlen((const char *) "ID : "), "ID : ");
    sciSend(sciREG1, strlen((const char *) ID1),(uint8 *) ID1);
    sciSend(sciREG1, strlen((const char *) "->"), "->");
    sciSend(sciREG1, strlen((const char *) ID2),(uint8 *) ID2);
    sciSend(sciREG1, strlen((const char *) "\r\n"), "\r\n");

    sciSend(sciREG1, strlen((const char *) "mode(0:stop 1:forward 2:backward): "), "mode(0:stop 1:forward -1:backward)
    sciSendByte(sciREG1, rx_data[0] + 48);
    sciSend(sciREG1, strlen((const char *) "\r\n"), "\r\n");

    sciSend(sciREG1, strlen((const char *) "speed_level: "), "speed_level: ");
    sciSend(sciREG1, strlen((const char *) speed),(uint8 *) speed);
    sciSend(sciREG1, strlen((const char *) "\r\n\r\n"), "\r\n\r\n");
```

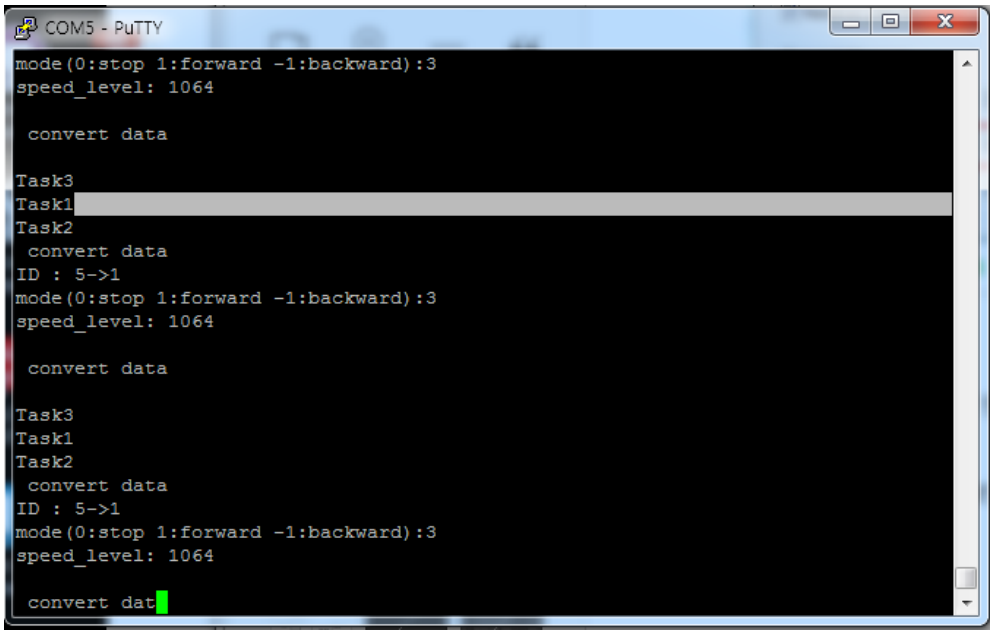
```
}

// 하나 하나가 메인이 라고 보면 된다. Task 마다 for문이 있는데 안 넣어주면 한번 동작을 하고 멈춘다.
// 함수의 들어가는 파라미터는 create 할때 지정해 준다?
void vTask1(void *pbParameters) {
    uint8 msg[32] = { 'T', 'a', 's', 'k', '1', '\r', '\n' };

    for (;;) {
        if (flag == 0) {
            if (xSemaphoreTake(mutex, ( TickType_t ) 40) == pdTRUE) {

                send_data(sciREG1, msg, strlen((const char *) msg));
                xSemaphoreGive(mutex);
                flag = 1;
                vTaskDelay(40);
            }
            else {
                // 키 값을 받지 못했을 경우 오류 처리.
            }
        }
        else {
            // flag 값이 다를 경우 동작 처리.
        }
    }
}

void vTask2(void *pbParameters) {

    for (;;) {
        if (flag == 1) {
            if (xSemaphoreTake(mutex, (TickType_t)40) == pdTRUE) {
                send_data(sciREG1, msg2, strlen((const char *) msg2));

                if (canIsRxMessageArrived(canREG1, canMESSAGE_BOX2)) {
                    canGetData(canREG1, canMESSAGE_BOX2, rx_data);

                    // +48을 해주어야 한다.
                    sciSend(sciREG1, strlen((const char *) rx_data), rx_data);
                    sciSend(sciREG1, 17, " convert data \r\n");

                    // 들어온 값을 문
                    print_rx_data();
                    convert_rx_data();
                }
                vTaskDelay(40);
                flag = 2;
                xSemaphoreGive(mutex);
            }
            else {
                // 키 값을 받지 못했을 경우 오류 처리.
                xSemaphoreGive(mutex);
                vTaskDelay(40);
            }
        }
```

```
        else {
            // flag 값이 다를 경우 동작 처리.
        }
    }
}
```

```
void vTask3(void *pbParameters) {
    uint8 msg[32] = { '\n', '\r', 'T', 'a', 's', 'k', '3', '\r', '\n' };
    for (;;) {
        if (flag == 2) {
            if (xSemaphoreTake(mutex, (TickType_t)40) == pdTRUE) {

                send_data(sciREG1, msg, strlen((const char *) msg));
                canTransmit(canREG1, canMESSAGE_BOX1, (const uint8 *) &tx_data[0]);
                // 키 값을 반납 한다.
                xSemaphoreGive(mutex);
                flag = 0;

                vTaskDelay(40);
            }
            else {
                // 키 값을 받지 못했을 경우 오류 처리.
            }
        }
        else {
            // flag 값이 다를 경우 동작 처리.
        }
    }
}

void task_create(void) {
    if (xTaskCreate(vTask1, "Task1", configMINIMAL_STACK_SIZE, NULL, 1, &xTask1Handle) != pdTRUE) {
        while (1) {
            if (xTaskCreate(vTask1, "Task1", configMINIMAL_STACK_SIZE, NULL, 1, &xTask1Handle) == pdTRUE)
                break;
        }
    }

    if (xTaskCreate(vTask2, "Task2", configMINIMAL_STACK_SIZE, NULL, 1, &xTask2Handle) != pdTRUE) {
        while (1) {
            if (xTaskCreate(vTask2, "Task2", configMINIMAL_STACK_SIZE, NULL, 1, &xTask2Handle) == pdTRUE)
                break;
        }
    }

    if (xTaskCreate(vTask3, "Task3", configMINIMAL_STACK_SIZE, NULL, 1, &xTask3Handle) != pdTRUE) {
        while (1) {
            if (xTaskCreate(vTask3, "Task3", configMINIMAL_STACK_SIZE, NULL, 1, &xTask3Handle) == pdTRUE)
                break;
        }
    }
}
```

```
void main(void) {

    sciInit();
    canInit();
    etpwmInit();

    etpwmStartTBCLK();
    delay(10);

// start message
    sciSend(sciREG1, 7, "start\n\r");
//이 기능은 오류 수준에 대한 경고 기능을 활성화 합니다.
    canEnableErrorNotification(canREG1);
// task를 만드는 함수.
    task_create();

// 뮤텍스 키 값을 이용해 현재 동작 중인 테스크를 고정 시켜주는 녀석이다. 콘텍스트 스위치 역할이랑 비슷하다.
    vSemaphoreCreateBinary(mutex);
// 위에서 Create로  만들어 놓은 Task들을  우선순위를 고려한 방식으로 스케쥴링 되어 동작하게 한다.
    vTaskStartScheduler();
    while (1) {

    }
}
void send_data(sciBASE_t* sci, uint8* msg, int length) {
    int i;
    for (i = 0; i < length; i++)
        sciSendByte(sci, msg[i]);
}
```





〈명령 방식〉

핵스 코드 2개를 날리면 된다.

그러면 sci(uart) 에서 보낸 곳의 ID 그리고 mode 스피드 레벨을 출력해 준다.

명령어 1byte 1byte
(mode) (speed)