



E D D I
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv2 과정

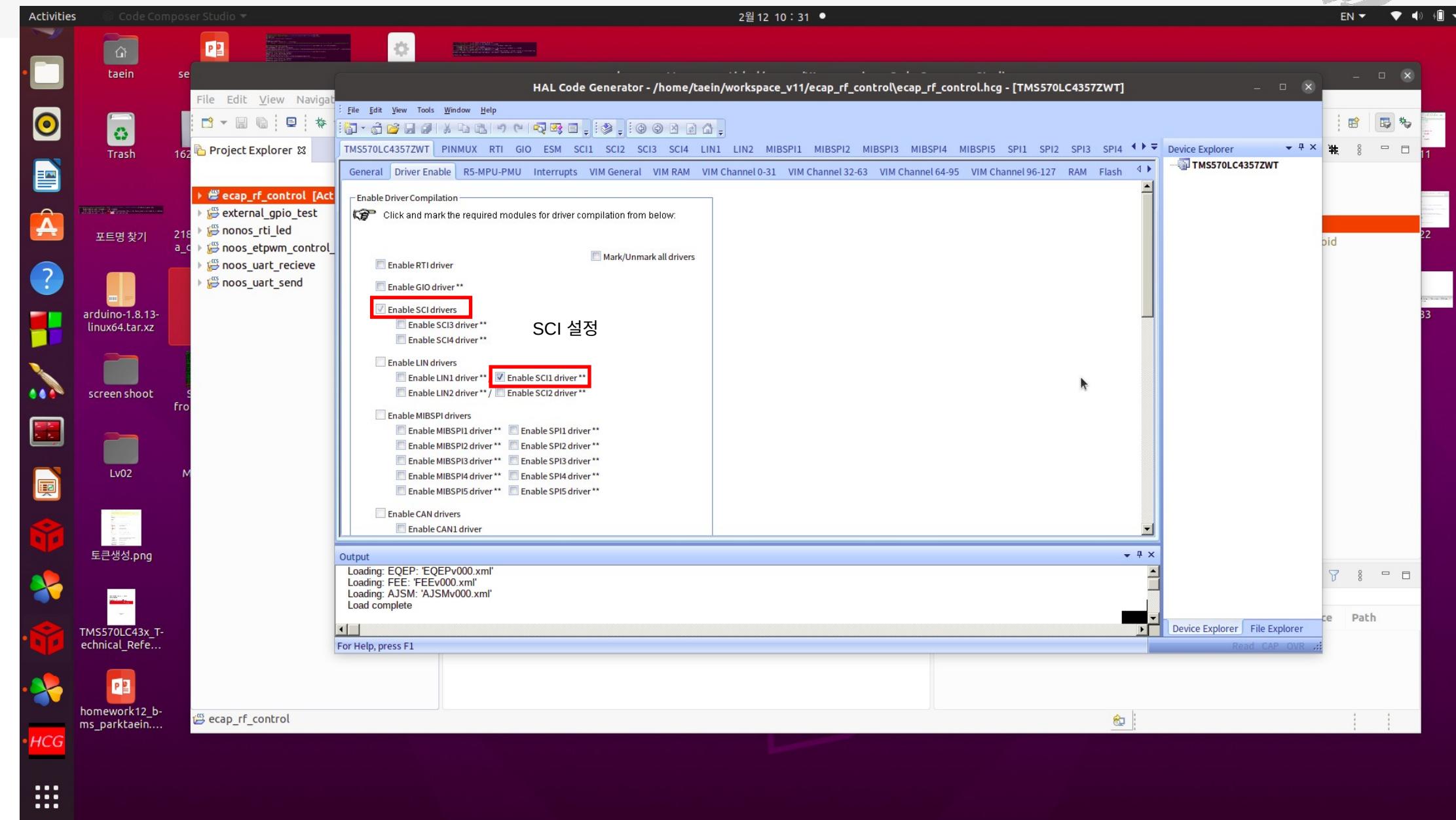
[TMS 570_eCAP(RF_Control)]

제 1기

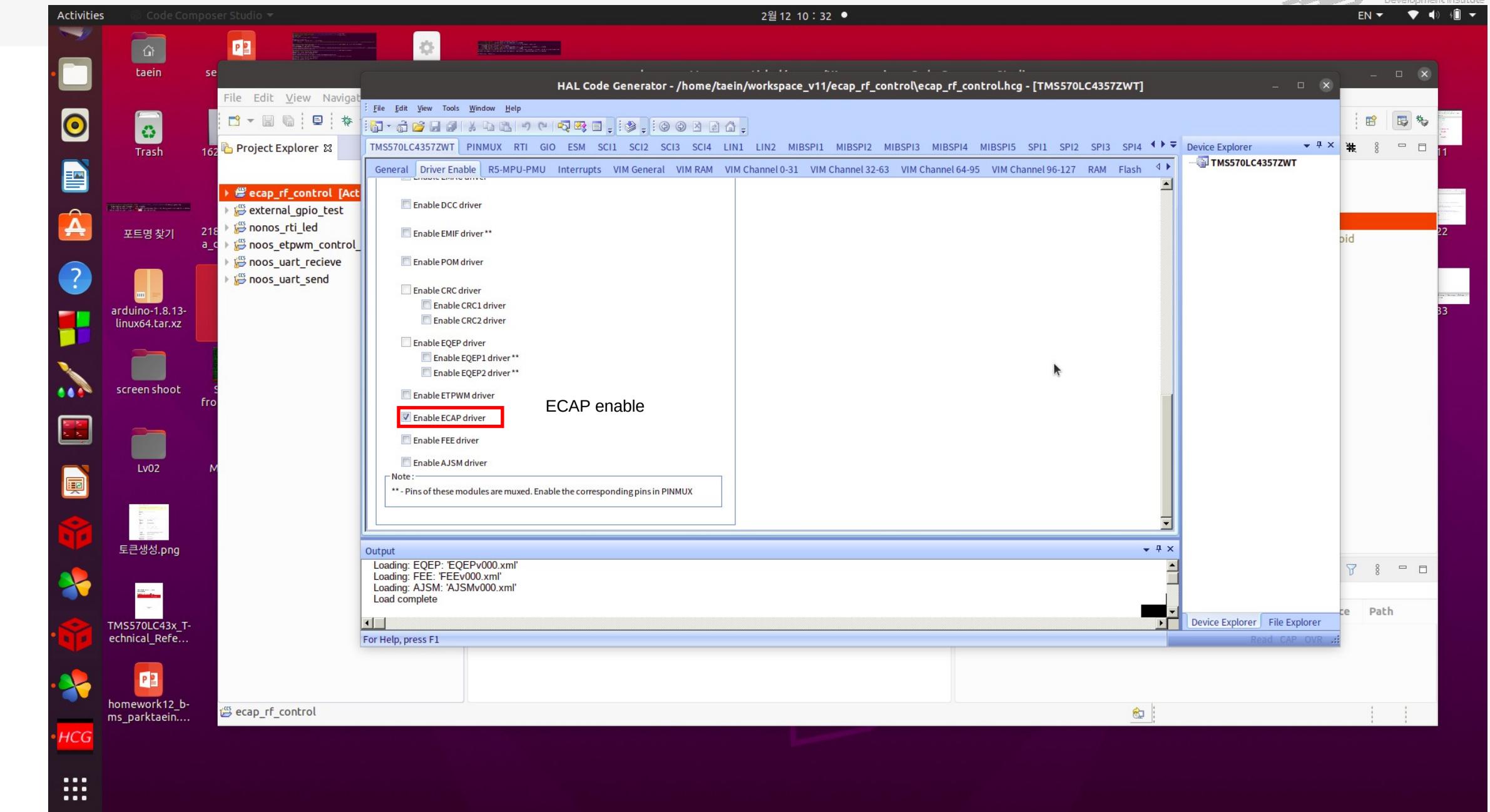
2022. 03. 18

박태인

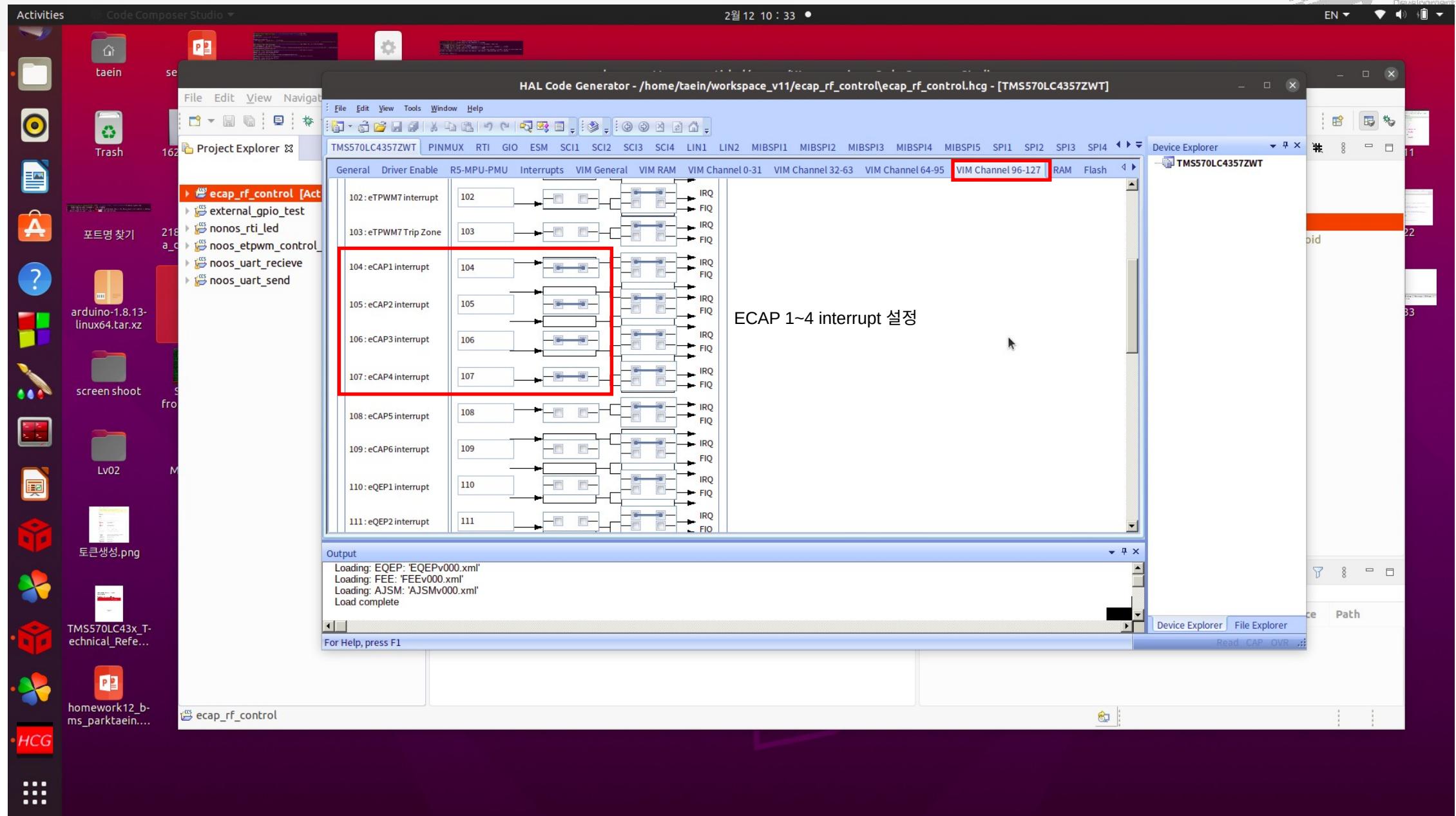
eCAP(RF_Control)



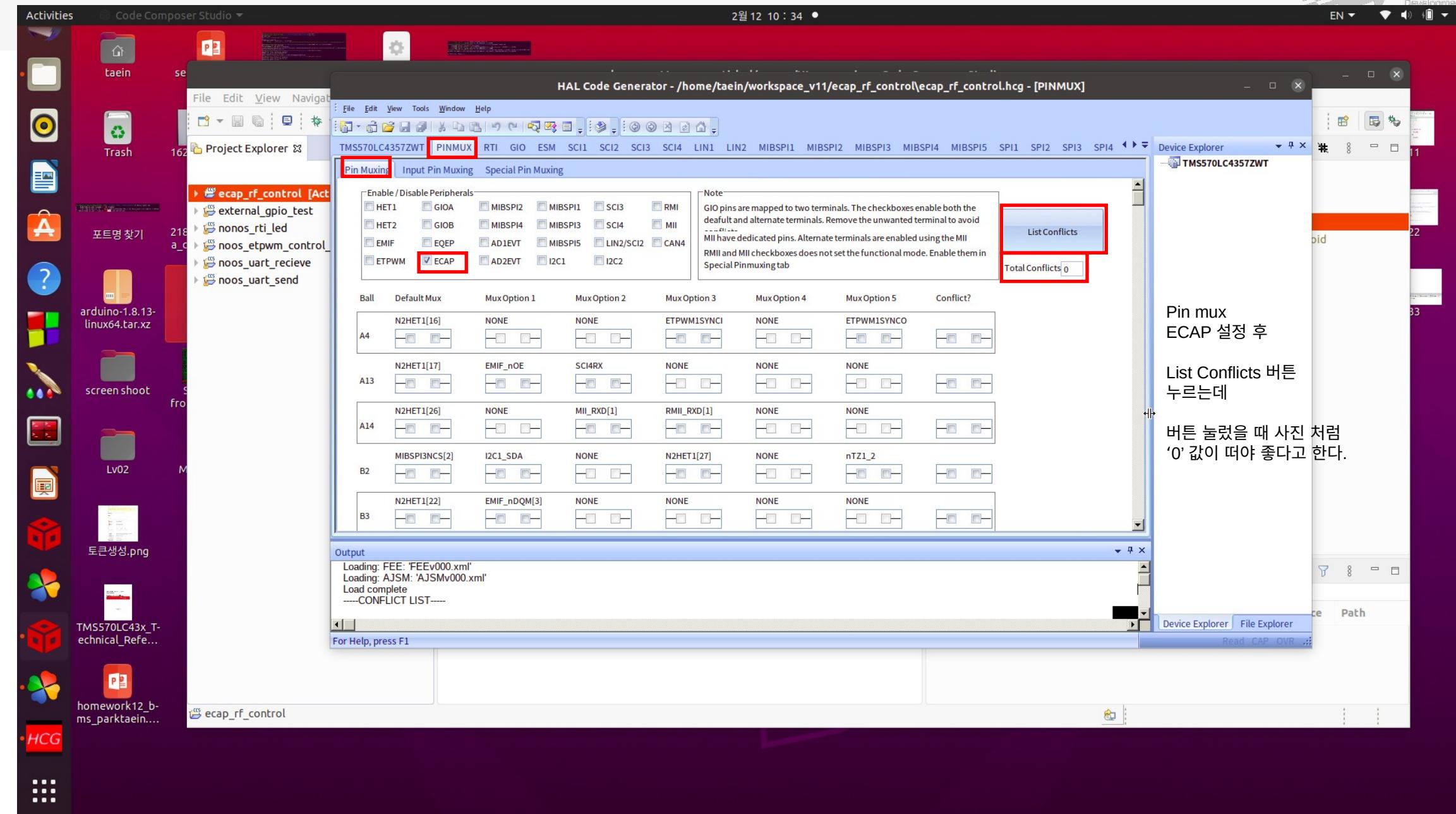
eCAP(RF_Control)



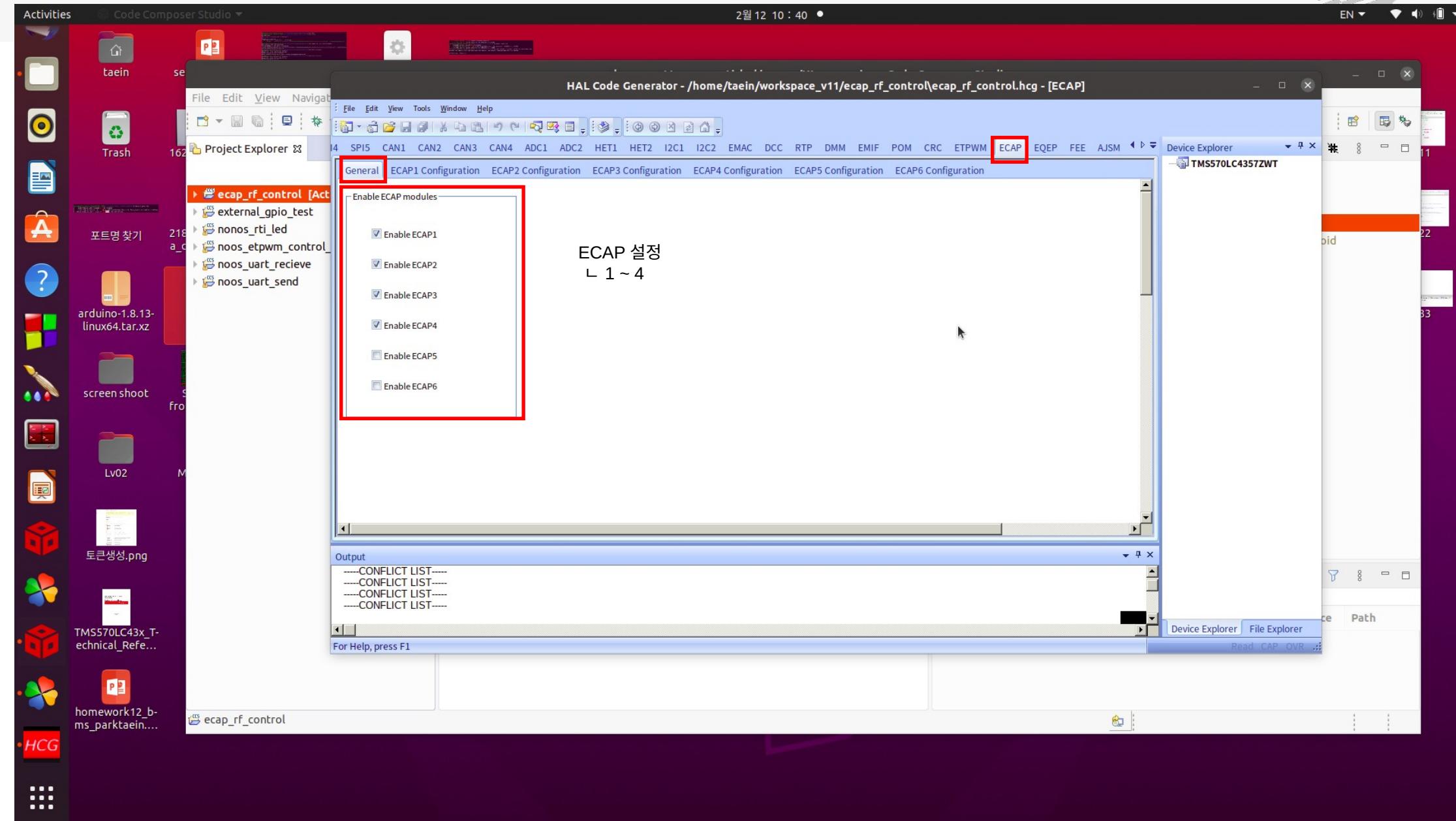
eCAP(RF_Control)



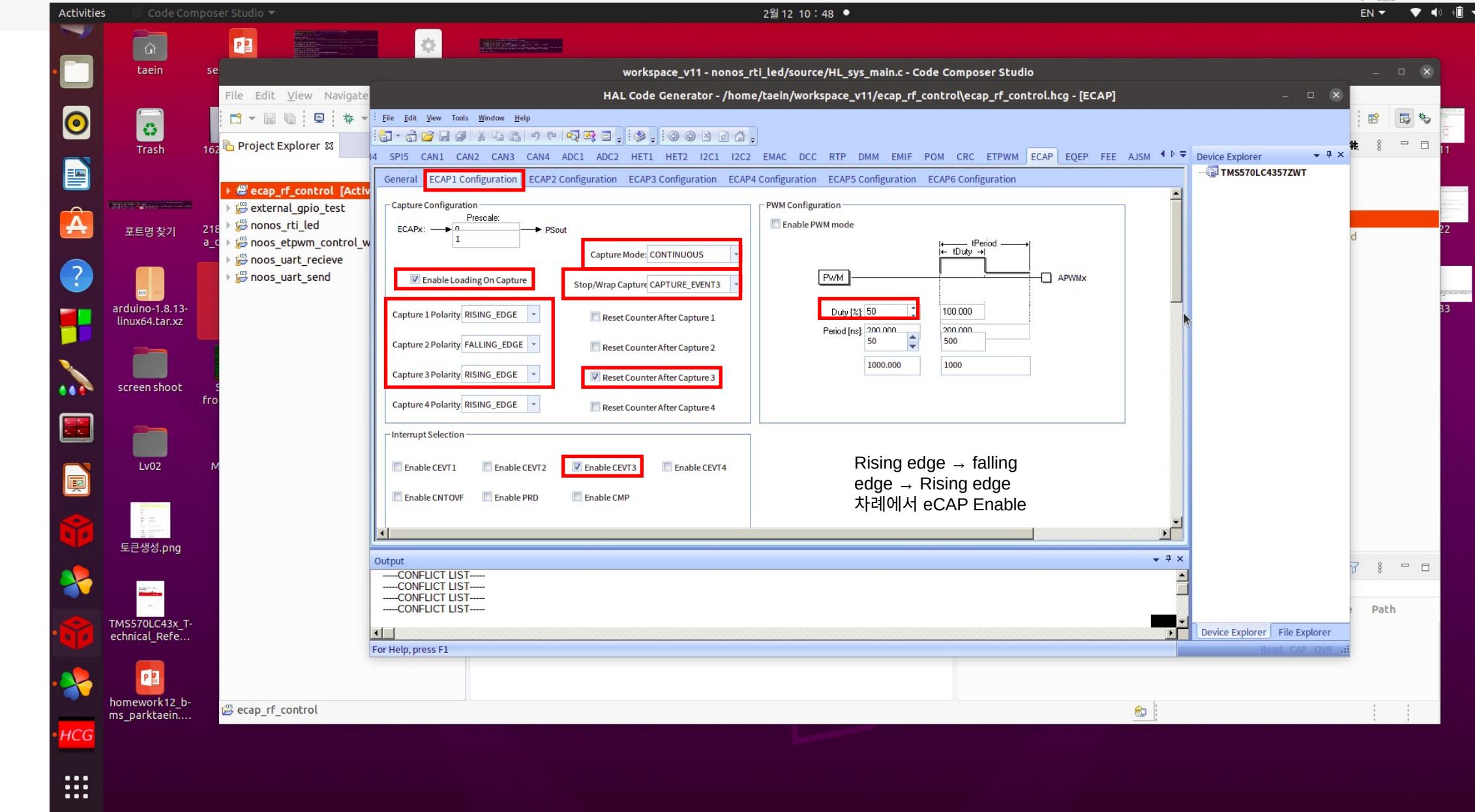
eCAP(RF_Control)



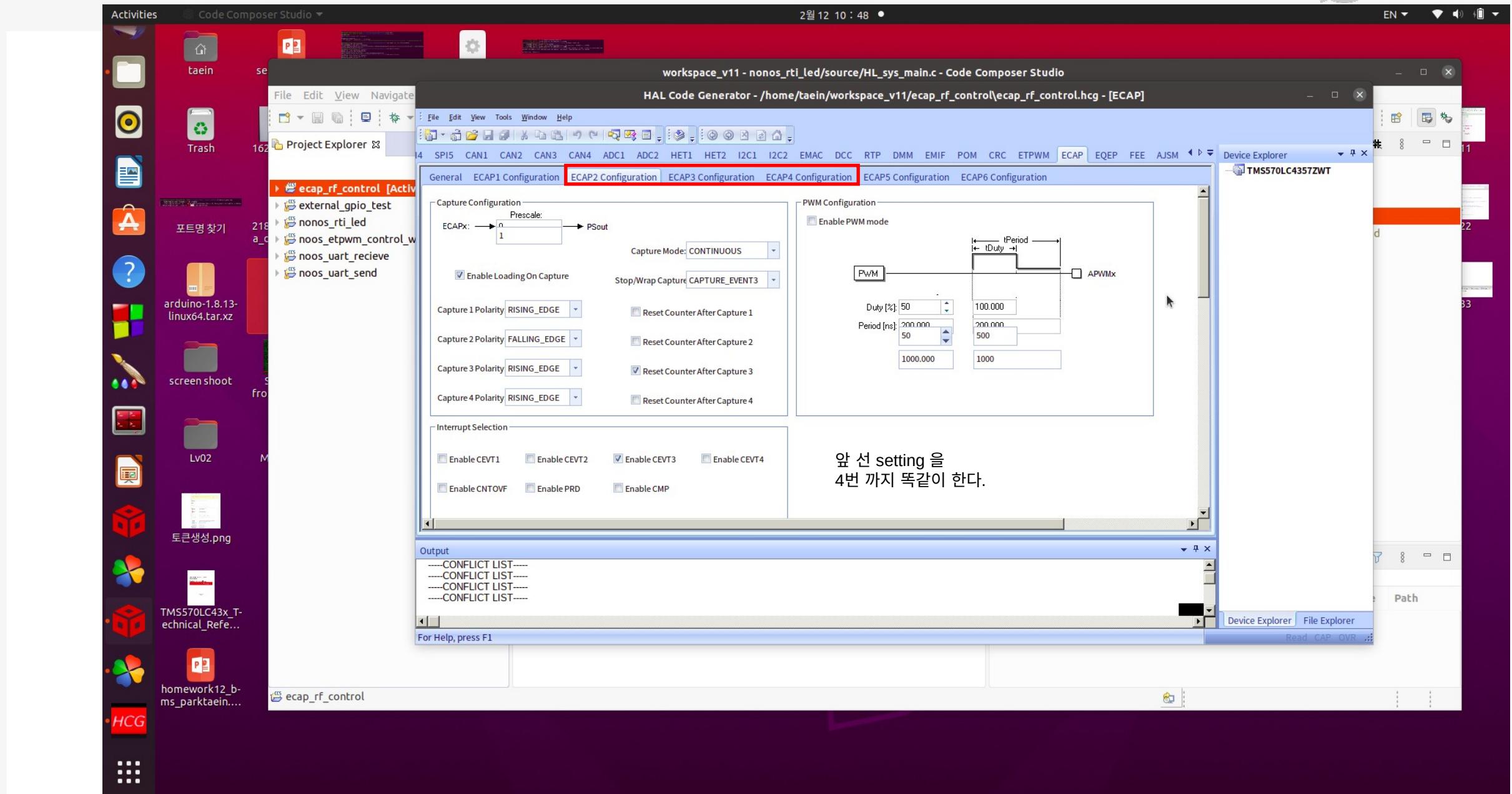
eCAP(RF_Control)



eCAP(RF_Control)



eCAP(RF_Control)



eCAP(RF_Control)

Activities Code Composer Studio ▾ 2월 12 16 : 12 workspace_v11 - ecap_rf_control/source/HL_sys_main.c - Code Composer Studio EN ▾

File Edit View Navigate Project Run Scripts Window Help

Project Explorer ▾ ecap_rf_control [Active - Debug] HL_sys_main.c

- Binaries
- Includes
- Debug
- include
- source
 - HL_ecap.c
 - HL_epc.c
 - HL_errata_SSWF021_45.c
 - HL_errata.c
 - HL_esm.c
 - HL_nmpu.c
 - HL_notification.c
 - HL_pinmux.c
 - HL_sci.c
 - HL_sys_core.asm
 - HL_sys_dma.c
 - HL_sys_intvecs.asm
 - HL_sys_link.cmd
 - HL_sys_main.c
 - HL_sys_mpu.asm
 - HL_sys_pcr.c
 - HL_sys_phantom.c
 - HL_sys_pmm.c
 - HL_sys_pmu.asm
 - HL_sys_startup.c
 - HL_sys_vim.c
 - HL_system.c
- targetConfigs
 - ecap_rf_control.dll
 - ecap_rf_control.hcg
- external_gpio_test
- nonos_rti_led
- noos_etpwm_control_with_uart
 - Binaries
 - Includes
 - Debug
 - include
 - SOURCE

Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: /home/taein/eddi_embedded_mster/EmbeddedMasterLv2/LSH/RC_control

File → Import 를 통해서 가져 옴

Import as: Eclipse project

Folder: RC_controller_ecap

1 of 2 selected

Select All Deselect All

Close newly imported projects upon completion

Use installed project configurators to:

Search for nested projects

Detect and configure project natures

Add project to working sets

Working sets: New... Select...

ECAP 동작시 코드 문제인지 보려고,
위의 경로에서 코드를
가져온 것.

Show other specialized import wizards

< Back Next > Cancel Finish

Console ▾ ecap_rf_control

CortexR5: GEL Output: Memory Map Setup for Flash @ Address 0x0CortexR5: GEL Output:
CortexR5: GEL Output: Memory Map Setup for Flash @ Address 0x0 due to System Reset

Problems Advice Memory Allocation Stack Usage

3 items

Description Resource Path Location

i Optimization Advice (3 items)

Writable Smart Insert 103 : 1 : 2985

eCAP(RF_Control)

ECAP 파트 부터는 선생님이 따로 정리 해주는 문서가 있어서 그 문서 내용을 이해하면서 더 조사가 필요한 부분은 추가하는 방향으로 복습 하도록 한다.

<https://cafe.naver.com/eddicorp/>

392 >

[링크쌤 칼럼] TMS570 eCAP 분석

링크쌤 카페매니저 M 1:1 채팅

2022.02.18. 13:49 조회 21

안녕하세요

에디로봇아카데미의 링크쌤입니다.

이번에 RF 수신기의 신호를 감지하기 위해 활용했던 eCAP을 파악하기 위해 봐야하는 코드는 아래와 같습니다.

ecapInit(), ecapStartCounter(), ecapEnableCapture(), ecapGetCAP1(), ecapGetCAP2()에 해당합니다.

먼저 스코프상에 잡히는 파형을 통해 이를 분석하도록 합니다.

먼저 스코프상에 잡히는 파형을 통해 이를 분석하도록 합니다.



실제 출력 결과를 보면 18ms의 주기가 잡히고 있으며 파형에서 잡히는 Duty는 8.333%임이 확인됩니다.
이와 같은 정보를 토대로 결과를 계산해 볼 수 있습니다.

eCAP(RF_Control)

실제로 화면상에 출력되는 결과에서도 이를 확인할 수 있습니다.

(18 * 10^-3) * 8.333 / 100

전체 쇼핑 이미지 지도 뉴스 더보기 도구

검색결과 약 0개 (0.81초)

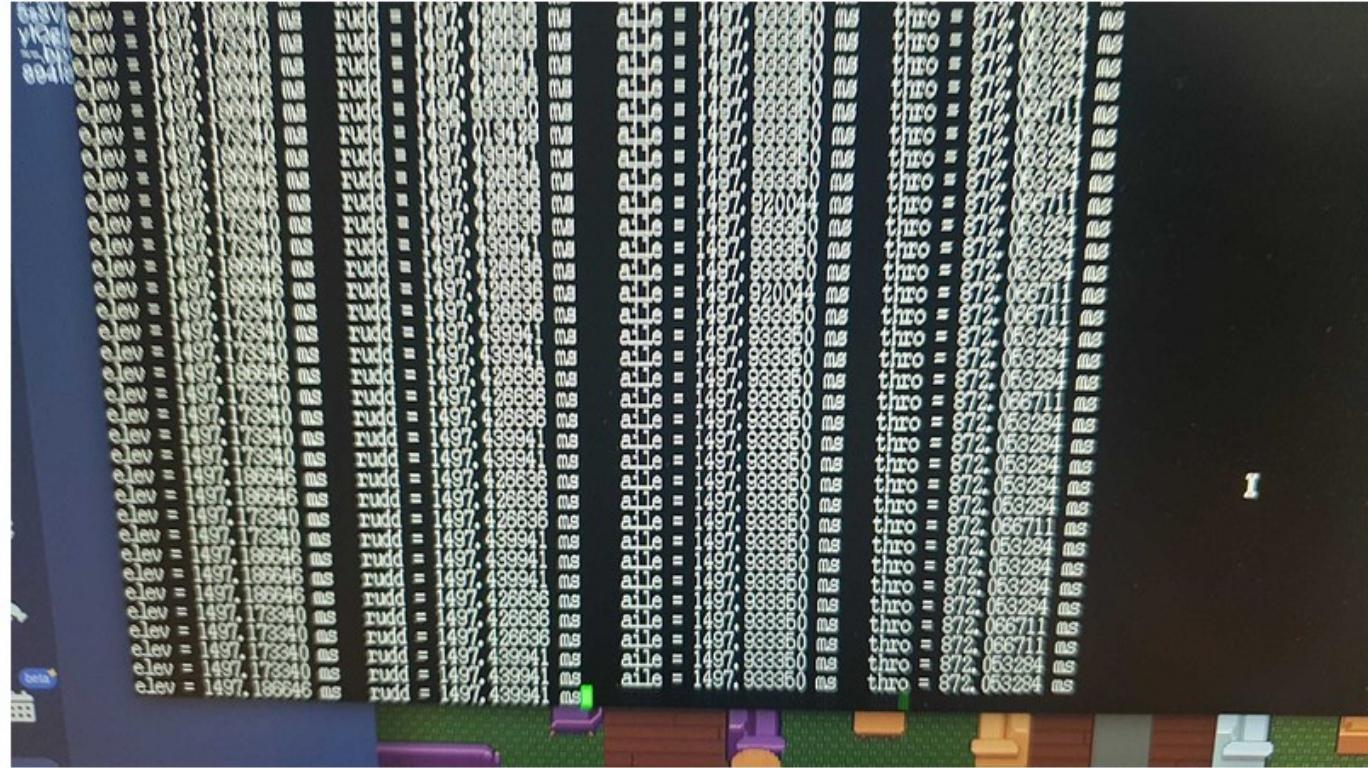
18ms * 8.333 %
18ms의 8.333 % 는 약 1.5 ms $((18 * (10^{-3})) * 8.33300) / 100 = 0.00140004$

이것은 1.5 ms에 해당하는 내용입니다.
그리고 아래 사이트에서 나오는 조종기 스펙을 보면 1.0 ~ 2.0 ms를 가진다고 나타납니다.

좀 더 구체적으로 아래와 같이 명시되어 있습니다.

Channel Pacing:	1200KHz
Spread Spectrum:	FHSS, 67 channels pseudo-random frequency hopping
Adjacent Channel Rejection:	>36dBm
Transmitter Power:	<100mW (20dBm)
Reception Sensitivity:	-104dBm
Transmission Rate :	38kbps
PWM Output Range:	1.0ms~2.0ms
Section precision:	4096, 0.5us per section
Cycle:	15ms/per frame
T8FB(BT) operating voltage:	4.8V~18V
T8FB(BT) operating current:	<80mA
R8EF operating voltage:	4.6~10V
R8EF operating current:	<30mA
Compatible receivers:	R8EF (Std), R8FM, R8SM, R8F, R7FG, R6FG, R6F, R4FGM, R4F

즉 계산된 결과인 1.5 ms는 조종기 스펙상 존재할 수 있는 수치이며
중간값에 배치한 결과임을 알 수 있습니다.



뒤쪽의 ms 는 잘못 표현된 수치에 해당합니다.

아마 이 부분 때문에 혼선을 유발했을 것 같다는 생각이 드는군요.

결론에 해당하는 부분은 확인을 하였으니 각 동작에 대해 살펴보도록 하겠습니다.

eCAP(RF_Control)

먼저 ecapInit() 부분에서 반복되는 패턴들부터 살펴보도록 하겠습니다.

```

71  /** - Setup control register 1
72  *   - Set polarity and reset enable for Capture Events 】
73  *   - Enable/Disable loading on a capture event
74  *   - Setup Event Filter prescale
75  */
76  ecapREG1->ECCTL1 = ((uint16)((uint16)RISING_EDGE << 0U)
77  | (uint16)((uint16)RESET_DISABLE << 1U)
78  | (uint16)((uint16)FALLING_EDGE << 2U)
79  | (uint16)((uint16)RESET_DISABLE << 3U)
80  | (uint16)((uint16)RISING_EDGE << 4U)
81  | (uint16)((uint16)RESET_ENABLE << 5U)
82  | (uint16)((uint16)RISING_EDGE << 6U)
83  | (uint16)((uint16)RESET_DISABLE << 7U)
84  | (uint16)((uint16)1U << 8U)
85  | (uint16)((uint16)0U << 9U));      /* §
86

```

실제 RISING_EDGE, RESET_DISABLE 은 첫 번째 상승 엣지 이후 리셋하지 않음을 의미합니다.

FALLING_EDGE, RESET_DISABLE 를 통해 두 번째 하강 엣지 이후 역시 노 리셋을 의미합니다.

다음 RISING_EDGE, RESET_ENABLE 을 통해 세 번째 상승 엣지에서 리셋을 수행합니다. 3번째 Rising Edge에서

캡처 이벤트 발생시 CAP1-4 레지스터를 활성화 합니다.

Prescaler 는 별도의 지정이 없습니다.

33.5.8 ECAP Control Register 1 (ECCTL1)

Figure 33-21. ECAP Control Register 1 (ECCTL1) [offset = 2Ah]

15	14	13	PRESCALE	9	8
FREE	SOFT				CAPLDEN
R/W-0	R/W-0		R/W-0		R/W-0
7	6	5	4	3	0
CTRRST4	CAP4POL	CTRRST3	CAP3POL	CTRRST2	CAP2POL
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
					R/W-0

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

5	CTRRST3	0	Counter Reset on Capture Event 3. Do not reset counter on Capture Event 3 (absolute time stamp).
		1	Reset counter after Event 3 time-stamp has been captured (used in difference mode operation).
8	CAPLDEN	0	Enable Loading of CAP1-4 registers on a capture event. Disable CAP1-4 register loads at capture event time.
		1	Enable CAP1-4 register loads at capture event time.

CAP1-4 라는게
1다시 4가 아니라
CAP 1~4 전부라는
소리구나... —.—;

TSCTR은 이벤트 시점에 초기화 하는 것을 억제 한다고 했고,
(어떤 것을 초기화 되는걸 억제 한다는 건지?)

CAP1-4 레지스터를 활성화 할 수 있도록 한다는 것이

CAP0이 CAP의 역할을 하게끔 했다는 말일까요??

```
/** - Setup control register 2
 *   - Set operating mode
 *   - Set Stop/Wrap after capture
 */
ecapREG1->ECCTL2 = (uint16)((uint16)CONTINUOUS << 0U)
| (uint16)((uint16)CAPTURE_EVENTS << 1U)
| (uint16)((uint16)0U << 9U)      /* Enable */
| (uint16)0x00000010U;           /* Start counter */
```

다음의 코드는 캡처가 One Shot 일지 Continuous 일지 결정하는 부분입니다.

이후의 부분은 연속 모드에서 캡처 이벤트 3이 발생한 이후 계속 반복함을 의미합니다.

다음의 9번 비트 파트는 FCAP 모듈이 캡처로 동작하게 만듭니다.

TSCTR은 CTR = PRD 이벤트 시점에 초기화되는 것을 억제하며

CAP1, CAP2 레지스터가 백업되는 것을 방지하여 사용자가 CAP1-4 레지스터를 활성화할 수 있도록 합니다.

Free Running Timer는 중간 상태에서 다시 로드되거나 중지되지 않고

끝에서 끝까지 반복적으로 계속 실행됩니다.

0에서 최대 카운트까지 입력 펄스를 계산하고 전체 카운트에 도달하면

플래그(인터럽트를 생성하는 데 사용할 수 있음)를 설정하고 자체를 0으로 재설정하고 다시 반복합니다.

이 Free Running Timer를 사용하여 일정한 간격으로 인터럽트를 생성하고 정확한 지연을 생성할 수 있습니다.

아무튼 TSCTR은 Free-Running 하게 구동하게 됩니다.

여기서부터 시작



33.5.7 ECAP Control Register 2 (ECCTL2)

Figure 33-20. ECAP Control Register 2 (ECCTL2) [offset = 28h]

15	Reserved	11	10	9	8
	R-0		APWM POL	CAP_APWM	SWSYNC
7	SYNCO_SEL	6	5	4	3
	R/W-0		SYNCI_EN	TSCTRSTOP	REARM
	R/W-0		R/W-0	R/W-0	R/W-0
				STOP_WRAP	CONT_ONESH
				R/W-3h	R/W-0

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

9	CAP_APWM	0	CAP/APWM operating mode select.
		0	ECAP module operates in capture mode. This mode forces the following configuration: <ul style="list-style-type: none">Inhibits TSCTR resets via CTR = PRD eventInhibits shadow loads on CAP1 and 2 registersPermits user to enable CAP1-4 register loadCAPx/APWMx pin operates as a capture input
		1	ECAP module operates in APWM mode. This mode forces the following configuration: <ul style="list-style-type: none">Resets TSCTR on CTR = PRD event (period boundary)Permits shadow loading on CAP1 and 2 registersDisables loading of time-stamps into CAP1-4 registersCAPx/APWMx pin operates as a APWM output

4	TSCTRSTOP	0	Time Stamp (TSCTR) Counter Stop (freeze) Control.
		0	TSCTR is stopped.
		1	TSCTR is free-running.

2-1	STOP_WRAP		Stop value for one-shot mode. This is the number (between 1-4) of captures allowed to occur before the CAP(1-4) registers are frozen, capture sequence is stopped.
		0	Wrap value for continuous mode. This is the number (between 1-4) of the capture register in which the circular buffer wraps around and starts again.
		1h	Stop after Capture Event 1 in one-shot mode. Wrap after Capture Event 1 in continuous mode.
		2h	Stop after Capture Event 2 in one-shot mode. Wrap after Capture Event 2 in continuous mode.
		3h	Stop after Capture Event 3 in one-shot mode. Wrap after Capture Event 3 in continuous mode.

0	CONT_ONESH	0	Continuous or one-shot mode control (applicable only in capture mode).
		1	Operate in continuous mode.
		2	Operate in one-shot mode.

eCAP(RF_Control)

```
/** - Set interrupt enable */
ecapREG1->ECEINT = 0x0000U
| 0x0000U
| 0x0008U
| 0x0000U
| 0x0000U
| 0x0000U
| 0x0000U
| 0x0000U;
```

다음으로 인터럽트 활성화 비트는 선택한 이벤트가 인터럽트를 생성하는 것을 차단할 수 있습니다.

이벤트는 여전히 플래그 비트(ECFLG 레지스터)에 래치되며

ECFRC/ECCLR 레지스터를 통해 강제/삭제될 수 있습니다.

결론적으로 Capture Event 3을 인터럽트 소스로 활성화하며 1, 2, 4를 비활성화 합니다.

오버플로우, 등주기 인터럽트, 동일 카운터에 따른 인터럽트는 비활성화합니다.

(상황에 따라서 이 부분이 변경 될 수 있기 때문에 모두 막은것)

아래 코드는 타이머를 실행합니다.

TSCTR 이 실행됩니다.

```
I38 * This function starts Time Stamp Counter
I39 */
I40 /* SourceId : ECAP_SourceId_013 */
I41 /* DesignId : ECAP_DesignId_010 */
I42 /* Requirements : HL_CONQ_ECAP_SR4 */
I43 void ecapStartCounter(ecapBASE_t *ecap)
I44 {
I45     ecap->ECCTL2 |= 0x0010U;
I46 }
```

앞 페이지의 4번 레지스터
자리가 1 되면서 시작

캡처 이벤트 발생시 CAP1-4 레지스터를 활성화합니다.

4	CEVT4	0	Capture Event 4 Interrupt Enable.
		1	Disable Capture Event 4 as an interrupt source.
			Capture Event 4 Interrupt Enable.
3	CEVT3	0	Capture Event 3 Interrupt Enable.
		1	Disable Capture Event 3 as an interrupt source.
			Enable Capture Event 3 as an interrupt source.
2	CEVT2	0	Capture Event 2 Interrupt Enable.
		1	Disable Capture Event 2 as an interrupt source.
			Enable Capture Event 2 as an interrupt source.
1	CEVT1	0	Capture Event 1 Interrupt Enable.
		1	Disable Capture Event 1 as an interrupt source.
			Enable Capture Event 1 as an interrupt source.
0	Reserved	0	Reserved

eCAP(RF_Control)

```
408 * This function enable loading of CAP1-  
409 */  
410 /* SourceId : ECAP_SourceId_011 */  
411 /* DesignId : ECAP_DesignId_008 */  
412 /* Requirements : HL_CONQ_ECAP_SR11 */  
413 void ecapEnableCapture(ecapBASE_t *ecap)  
414 {  
    ecap->ECCTL1 |= 0x0100U;  
416 }
```

원본 저장하기 ▾

결국 아래 코드를 해석하자면 아래와 같습니다.

초기 파형이 뜨는 순간을 CEVT1 (무시)

이후 파형이 가라앉는 순간을 CEVT2 (무시)

다시 뜨는 케이스에서 CEVT3 으로 인해 실제 인터럽트가 발생합니다.

```
139     cap[0] = ecapGetCAP1(ecapREG1);  
140     cap[1] = ecapGetCAP2(ecapREG1);  
141     cap[2] = ecapGetCAP1(ecapREG2);  
142     cap[3] = ecapGetCAP2(ecapREG2);  
143     cap[4] = ecapGetCAP1(ecapREG3);  
144     cap[5] = ecapGetCAP2(ecapREG3);  
145     cap[6] = ecapGetCAP1(ecapREG4);  
146     cap[7] = ecapGetCAP2(ecapREG4);  
147
```

원본 저장하기 ▾

33.5.8 ECAP Control Register 1 (ECCTL1)

Figure 33-21. ECAP Control Register 1 (ECCTL1) [offset = 2Ah]

15	14	13	PRESCALE				9	8	CAPLDEN
FREE	SOFT						R/W-0		R/W-0
R/W-0	R/W-0						R/W-0		R/W-0
7	6	5	4	3	2	1	0	8	CAP1POL
CTRRST4	CAP4POL	CTRRST3	CAP3POL	CTRRST2	CAP2POL	CTRRST1	CAP1POL	R/W-0	R/W-0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

8	CAPLDEN	0	Enable Loading of CAP1-4 registers on a capture event.
		0	Disable CAP1-4 register loads at capture event time.
		1	Enable CAP1-4 register loads at capture event time.

eCAP(RF_Control)

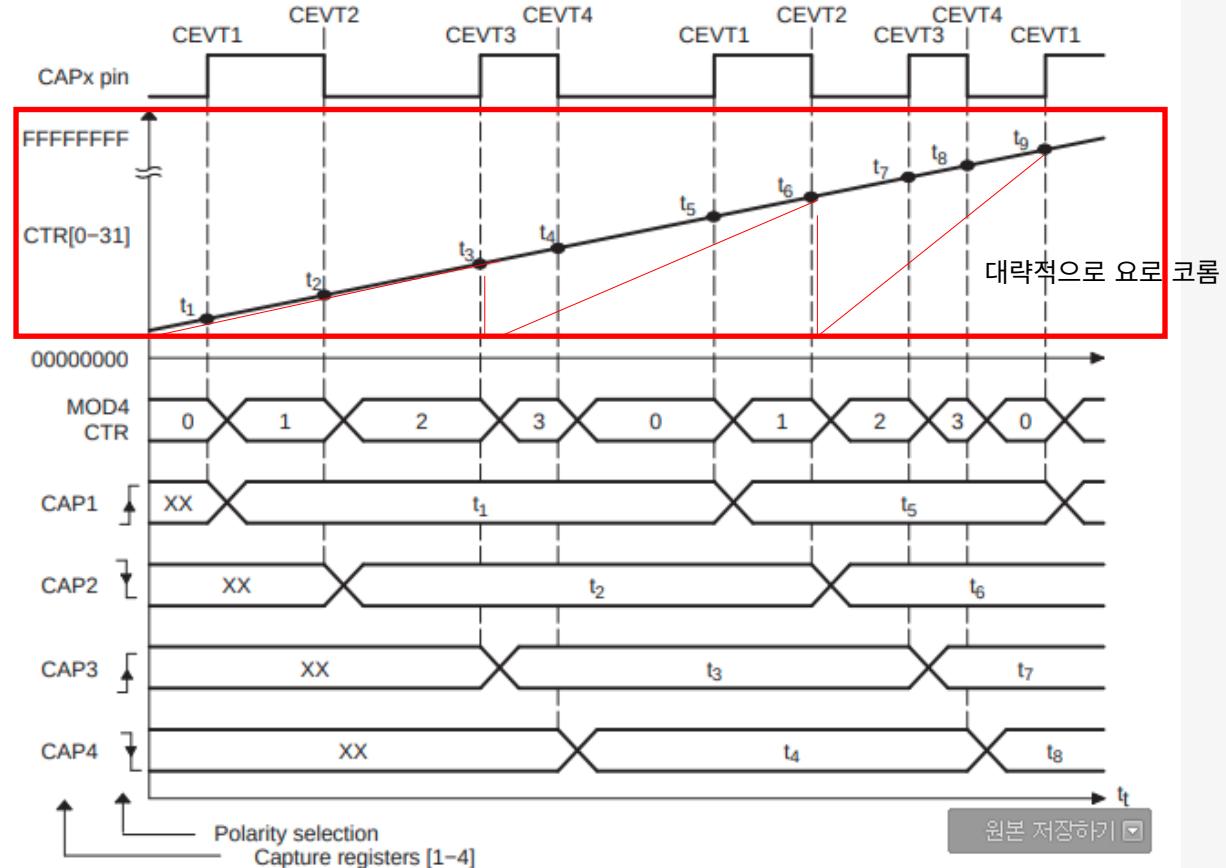
결국 데이터시트상에서 4번째 예와 가장 유사한 형태라 보면 되겠습니다.
물론 차이가 있습니다.

즉 아래와 같은 2번 예의 형태와 위의 4번 형태가 적절하게 혼합되어야 합니다.

33.3.2 Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger

In Figure 33-10, the eCAP operating mode is almost the same as in the previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information: Period1 = $t_3 - t_1$, Period2 = $t_5 - t_3$, ...etc Duty Cycle1 (on-time %) = $(t_2 - t_1) / \text{Period1} \times 100\%$, etc. Duty Cycle1 (off-time %) = $(t_3 - t_2) / \text{Period1} \times 100\%$, etc.

Figure 33-10. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect

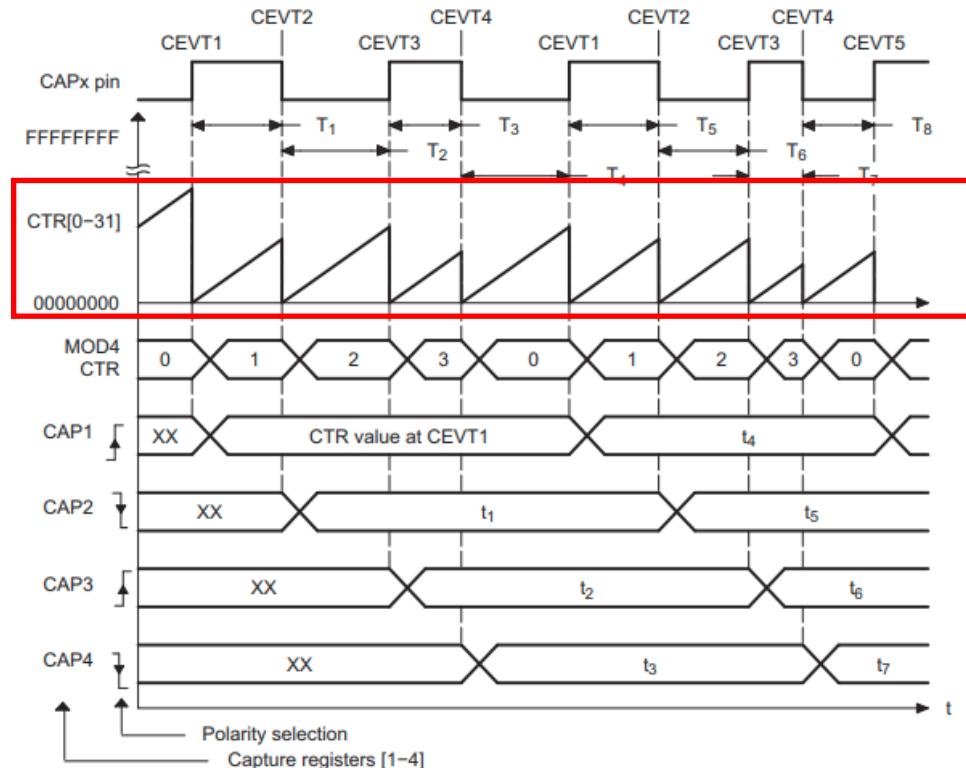


33.3.4 Example 4 - Time Difference (Delta) Operation Rising and Falling Edge Trigger

In Figure 33-12, the eCAP operating mode is almost the same as in previous section except Capture events are qualified as either Rising or Falling edge, this now gives both Period and Duty cycle information: Period1 = $T_1 + T_2$, Period2 = $T_3 + T_4$, ...etc Duty Cycle1 (on-time %) = $T_1 / \text{Period1} \times 100\%$, etc. Duty Cycle1 (off-time %) = $T_2 / \text{Period1} \times 100\%$, etc

During initialization, you must write to the active registers for both period and compare. This will then automatically copy the init values into the shadow values. For subsequent compare updates, during runtime, only the shadow registers must be used.

Figure 33-12. Capture Sequence for Delta Mode Time-stamp With Rising and Falling Edge Detect



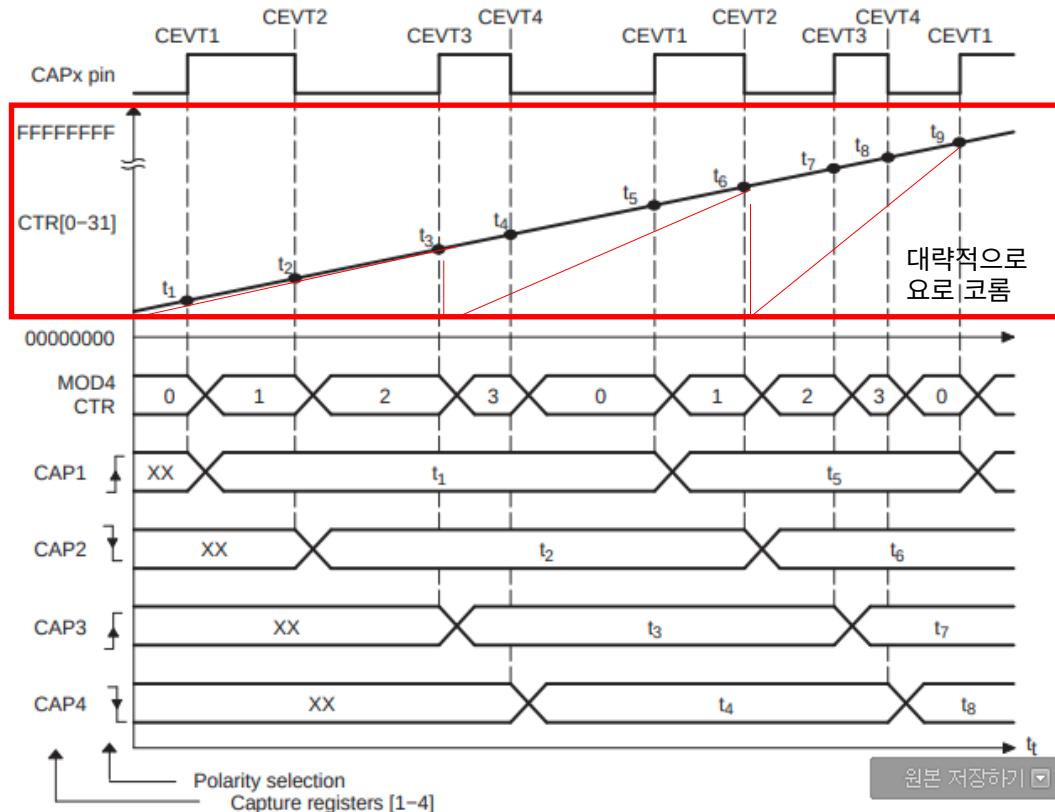
실제로는 카운터가 초기화 되지 않은 상태로 CEVT3 까지 올라간 이후 초기화되며 다시 카운팅합니다.

eCAP(RF_Control)

33.3.2 Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger

In Figure 33-10, the eCAP operating mode is almost the same as in the previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information: Period1 = $t_3 - t_1$, Period2 = $t_5 - t_3$, ...etc. Duty Cycle1 (on-time %) = $(t_2 - t_1) / \text{Period1} \times 100\%$, etc. Duty Cycle1 (off-time %) = $(t_3 - t_2) / \text{Period1} \times 100\%$, etc.

Figure 33-10. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect



Falling에서 카운팅한 값이 CAP2에 들어갑니다.

그리고 다시 Rising을 하는 순간 새롭게 카운터를 리셋하여 카운팅을 합니다.

이후 Falling에서 카운팅을 하게 되므로 CAP2 - CAP1의 차이를 샘하면 적정한 시간을 얻을 수 있습니다.

Falling에서 카운팅
하므로 CAP2-CAP1
차이

질문)

여기서 Period는 CAP4 까지의 시간을 말하는건가요??

결론적으로 TSCTR과 CTRPHS를 잘 봐야합니다.

실제 HalCoGen 구성을 보면 Period가 1000 ns (1 us) 임을 볼 수 있습니다.

실제 코드상에서 VCLK3_FREQ 부분이 75로 처리되는 것을 볼 수 있습니다.

이 숫자는 아래와 같이 도출됩니다.

```
232 */  
233 #define VCLK3_FREQ 75.000F  
234
```

실제 1000 ns라는 주기를 75 MHz라는 VCLK3의 속도로 카운팅한다고 가정하면
정말 재미있게도 75라는 수치값이 나온다는 것입니다.

이 뜻은 1000 ns라는 한 주기를 카운팅하는데 75 MHz의 속도로 75번 카운팅을 해야함을 의미합니다.

사실 아래와 같이 복잡하게 생각할 필요도 없이 10^{-6} 과 75에 붙은 10^6 이
서로 상쇄되면서 75가 된다는 것도 빠르게 알 수 있습니다.

$(1000 * 10^{-9}) / (1 / (75 * 10^6))$

전체 쇼핑 뉴스 이미지 지도 더보기 도구

검색결과 약 1개 (1.11초)

$(1000 * (10^{-9})) / (1 / (75 * (10^6))) =$
75

아하!

eCAP(RF_Control)

실제 코드 상에서 VCLK3_FREQ라는 부분이 추가되는 이유는 위와 같은 이유 때문입니다.
결국 1000 ns를 75 MHz로 카운팅하는데 필요한 횟수가 75니까
cap2 - cap2 사이값을 75(한 주기)로 나눠서 Duty를 본다고 보면 되겠습니다.

질문) Cap2 - cap2 는 cap2 - cap1의 오타 인걸까요?!?

Handwritten notes on a spiral notebook page:

- Top right: $\frac{1000}{75}$
- Equation: $F = \frac{1}{T} \rightarrow \frac{1}{(cap[1] - cap[0])} \times \frac{1000}{75} = \text{xxx ms}$
- Note: 개수(카운팅), 양변 1000 나누기, 10³.
- Equation: $(cap[1] - cap[0]) / 75 = \text{xxx us}$
- Text: 75 MHz의 주기는 곱하는 형태가 된다?
75 MHz의 주기 형식은 $\frac{1}{75 \times 10^6}$ (1/T).
- Text: 두식을 서로 곱하면 두사의 시간초가 나온다.
 $(cap[1] - cap[0]) / 75 = 10^6 * \text{xxx us}$
- Equation: $(cap[1] - cap[0]) / 75 = \text{xxx us}$
- Text: 1.0 ms 가 1000×10^{-3} 에 해당?
 $1000 \times 10^{-3} \times 10^6 = 1 \text{ ms}$
- Equation: $75000 \times 10^{-3} = (cap[1] - cap[0])$ 갱이랑
75ms 가 갱?

다소 혼동이 발생한 이유라면 1000 ns를 가상의 주기로 잡은 부분 때문에 혼동이 발생한 것 같습니다.
실제 VCLK3의 주기는 13.3333 ns라고 보면 됩니다.

$(cap[1] - cap[0]) * 1000 / 75 = \text{xxx ms}$ 에 대한 해석을 역으로 추적해봐도 됩니다.

$(cap[1] - cap[0]) * 1000 / 75 = \text{xxx ms} \leqslant$ 양변에 1000을 나눕니다.

$(cap[1] - cap[0]) / 75 = \text{xxx us}$

질문) 계산 과정이 잘 이해가 안되네요..

위 식은 결국 형태가 75 MHz의 주기를 곱하는 형태가 됩니다.

75 MHz의 주기 형식은 $1 / (75 * 10^6)$ 이기 때문입니다.

이를 활용하여 위 식을 다시 표기하면 아래와 같이 표기할 수 있습니다.

$(cap[1] - cap[0]) = \text{개수(카운팅)}$

$(1 / 75) * 10^{-6} = \text{카운터의 1주기}$

어떤 공식이 활용 된 건지 잘 모르겠습니다!

위의 두 식을 서로 곱하면 둘 사이의 시간 초가 나옵니다.

약간 변형을 가해주면 아래와 같이 적을 수 있습니다.

$(cap[1] - cap[0]) / 75 = 10^6 * \text{xxx us}$

이를 다시 정리하면 아래와 같이 정리됩니다.

$(cap[1] - cap[0]) / 75 = \text{xxx 초}$ (10^6 과 us에 해당하는 10^{-6} 이 서로 상쇄)

실제로 1.0 ms는 $1000 * 10^{-3}$ 에 해당하므로 이 값을 위에 적용하면
 $75000 * 10^{-3} = (cap[1] - cap[0])$ 사이의 갭임을 알 수 있습니다.

eCAP(RF_Control)



아래는 ecap3에 대한 인터럽트가 실제 처리되는 부분입니다.

```
979 void ecap3Interrupt(void)
980 {
981     uint16 Int_Flag = ecapREG3->ECFLG & ecapREG3->ECEINT;
982
983 /* USER CODE BEGIN (6) */
984 /* USER CODE END */
985
986     /* Clear Events, */
987     /* Note : Current Implementation clears multiple all events
988         before this point, User notification function is called
989         after this point. */
990     ecapREG3->ECCLR = Int_Flag;
991
992     /* Clears the interrupt flag and enables further interrupt
993         if an event flags is set to 1. */
994     ecapREG3->ECCLR = 1U;
995
996     /* Passing the Interrupt Flag to the user Notification */
997     ecapNotification(ecapREG3,Int_Flag);
998
999 /* USER CODE BEGIN (7) */
000 /* USER CODE END */
001 }
```

감사합니다.

eCAP(RF_Control)

링크쌤 칼럼 >

<https://cafe.naver.com/eddicorp/417>



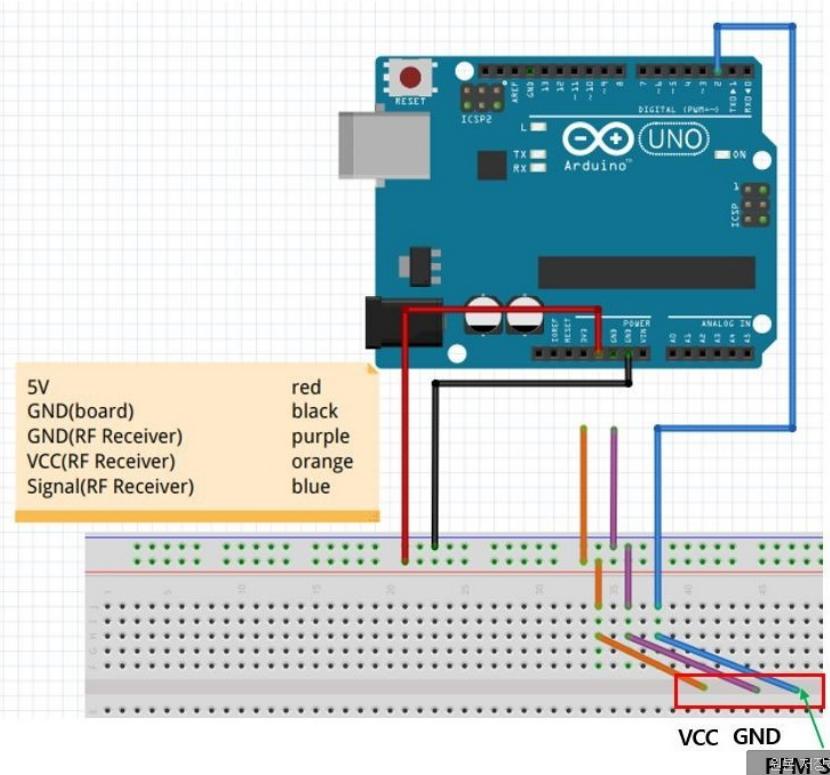
[링크쌤 컬럼] R8EF RF 수신기의 PPM 동작



안녕하세요

에디로봇아카데미의 링크쌤입니다.

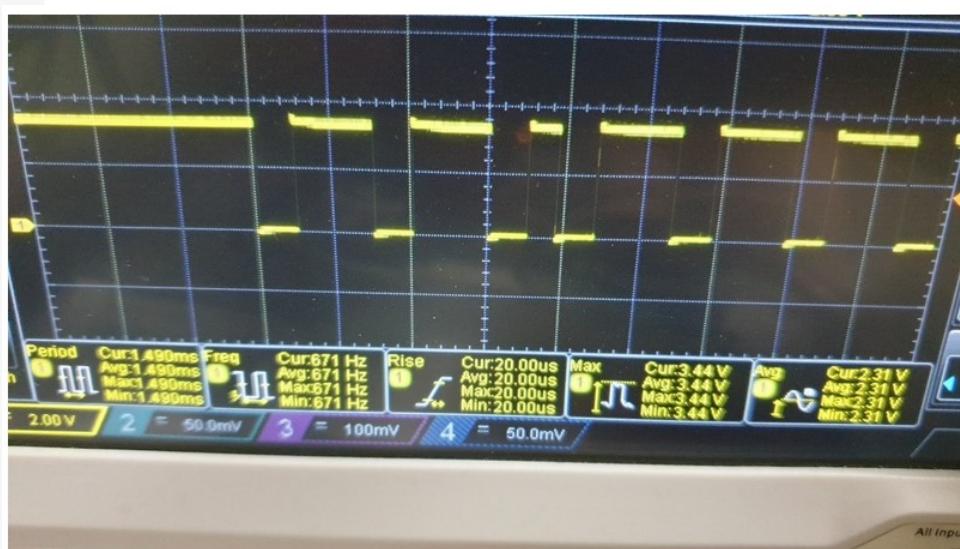
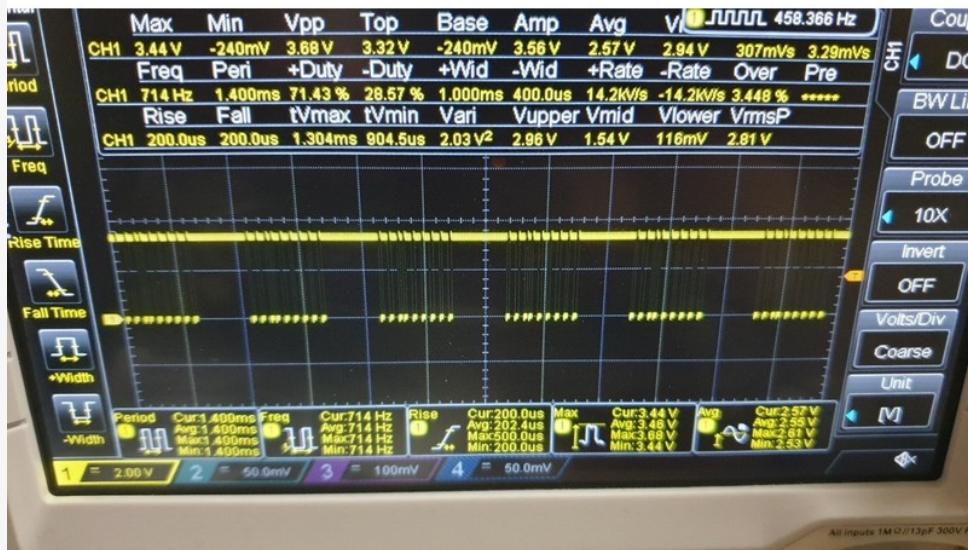
실질적으로 신호 캡처가 어려운 경우 활용할 수 있는 전략으로 PPM 을 사용할 수 있습니다.



실제 RF 조종기에 적용하여 Throttle 값을 조정해보는 모습입니다.

eCAP(RF_Control)

실제 스코프로 PPM 신호를 찍어보면 아래와 같은 출력을 확인할 수 있습니다.



표준 RC 수신기는 4개의 주요 블록으로 구성됩니다.

무선 주파수 블록(FM 수신기 + 안테나)은 복조를 통해 신호를 다른 블록에서 사용할 수 있는 형태로 변환합니다.

이 특정 신호는 PPM(펄스 위치 변조) 코딩된 신호입니다.

펄스 위치를 PWM(Pulse Width Modulation) 서보 신호로 변환하는 PPM 신호 디코더가 존재합니다.

PPM 디코더를 다시 초기화하기 위해 PPM 프레임 분리를 감지하는 Synchro Blank Detector 블록이 존재합니다.

수신기의 모든 블록에 전력을 공급하는 전원 공급 장치 블록으로 구성됩니다.

위의 파형에서 주의해야 할 점은 아래와 같습니다.

1. PPM 프레임 주기는 RC 송신기 유형 및 제조업체에 따라 다릅니다.
2. 프레임 분리 공백 시간은 송신기 채널 수에 따라 달라지며 모든 채널 지속 시간(T_n)에 따라 다릅니다.

eCAP(RF_Control)

수신기의 파형

결국 실제 출력으로 나오는 파형을 스코프에서 찍어보면서

어떤식으로 신호를 감지할지 전략을 구상해야 한다는 뜻입니다.

PPM 신호(또는 펄스 위치 변조)는 대부분의 송신기와 수신기에서 정보를 송신기에서 수신기로 전송하는 데 사용됩니다.

신호 자체는 고정 길이의 일련의 펄스입니다.

전자 장치에서 이러한 펄스는 다양한 전압의 전기 신호입니다
(+2.5V와 +5V 사이의 값을 가집니다).

이 펄스 사이에는 다양한 길이의 일시 중지가 있습니다.

전자 제품에서 이러한 펄스는 0V에서 +0.5V 사이입니다.

일부 장비에서는 펄스가 낮고 일시 중지가 높지만 이는 그다지 중요하지 않습니다.

이러한 유형의 신호에 대한 정보는 펄스 길이에 다음 일시 중지 길이를 더한 값으로 인코딩됩니다.

또는 두 펄스의 시작 사이의 시간(또는 끝은 펄스가 고정 길이이므로 문제가 되지 않음)입니다.

이것은 펄스가 높든 낮든 중요하지 않은 이유이기도 합니다.

신호의 상승 또는 하강 에지 중 하나를 선택하고

두 상승 또는 하강 에지 사이의 시간을 측정하기만 하면 됩니다.

채널은 신호에서 시간 순서대로 서로를 따릅니다.

즉, 채널 1이 먼저 나오고 채널 N이 마지막에 옵니다.

펄스 사이의 시간은 RC 표준의 1.0 ~ 2.0 ms 신호를 생성하는 데 사용됩니다.

펄스의 길이는 일반적으로 약 500마이크로초이며 일시 중지는 500~1500마이크로초입니다.

이들을 함께 추가하면 1000~2000마이크로초의 채널 값을 얻을 수 있으며

이는 서보 신호에서 사용된 것과 동일한 범위입니다.

마지막 채널 뒤에 프레임 끝(또는 프레임 시작) 일시 중지가 있습니다.

이 일시 중지의 길이는 5000~20000마이크로초입니다.

리시버는 이에 대해 크게 고려하지 않습니다.

일시 정지의 길이는 다양할 수 있으므로 프레임을 고정 길이로 채울 수 있습니다.

수신기는 상승 또는 하강 에지 중 하나를 선택하고,

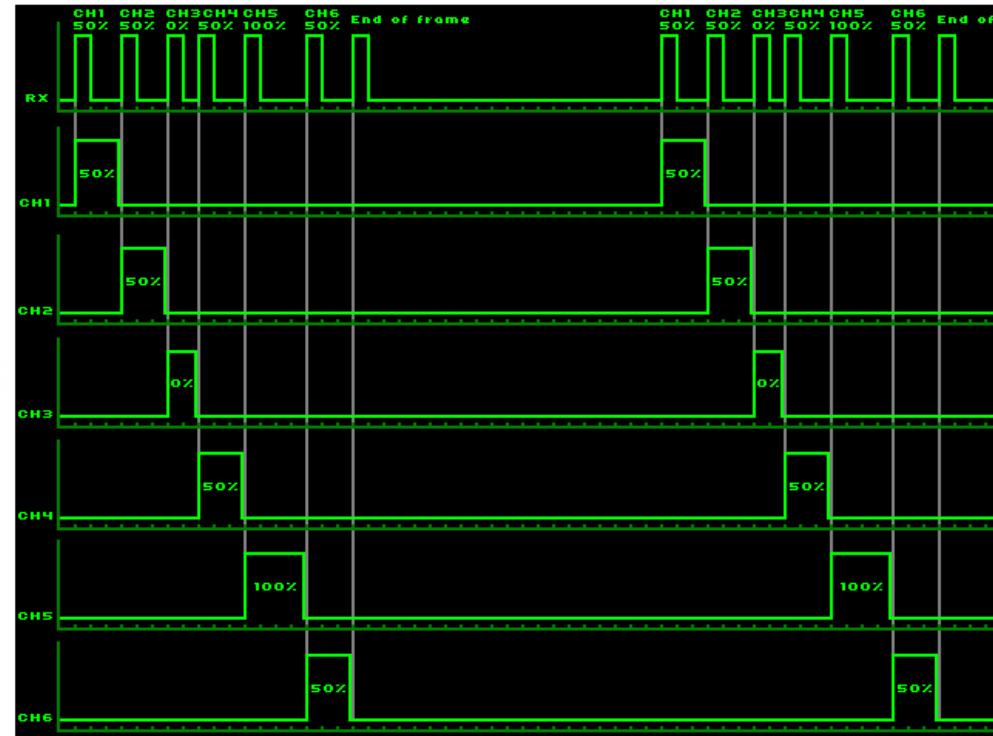
그렇게 하면 첫 번째 채널의 신호 핀에 하이 신호를 넣습니다.

다음 에지를 수신하면 채널 1에 로우 신호를, 채널 2의 신호 핀에 하이 신호를 넣습니다.

이것은 모든 채널이 처리될 때까지 계속됩니다.

그런 다음 End Of Frame 일시 중지를 수신할 때까지 기다립니다.

시각화하면 다음과 같습니다.



그러므로 PPM 신호의 해석은 Rising, Falling을 감지하고

그 다음 신호의 Rising을 감지하면서 전체 펄스를 파악하면 됩니다.

이를 통해 저속의 샘플링 시간을 가진 시스템도 RF 신호를 감지할 수 있도록 구성할 수 있습니다.

감사합니다.

PPM 신호라는 것에 대한 개념은 잘 이해가 안되는데,
송신기로 받은 RX 신호를
Ch1~6 으로 감지해서(end of frame 까지) RF 신호를
감지 할 수 있도록 하고
이것을 수신기로 알려 준다? 그런 느낌인 것 같다.

[링크쌤 컬럼] eCAP Alternatives Conflict 문제

링크쌤 카페매니저 M 1:1 채팅
2022.02.25. 14:49 조회 20

안녕하세요

에디로봇아카데미의 링크쌤입니다.

eCAP4 번을 보면 MiBSPI1NENA에 대한 Alternatives임을 알 수 있습니다.

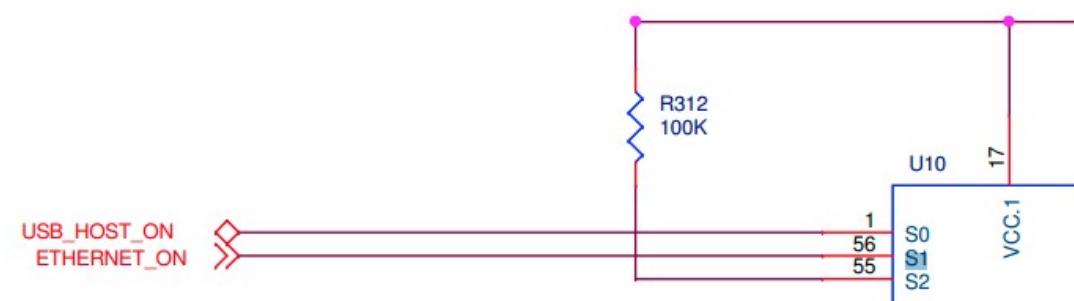


여기서 이제 몇 가지 간과하게 되는 사항들이 있는데 주의해야 합니다.

보드상에서 Ethernet ON 부분이 OFF일 수도 있고 ON일 수도 있습니다. Dip switch 조절 필요

테스트하는 쪽에서 이 부분을 테스트 이후 원상복구 안 할 수 있어서 주의가 필요합니다.

댓글



실제 S1에 해당하는 사항이 Ethernet ON에 해당합니다.

여기서 또 확인해야 하는 사항은 아래 회로도입니다.



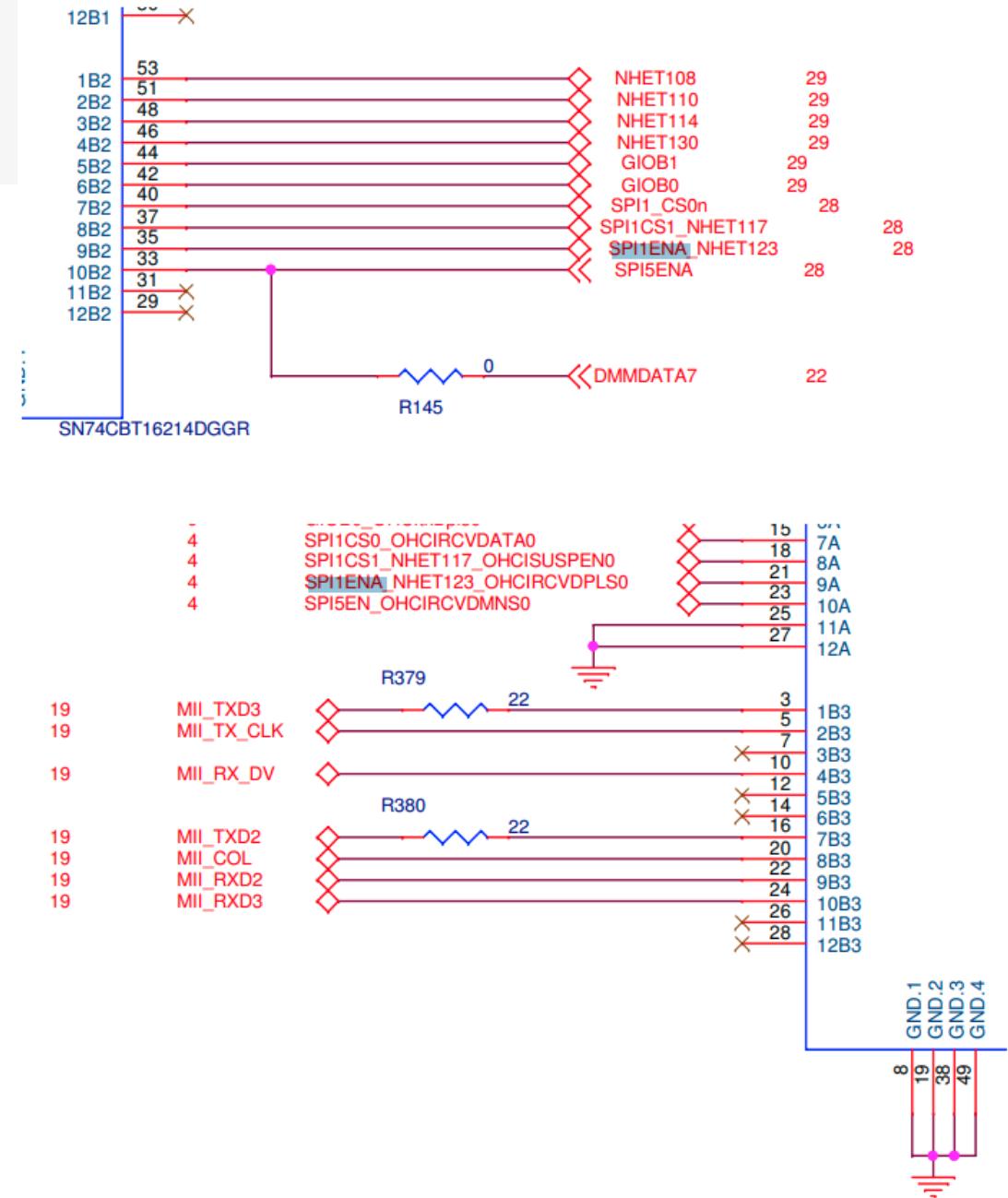
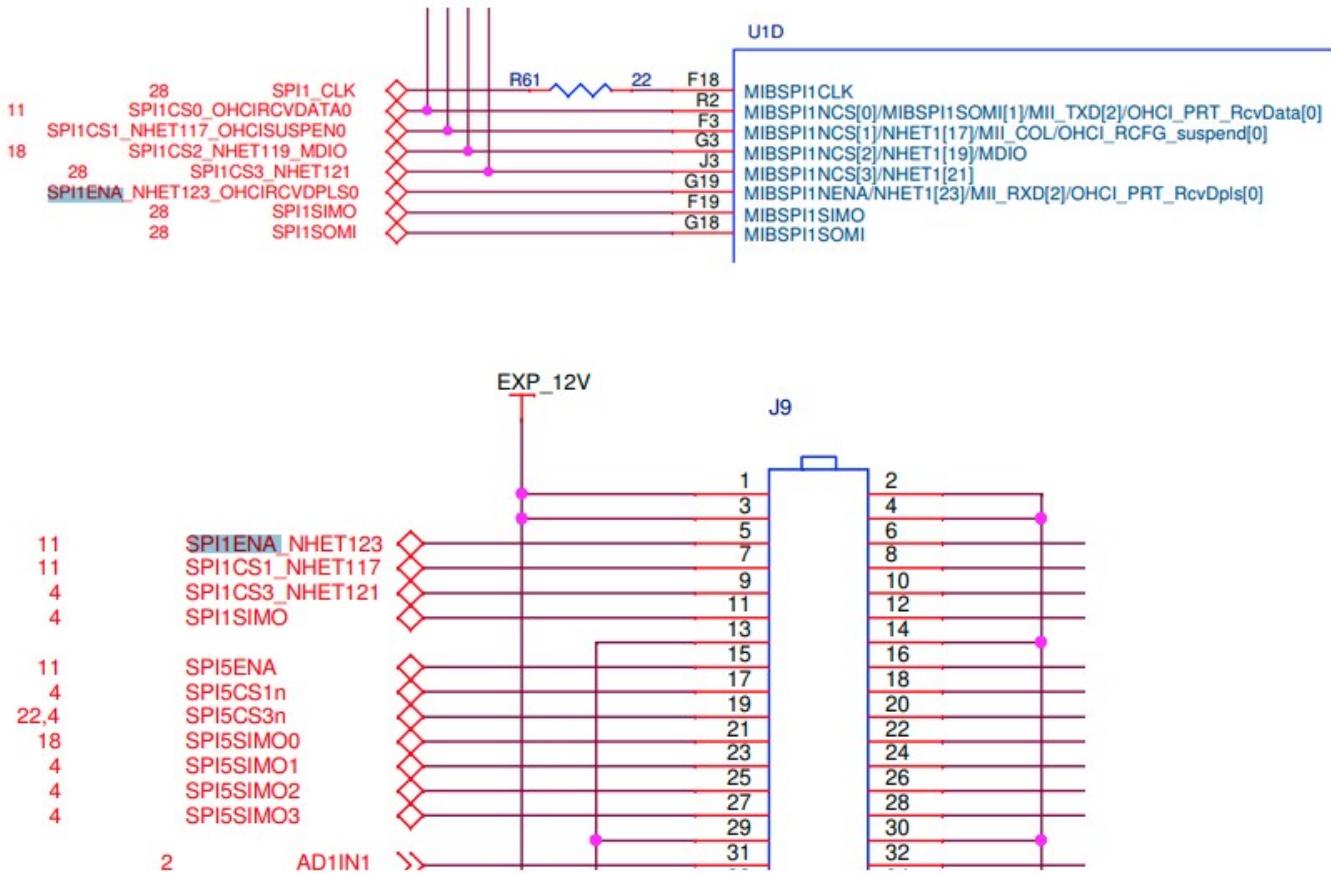
실제 ETHERNET_ON을 보면 11, 18이 보이는데 11이나 18이 적힌 녀석들을 살펴보면 Ethernet과 관련된 녀석들이 자동으로 맵핑된다고 볼 수 있습니다.

MII_TXD, MDIO, MII_COL, MII_RXD 등을 볼 수 있습니다.

실제 eCAP4는 이 MII_RXD와 Alternatives입니다.

PinMux 지정이 없더라도 HW SW에 의해 이것이 지정될 수 있음을 간과해서는 안될 것입니다.

이런 부분을 전혀 신경쓰고 있지 않다면 다소 봉변을 당할 수 있는 부분입니다.



감사합니다.