임베디드 기반 SW 개발 프로젝트

AURIX TC275 보드 ADC 사용

- Analog Data Sampling to Digital Conversion -

현대자동차 입문교육
박대진 교수

실습 목표

1. 보드의 **GPIO** 출력을 제어해서 확장 보드 **RGB LED 각각의 색상을 on/off** 해보도록 한다.

2. 확장 보드의 가변 저항에 의한 전압을 **ADC**로 읽고, **샘플링된 ADC값의 범위에 따라 RGB LED 출력을 제어**한다.

사용된 약어 정리

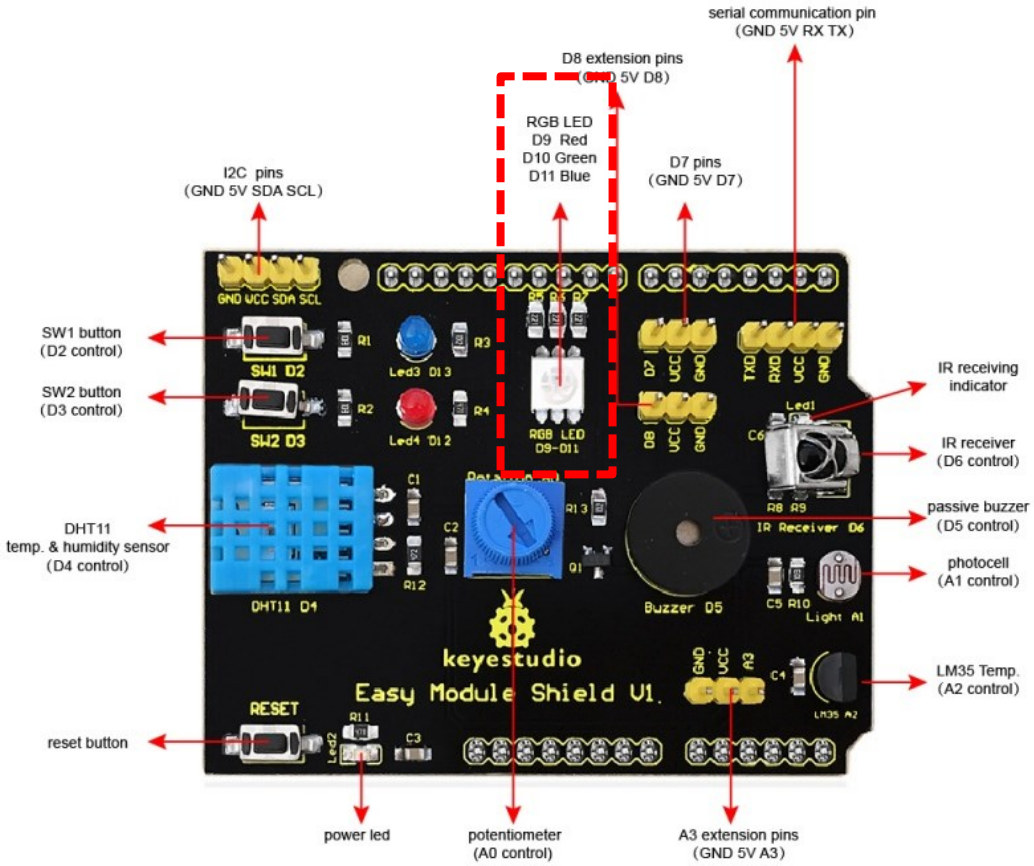
- VADC (Versatile Analog-to-Digital Converter)
- SAR (Successive Approximation Register)
- SCU (System Control Unit)

RGB LED 회로 구조 파악

: 확장 보드 PCB에 RGB LED가 연결된 회로

- 확장 보드의 RGB LED를 사용하기 위해서는 TC275 보드의 어떤 핀이 Easy Module Shield 확장 보드의 RGB LED와 연결되어 있는지 알아야 함

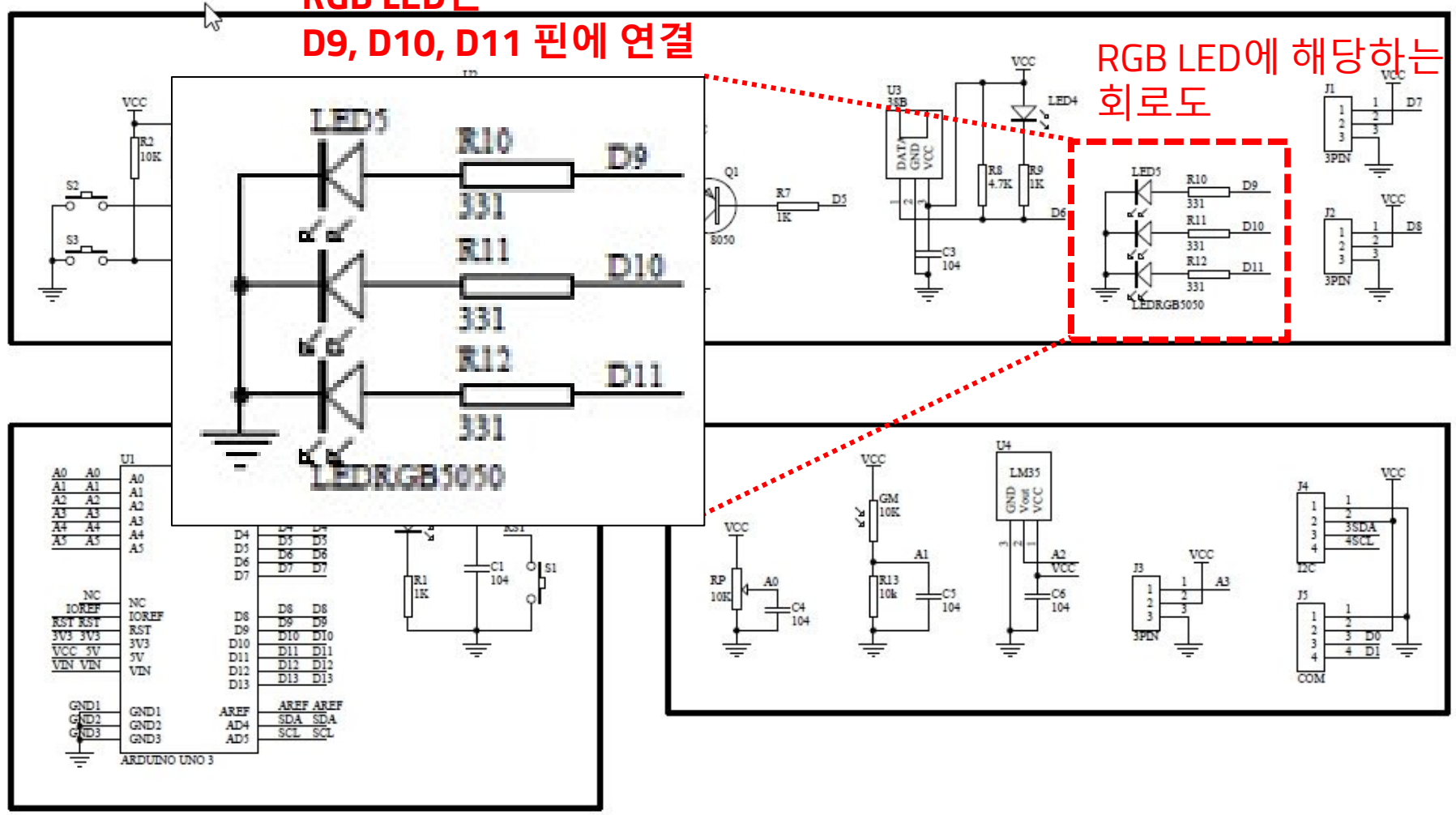
  → 데이터 시트 분석 필요



회로도 @ 확장 보드 데이터 시트

: RGB LED 회로 부분 확인

RGB LED는
D9, D10, D11 핀에 연결



Pin Map @ TC275 보드

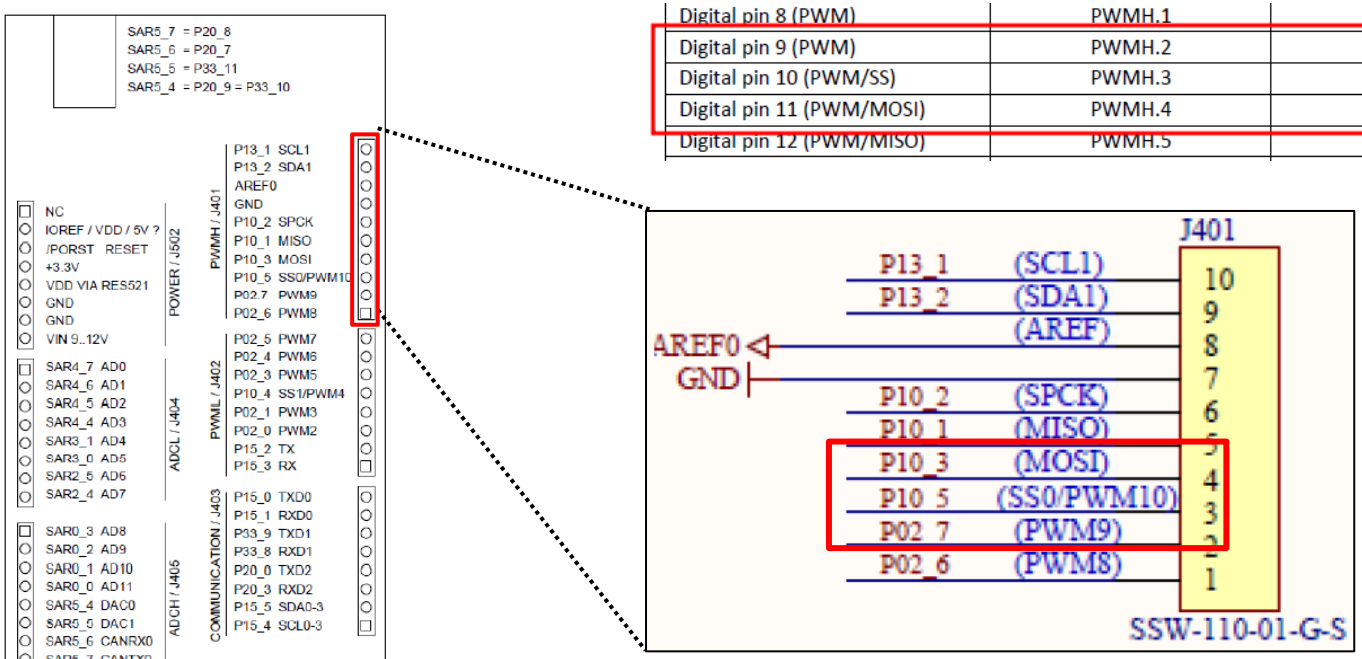
: RGB LED핀과 TC275 보드 핀과의 회로 연결 정보 확인

- 앞서 RGB LED는 확장 보드의 핀 D9, D10, D11에 연결 되어있는 것을 확인

→ 그렇다면 확장 보드의 핀 **D9, D10, D11**는 **TC275** 보드의 어느 핀에 연결?

Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.44

Digital pin 8 (PWM)	PWMH.1	P2.6
Digital pin 9 (PWM)	PWMH.2	P2.7
Digital pin 10 (PWM/SS)	PWMH.3	P10.5
Digital pin 11 (PWM/MOSI)	PWMH.4	P10.3
Digital pin 12 (PWM/MISO)	PWMH.5	P10.1



TC275 MCU I/O 중,
Port 02 pin 7 (P02.7)
Port 10 pin 5 (P10.5)
Port 10 pin 3 (P10.3)
와 연결됨

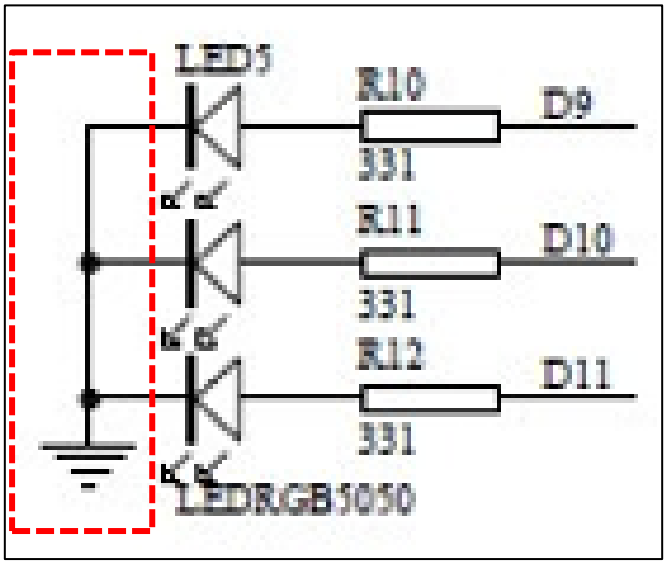
Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.42

Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.46

RGB LED를 제어하려면...

- RGB LED 회로를 보면 D9, D10, D11 핀 반대쪽에 GND가 있으므로
  - → HIGH 값을 인가해야 LED가 켜질 수 있음
  - → 반대로 LOW 값을 인가하면 LED는 꺼짐

- **즉, LED를 켜려면 MCU GPIO P02.7, P10.3, P10.5 핀에 HIGH 값을 인가해야 함**



GPIO 레지스터 설정

: 핀의 입출력 방향 및 출력값 설정

- P10.3, P10.5, P02.7, 핀의 출력 모드 동작을 위해서는 GPIO Port 레지스터 항목에서
  - **P10_OUT**
  - **P10_IOCR0**
  - **P10_IOCR4**
  - **P02_OUT**
  - **P02_IOCR4**
- 5가지 레지스터에 설정이 필요함

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1164

Table 13-16 Port 10 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select. Reg./Bit Field	Value
P10.3	I	General-purpose input	P10_IN.P3	P10_IOCR0. PC3	$0XXXX_B$
		GTM input	TIN105		
		QSPI1 input	MTSR1A		
		SCU input	REQ3		
		GPT120 input	T5INB		
	O	General-purpose output	P10_OUT.P3		$1X000_B$
		GTM output	TOUT105		$1X001_B$
		VADC output	VADCG6BFL3		$1X010_B$
P10.5	I	General-purpose input	P10_IN.P5	P10_IOCR4. PC5	$0XXXX_B$
		GTM input	TIN107		
		SCU input	HWCFG4		
		MSC0 input	INJ01		
	O	General-purpose output	P10_OUT.P5		$1X000_B$
		GTM output	TOUT107		$1X001_B$
		ASCLIN2 output	ATX2		$1X010_B$
		QSPI3 output	SLSO38		$1X011_B$
P02.7	I	General-purpose input	P02_IN.P7	P02_IOCR4. PC7	$0XXXX_B$
		GTM input	TIN7		
		QSPI3 input	SCLK3A		
		PSI5 input	PSIRX2B		
		SENT input	SENT1C		
		CCU60 input	CC61INC		
		CCU60 input	CCPOS1A		
		CCU61 input	T13HRB		
		GPT120 input	T3EUDA		
		DSADC input	DSCIN3B		
		CIF input	CIFD7		
		DSADC input	DSITR4E		
	O	General-purpose output	P02_OUT.P7		$1X000_B$
		GTM output	TOUT7		$1X001_B$
		Reserved	–		$1X010_B$

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1156

GPIO 레지스터 설정 – P02 Address

: Port 02 영역이 위치한 주소 확인

- 레지스터 설정을 위해 레지스터가 위치한 주소 파악 필요

1. GPIO (Ports) 레지스터 영역 찾기

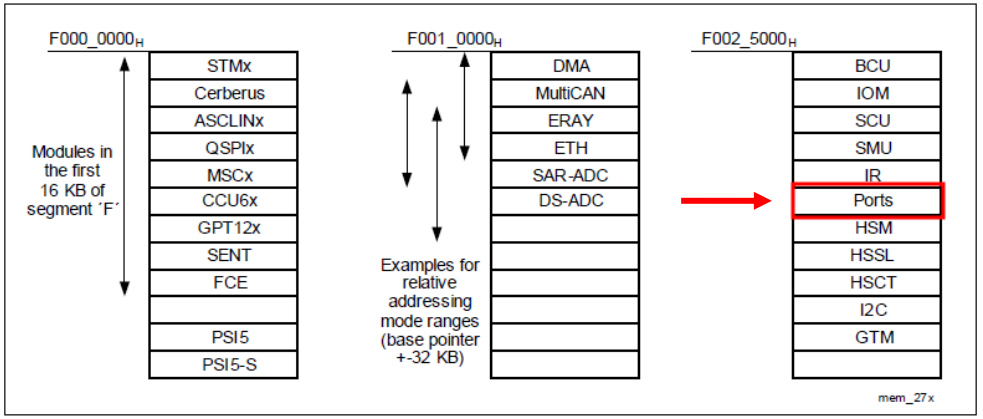


Figure 3-1 Segment F Structure

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.227

2. 실습에서 사용하는 Port 02 에 해당하는 레지스터 영역 찾기
   - 시작 주소 (Base address)
   - **= 0xF003A200**

	$F003\ 9FFF_H$	Kbyte		
Port 00	$F003\ A000_H$ - $F003\ A0FF_H$	256 byte	access	access
Port 01	$F003\ A100_H$ - $F003\ A1FF_H$	256 byte	access	access
Port 02	$F003\ A200_H$ - $F003\ A2FF_H$	256 byte	access	access

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.230

GPIO 레지스터 설정 – P02 Address 계산

: Port 02의 각 레지스터가 위치한 주소 확인

3. 사용할 특정 레지스터의 주소 찾기
   - **P02_OUT** Offset Address = 0x0000
   → P02_OUT 레지스터 주소
   → = 0xF003A200 + 0x0000 = **0xF003A200**

   - **P02_IOCR4** Offset Address = 0x0014
   → P02_IOCR4 레지스터 주소
   → = 0xF003A200 + 0x0014 = **0xF003A214**

Table 13-4 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
Pn_OUT	Port n Output Register	0000_H	U, SV	U, SV, P	Application Reset	Page 13-38
Pn_OMR	Port n Output Modification Register	0004_H	U, SV	U, SV, P	Application Reset	Page 13-39
ID	Module Identification Register	0008_H	U, SV	BE	Application Reset	Page 13-13
Pn_IOCR0	Port n Input/Output Control Register 0	0010_H	U, SV	U, SV, P	Application Reset	Page 13-14
Pn_IOCR4	Port n Input/Output Control Register 4	0014_H	U, SV	U, SV, P	Application Reset	Page 13-14
Pn_IOCR8	Port n Input/Output Control Register 8	0018_H	U, SV	U, SV, P	Application Reset	Page 13-14

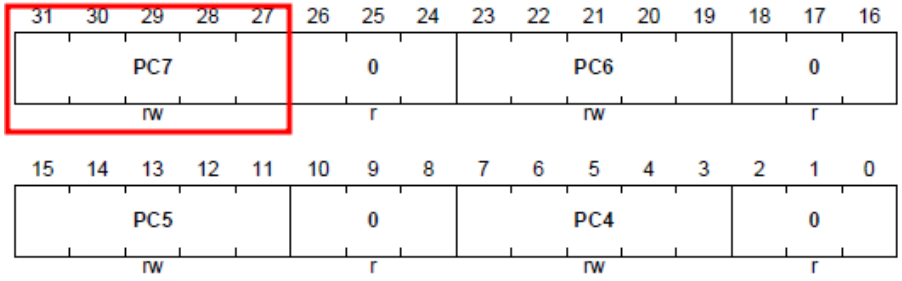
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1074

GPIO 레지스터 설정 – P02 Write Value

: Port 02의 핀 7를 출력 모드로 사용

- **P02.7** 을 General Purpose Output (push-pull)으로 사용하기 위해 P02_IOCR4 레지스터에 어떤 값을 써야 하는지 확인 → **0x10 (2진수 10000b)**

P02_IOCR4 레지스터 @ 0xF003A214



Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 4 to 7** This bit field determines the Port n line x functionality (x = 4-7) according to the coding table (see Table 13-5).
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1083

Table 13-5 PCx Coding

PCx[4:0]	I/O	Characteristics	Selected Pull-up / Pull-down / Selected Output Function
10000_B	Output	Push-pull	General-purpose output
10001_B			Alternate output function 1
10010_B			Alternate output function 2
10011_B			Alternate output function 3
10100_B			Alternate output function 4
10101_B			Alternate output function 5
10110_B			Alternate output function 6
10111_B			Alternate output function 7
11000_B		Open-drain	General-purpose output
11001_B			Alternate output function 1
11010_B			Alternate output function 2
11011_B			Alternate output function 3
11100_B			Alternate output function 4
11101_B			Alternate output function 5
11110_B			Alternate output function 6
11111_B			Alternate output function 7

1) This is the default pull device setting after reset for powertrain applications.

what?
- 출력 모드로 사용하기 위해
0x10 값을 PC7 영역에 write

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1090

Lab1: GPIO 레지스터 설정 – P10 Write Value

: Port 02의 7번 핀을 출력 모드로 사용

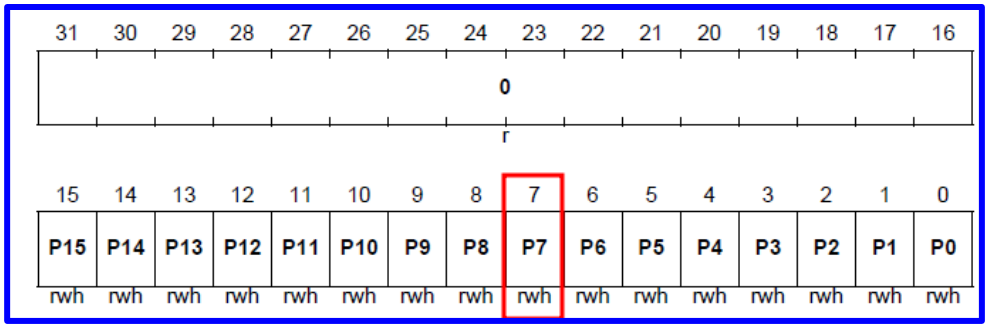
```
268 void initRGBLED(void)
269 {
270     // reset Port IOCR register
271     P02_IOCR4.U &= ~(0x1F << PC7_BIT_LSB_IDX);
272     P10_IOCR4.U &= ~(0x1F << PC5_BIT_LSB_IDX);
273     P10_IOCR0.U &= ~(0x1F << PC3_BIT_LSB_IDX);
274
275     // set Port as general purpose output (push-pull)
276     P02_IOCR4.U |= 0x10 << PC7_BIT_LSB_IDX;
277     P10_IOCR4.U |= 0x10 << PC5_BIT_LSB_IDX;
278     P10_IOCR0.U |= 0x10 << PC3_BIT_LSB_IDX;
279 }
```

GPIO 레지스터 설정 – P02 Write Value

: Port 02의 핀에 출력할 값 설정

- **P02.7** 출력에 HIGH(“1”) 또는 LOW(“0”)의 값을 인가하기 위해 P02_OUT 레지스터에 어떤 값을 써야 하는지 확인

P02_OUT 레지스터 @ 0xF003A200



Field	Bits	Type	Description
Px **(x = 0-15)**	x	rwh	**Port n Output Bit x** This bit determines the level at the output pin Pn.x if the output is selected as GPIO output. 0_B The output level of Pn.x is 0. 1_B The output level of Pn.x is 1. Pn.x can also be set or cleared by control bits of the Pn_OMSR, Pn_OMCR or Pn_OMR registers.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

where?
- Pin 7 이므로 P7 영역에 write 필요

what?
- 출력하고자 하는 값(“0” 또는 “1”)을 P7 영역에 write

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1103

Lab2: 번갈아가며 RGB LED 색상 변경하는 코드
: main 함수 작성해보기

초기화 함수 호출
initRGBLED()

RGB LED 색상이 돌아가면서 변경되는 코드
RGB의 조합으로 원하는 색상을 만들어 보기

```
127     IfxCpu_emitEvent(&g_cpuSyncEvent);
128     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
129
130     //initERU();
131     //initCCU60();
132     initLED();
133     initRGBLED();
134     //initButton();
135
136
137     while(1)
138     {
139         for(unsigned int i = 0; i < 100000; i++);
140         P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
141         P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
142         P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
143
144         for(unsigned int i = 0; i < 100000; i++);
145         P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
146         P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
147         P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
148
149         for(unsigned int i = 0; i < 100000; i++);
150         P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
151         P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
152         P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
153
154         for(unsigned int i = 0; i < 100000; i++);
155         P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
156         P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
157         P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
158
159     }
160     return (1);
161 }
162
```

GPIO 레지스터 설정 – P10 Address

: Port 10 영역이 위치한 주소 확인

- 레지스터 설정을 위해 레지스터가 위치한 주소 파악 필요

1. **GPIO (Ports)** 레지스터 영역 찾기

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.227

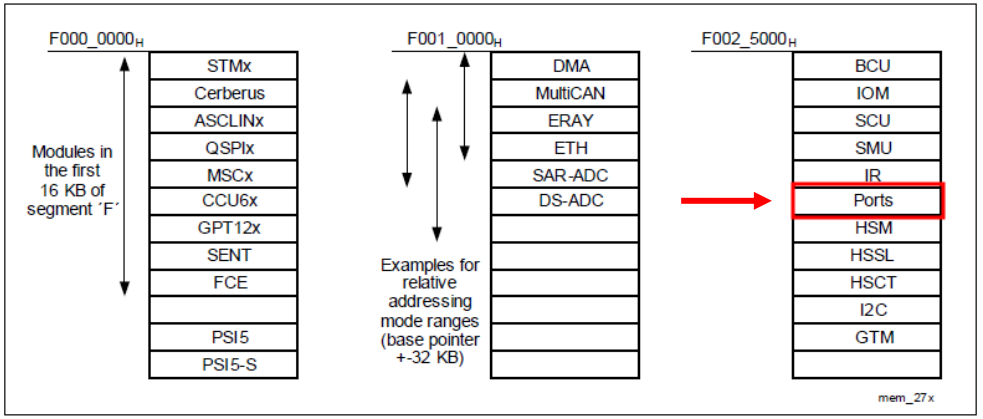


Figure 3-1 Segment F Structure

2. 실습에서 사용하는 Port 10 에 해당하는 레지스터 영역 찾기
   - 시작 주소 (Base address)
   - **= 0xF003B000**

Table 3-3 On Chip Bus Address Map of Segment 15 (cont'd)

Unit	Address Range	Size	Access Type	
			Read	Write
Reserved	$F003\ A300_H$ - $F003\ AFFF_H$	–	SPBBE	SPBBE
Port 10	$F003\ B000_H$ - $F003\ B0FF_H$	256 byte	access	access
Port 11	$F003\ B100_H$ - $F003\ B1FF_H$	256 byte	access	access

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.231

GPIO 레지스터 설정 – P10 Address 계산

: Port 10의 각 레지스터가 위치한 주소 확인

3. 사용할 특정 레지스터의 주소 찾기

– **P10_OUT** Offset Address = 0x0000

→ P10_OUT 레지스터 주소

→ = 0xF003B000 + 0x0000 = **0xF003B000**

– **P10_IOCR0** Offset Address = 0x0010

→ P10_IOCR0 레지스터 주소

→ = 0xF003B000 + 0x0010 = **0xF003B010**

– **P10_IOCR4** Offset Address = 0x0014

→ P10_IOCR4 레지스터 주소

→ = 0xF003B000 + 0x0014 = **0xF003B014**

Table 13-4 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode Read	Access Mode Write	Reset	Desc. see
Pn_OUT	Port n Output Register	0000_H	U, SV	U, SV, P	Application Reset	Page 13-38
Pn_OMR	Port n Output Modification Register	0004_H	U, SV	U, SV, P	Application Reset	Page 13-39
ID	Module Identification Register	0008_H	U, SV	BE	Application Reset	Page 13-13
Pn_IOCR0	Port n Input/Output Control Register 0	0010_H	U, SV	U, SV, P	Application Reset	Page 13-14
Pn_IOCR4	Port n Input/Output	0014_H	U, SV	U,	Application	Page 13-1

Table 13-4 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode Read	Access Mode Write	Reset	Desc. see
Pn_OUT	Port n Output Register	0000_H	U, SV	U, SV, P	Application Reset	Page 13-38
Pn_OMR	Port n Output Modification Register	0004_H	U, SV	U, SV, P	Application Reset	Page 13-39
ID	Module Identification Register	0008_H	U, SV	BE	Application Reset	Page 13-13
Pn_IOCR0	Port n Input/Output Control Register 0	0010_H	U, SV	U, SV, P	Application Reset	Page 13-14
Pn_IOCR4	Port n Input/Output	0014_H	U, SV	U,	Application	Page 13-1

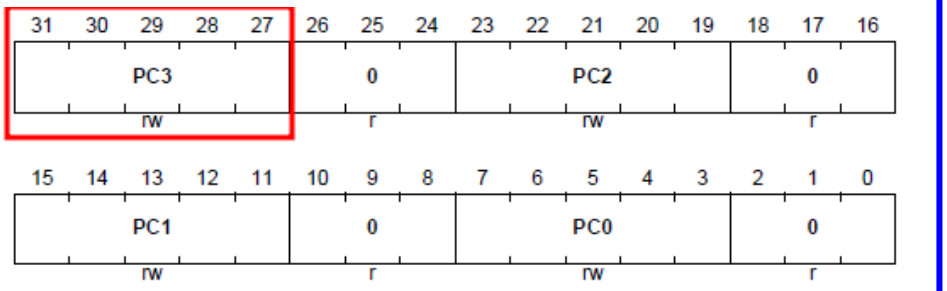
ID	Module Identification Register	0008_H	U, SV	BE	Application Reset	Page 13-13
Pn_IOCR0	Port n Input/Output Control Register 0	0010_H	U, SV	U, SV, P	Application Reset	Page 13-14
Pn_IOCR4	Port n Input/Output Control Register 4	0014_H	U, SV	U, SV, P	Application Reset	Page 13-14
Pn_IOCR8	Port n Input/Output Control Register 8	0018_H	U, SV	U, SV, P	Application Reset	Page 13-14

GPIO 레지스터 설정 – P10 Write Value

: Port 10의 핀 3을 출력 모드로 사용

- **P10.3** 을 General Purpose Output (push-pull)으로 사용하기 위해 P10_IOCR0 레지스터에 어떤 값을 써야 하는지 확인 → **0x10 (2진수 10000b)**

P10_IOCR0 레지스터 @ 0xF003B010



Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 0 to 3** This bit field determines the Port n line x functionality (x = 0-3) according to the coding table (see Table 13-5).
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1080

Table 13-5 PCx Coding

PCx[4:0]	I/O	Characteristics	Selected Pull-up / Pull-down / Selected Output Function
10000_B	Output	Push-pull	General-purpose output
10001_B			Alternate output function 1
10010_B			Alternate output function 2
10011_B			Alternate output function 3
10100_B			Alternate output function 4
10101_B			Alternate output function 5
10110_B			Alternate output function 6
10111_B			Alternate output function 7
11000_B		Open-drain	General-purpose output
11001_B			Alternate output function 1
11010_B			Alternate output function 2
11011_B			Alternate output function 3
11100_B			Alternate output function 4
11101_B			Alternate output function 5
11110_B			Alternate output function 6
11111_B			Alternate output function 7

1) This is the default pull device setting after reset for powertrain applications.

what?
- 출력 모드로 사용하기 위해
0x10 값을 PC3 영역에 write

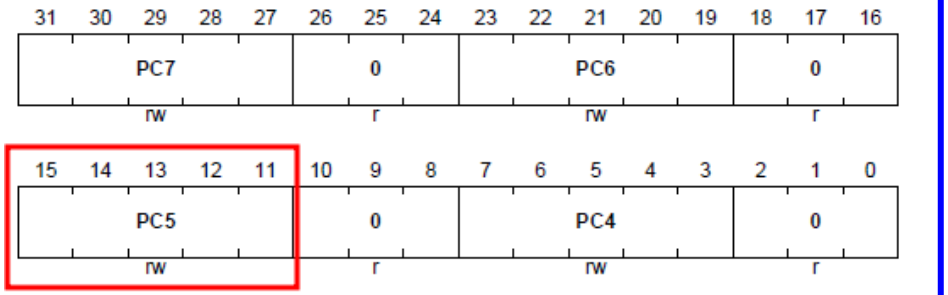
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1090

GPIO 레지스터 설정 – P10 Write Value

: Port 10의 핀 5를 출력 모드로 사용

- **P10.5** 을 General Purpose Output (push-pull)으로 사용하기 위해 P10_IOCR4 레지스터에 어떤 값을 써야 하는지 확인 → **0x10 (2진수 10000b)**

P10_IOCR4 레지스터 @ 0xF003B014



Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 4 to 7** This bit field determines the Port n line x functionality (x = 4-7) according to the coding table (see Table 13-5).
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1083

Table 13-5 PCx Coding

PCx[4:0]	I/O	Characteristics	Selected Pull-up / Pull-down / Selected Output Function
10000_B	Output	Push-pull	General-purpose output
10001_B			Alternate output function 1
10010_B			Alternate output function 2
10011_B			Alternate output function 3
10100_B			Alternate output function 4
10101_B			Alternate output function 5
10110_B			Alternate output function 6
10111_B			Alternate output function 7
11000_B		Open-drain	General-purpose output
11001_B			Alternate output function 1
11010_B			Alternate output function 2
11011_B			Alternate output function 3
11100_B			Alternate output function 4
11101_B			Alternate output function 5
11110_B			Alternate output function 6
11111_B			Alternate output function 7

1) This is the default pull device setting after reset for powertrain applications.

what?
- 출력 모드로 사용하기 위해
0x10 값을 PC5 영역에 write

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1090

Lab3: GPIO 레지스터 설정 – P10 Write Value

: Port 10의 핀3, 5를 출력 모드로 사용

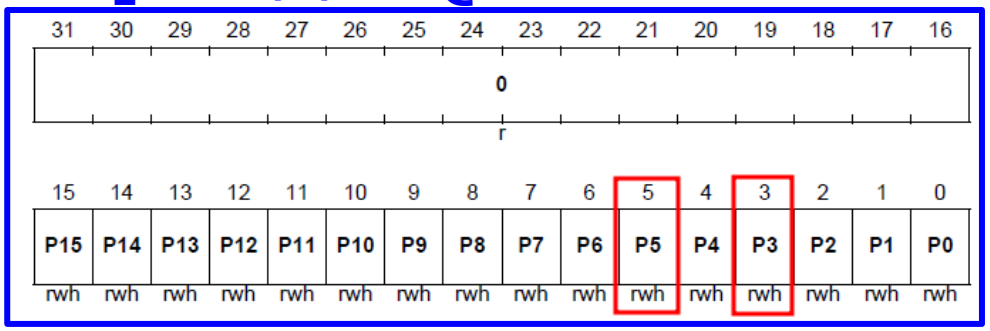
```
268 void initRGBLED(void)
269 {
270     // reset Port IOCR register
271     P02_IOCR4.U &= ~(0x1F << PC7_BIT_LSB_IDX);
272     P10_IOCR4.U &= ~(0x1F << PC5_BIT_LSB_IDX);
273     P10_IOCR0.U &= ~(0x1F << PC3_BIT_LSB_IDX);
274
275     // set Port as general purpose output (push-pull)
276     P02_IOCR4.U |= 0x10 << PC7_BIT_LSB_IDX;
277     P10_IOCR4.U |= 0x10 << PC5_BIT_LSB_IDX;
278     P10_IOCR0.U |= 0x10 << PC3_BIT_LSB_IDX;
279 }
```

GPIO 레지스터 설정 – P10 Write Value

: Port 10의 핀에 출력할 값 설정

- **P10.3, P10.5** 출력에 HIGH(“1”) 또는 LOW(“0”)의 값을 인가하기 위해 P10_OUT 레지스터에 어떤 값을 써야 하는지 확인

P10_OUT 레지스터 @ 0xF003B000



Field	Bits	Type	Description
Px **(x = 0-15)**	x	rwh	**Port n Output Bit x** This bit determines the level at the output pin Pn.x if the output is selected as GPIO output. 0_B The output level of Pn.x is 0. 1_B The output level of Pn.x is 1. Pn.x can also be set or cleared by control bits of the Pn_OMSR, Pn_OMCR or Pn_OMR registers.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

where?
- Pin 3 과 Pin 5 이므로 P3, P5 영역에 write 필요

what?
- 출력하고자 하는 값(“0” 또는 “1”)을 P3, P5 영역에 write

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1103

Lab4: 번갈아가며 RGB LED 색상 변경하는 코드
: main 함수 작성해보기

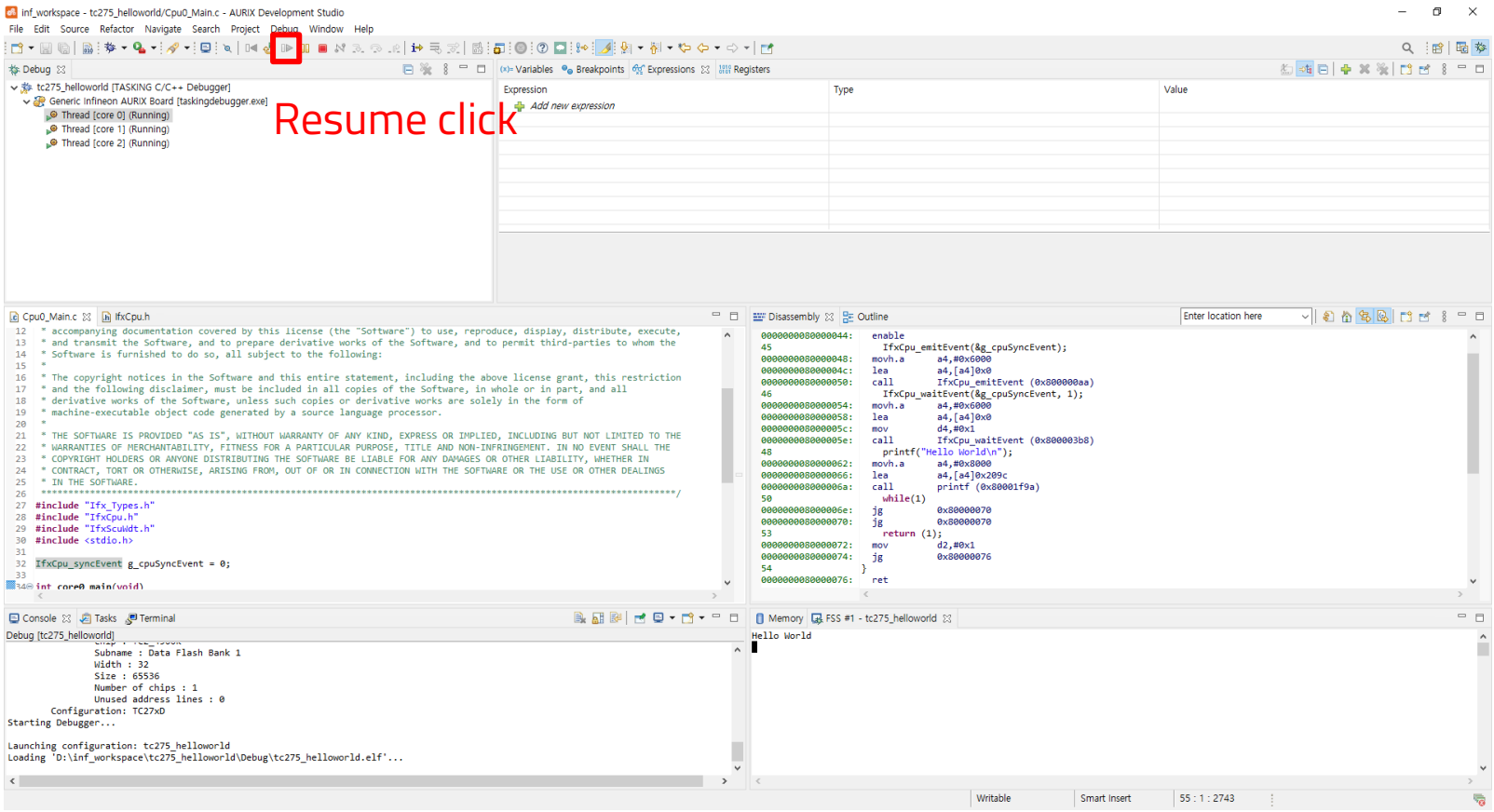
초기화 함수 호출
initRGBLED()

RGB LED 색상이 돌아가면서 변경되는 코드
RGB의 조합으로 원하는 색상을 만들어 보기

```
127     IfxCpu_emitEvent(&g_cpuSyncEvent);
128     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
129
130     //initERU();
131     //initCCU60();
132     initLED();
133     initRGBLED();
134     //initButton();
135
136
137     while(1)
138     {
139         for(unsigned int i = 0; i < 100000; i++);
140         P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
141         P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
142         P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
143
144         for(unsigned int i = 0; i < 100000; i++);
145         P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
146         P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
147         P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
148
149         for(unsigned int i = 0; i < 100000; i++);
150         P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
151         P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
152         P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
153
154         for(unsigned int i = 0; i < 100000; i++);
155         P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
156         P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
157         P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
158
159     }
160     return (1);
161 }
162
```

Build 및 Debug

- 프로젝트 빌드 (ctrl + b)
- 디버그 수행하여 보드에 실행 파일 flash

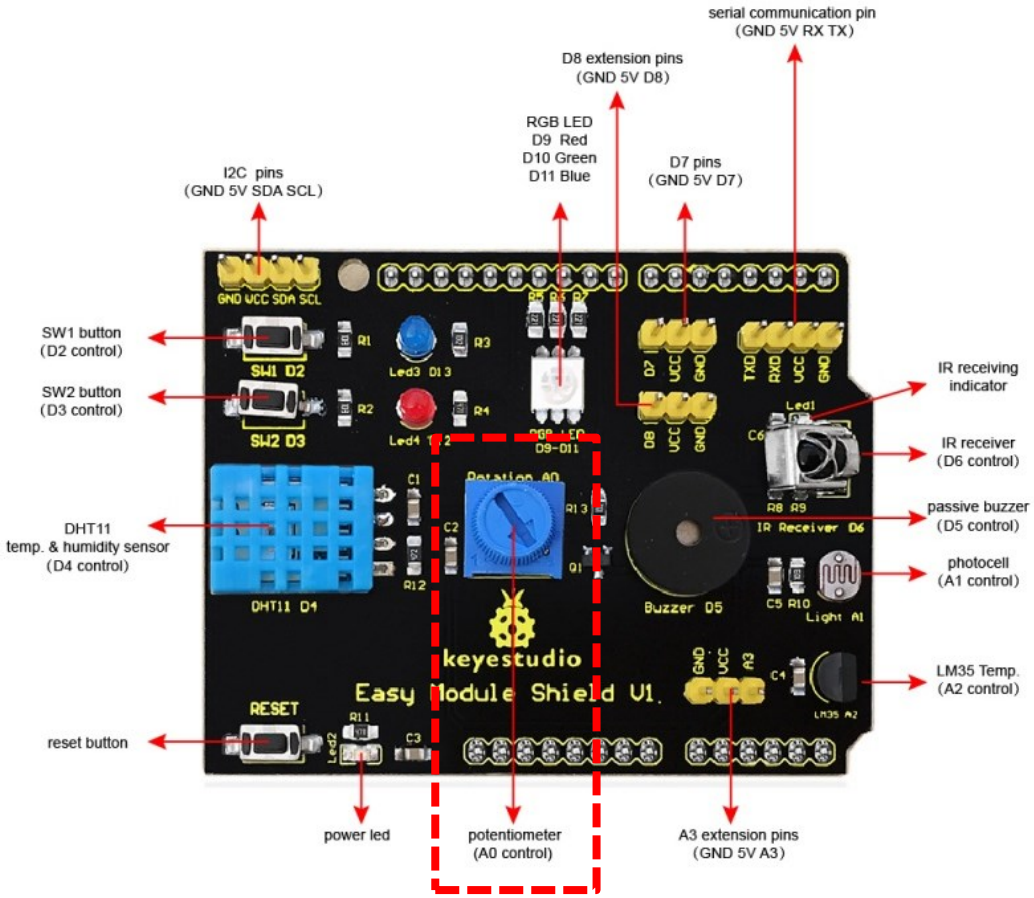


ADC 회로 구조 파악

: 확장 보드의 ADC는 어떤 핀을 사용?

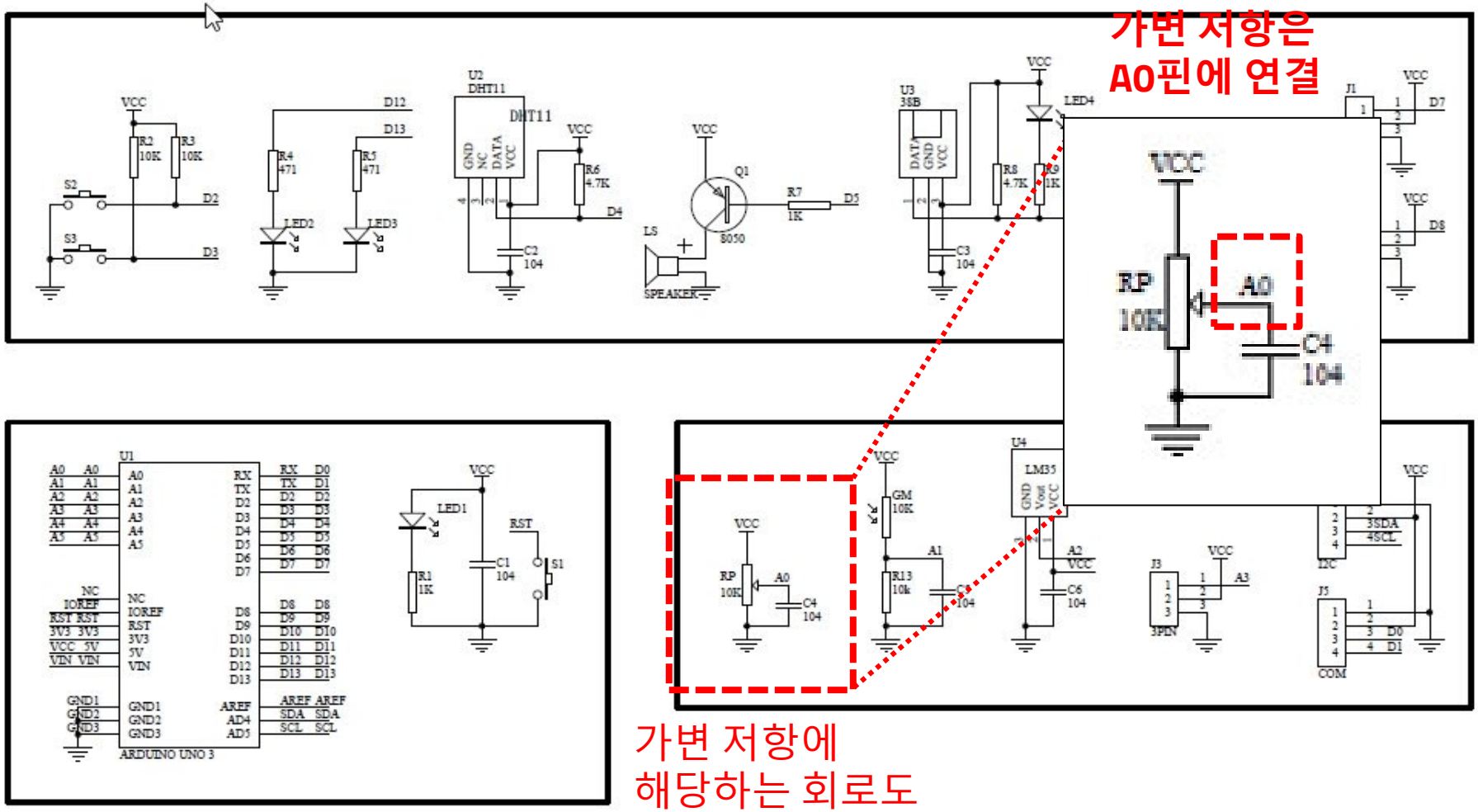
- 확장 보드의 가변 저항을 사용하기 위해서는 TC275 보드의 어떤 핀이 Easy Module Shield 확장 보드의 가변 저항과 연결되어 있는지 알아야 함

  → 데이터 시트 분석 필요



회로도 @ 확장 보드 데이터 시트

: 가변 저항 회로 부분 확인



Pin Map @ TC275 보드

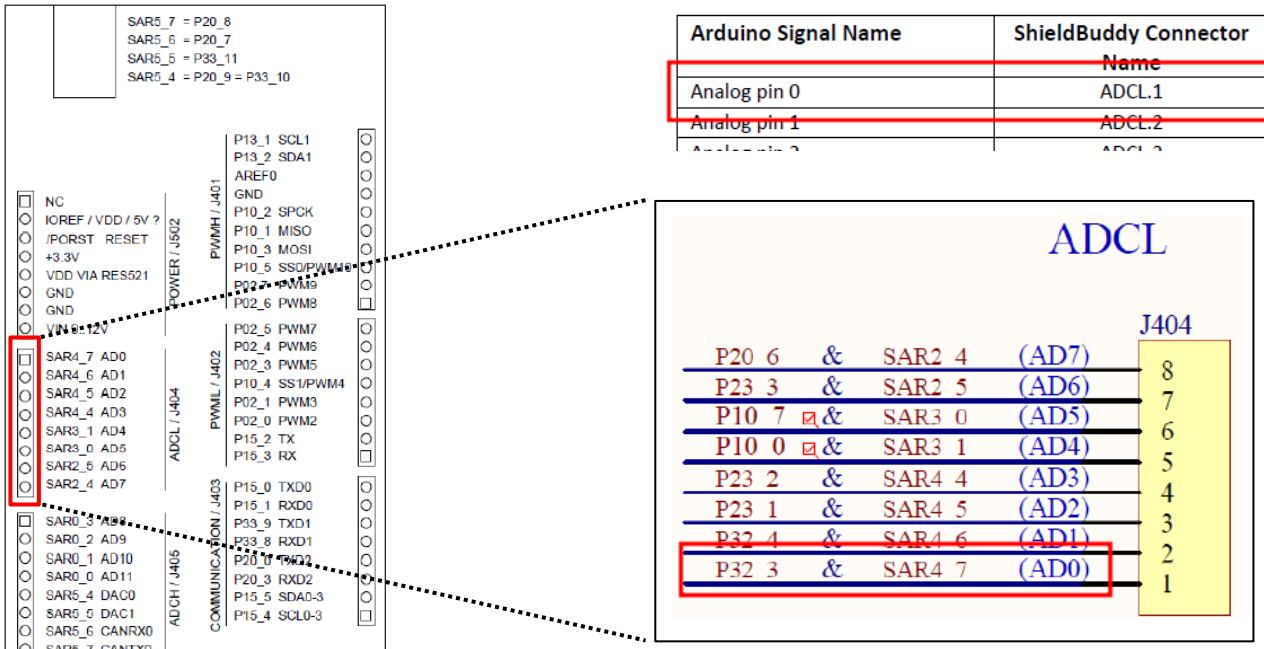
: 가변 저항 핀과 TC275 보드 핀과의 회로 연결 정보 확인

- 앞서 가변 저항은 확장 보드의 핀 A0에 연결 되어있는 것을 확인

→ 그렇다면 확장 보드의 핀 **A0**는 **TC275** 보드의 어느 핀에 연결?

Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.44

Arduino Signal Name	ShieldBuddy Connector Name	TC275T Pin Assignment
Analog pin 0	ADCL.1	SAR4.7/P32.3
Analog pin 1	ADCL.2	SAR4.6/P32.4



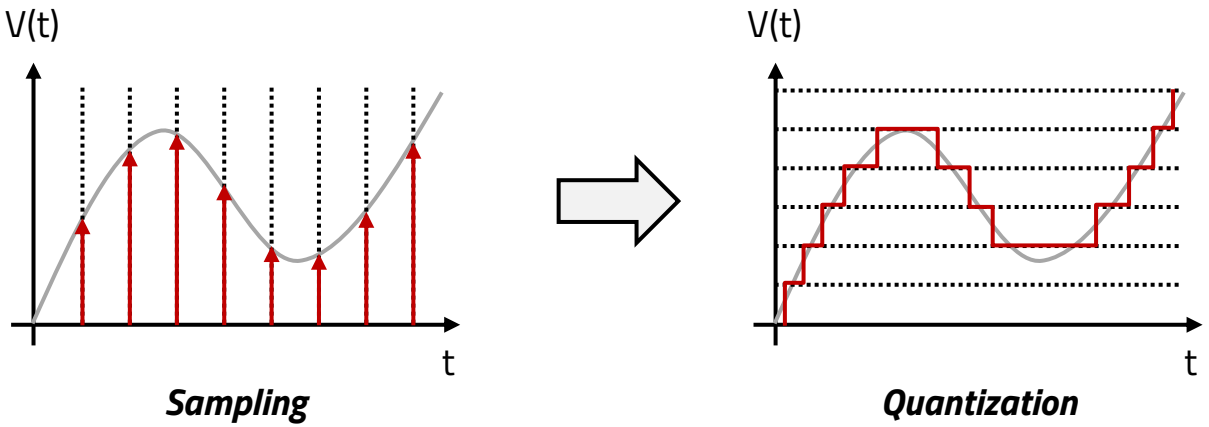
TC275 MCU I/O 중,
SAR4_7
과 연결됨

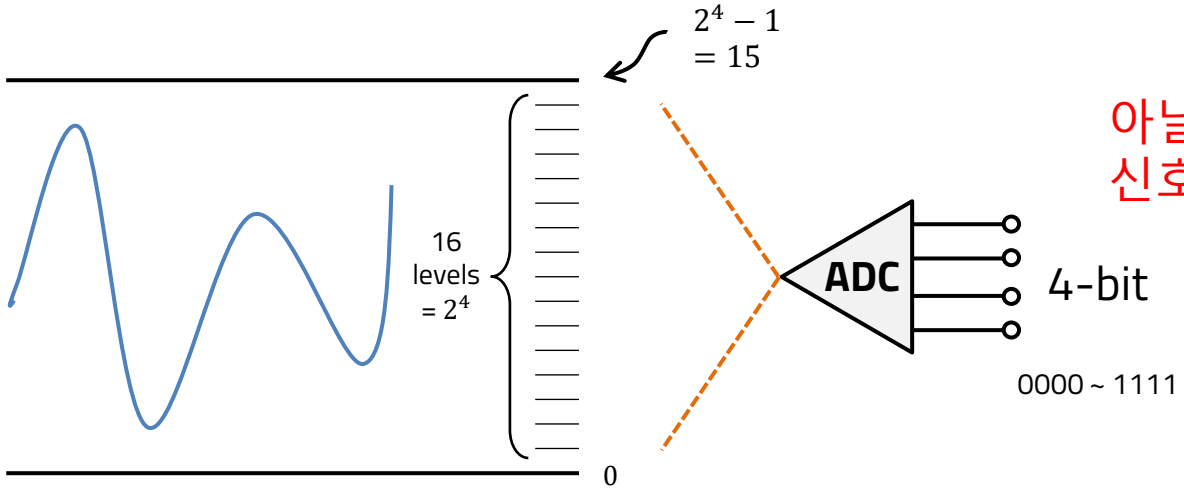
Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.42

Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.46

VADC (Versatile Analog-to-Digital Converter)

- MCU 칩 외부 환경에서 입력되는 아날로그 신호들을 디지털 도메인으로 변환





아날로그 신호를 4bit 디지털 신호로 변환하는 예시

VADC 변환 flow 및 원리
: VADC 모듈 구성 및 변환 수행

- VADC는 여러 개의 Group / Kernel 로 구성됨

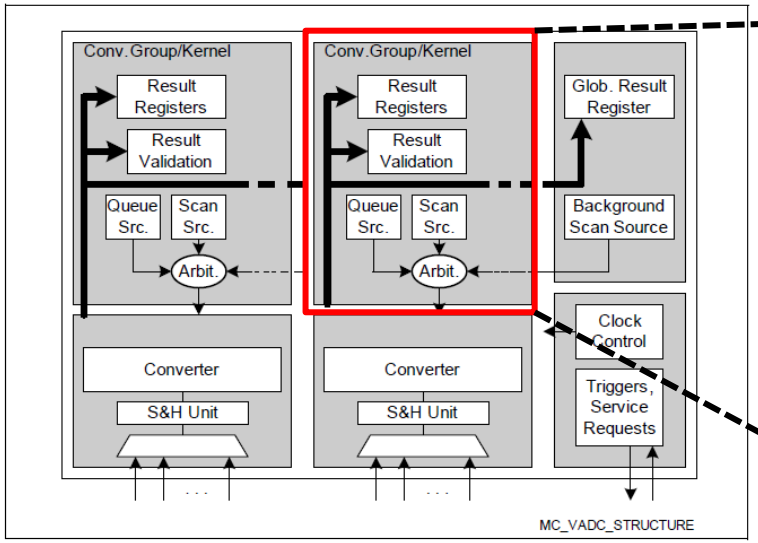
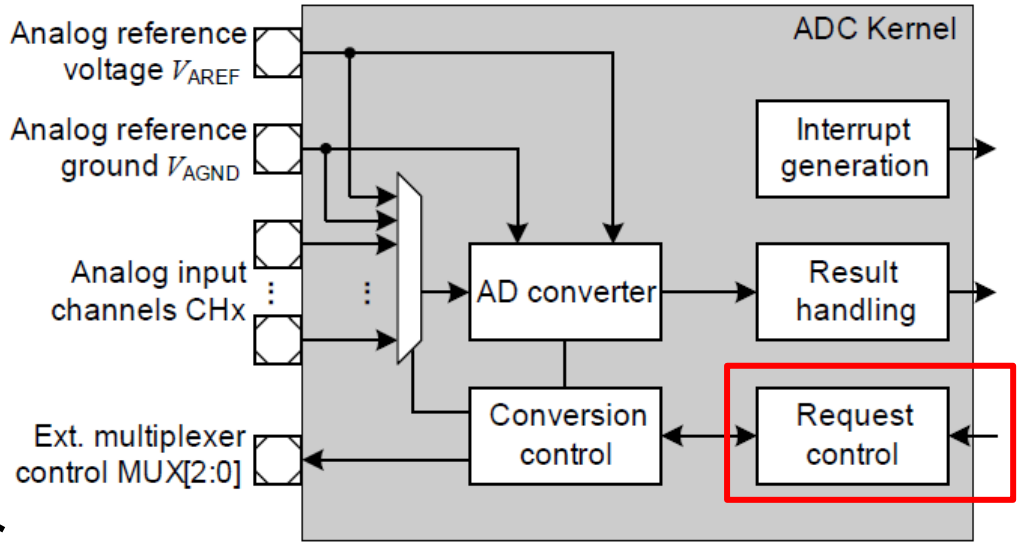


Figure 28-1 ADC Structure Overview

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3875

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3878



ADC Kernel 로 입력되는 Conversion Request 에 따라서 Conversion 수행

VADC 사용을 위한 레지스터 설정
: VADC모듈 사용 전 보호 유닛 해제 필요

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3875

- VADC 모듈 사용을 위해 Clock Control 레지스터에서 **VADC 모듈 enable 필요**

- VADC 레지스터 항목에서
  - **CLC** 레지스터 설정 필요

- VADC_CLC 레지스터는 System Critical 레지스터이므로 CPU ENDINIT 해제 후 수정해야함
- (CPU0 만을 사용하므로) SCU 레지스터 항목에서
  - **WDTCPU0CON0** 레지스터 설정 필요

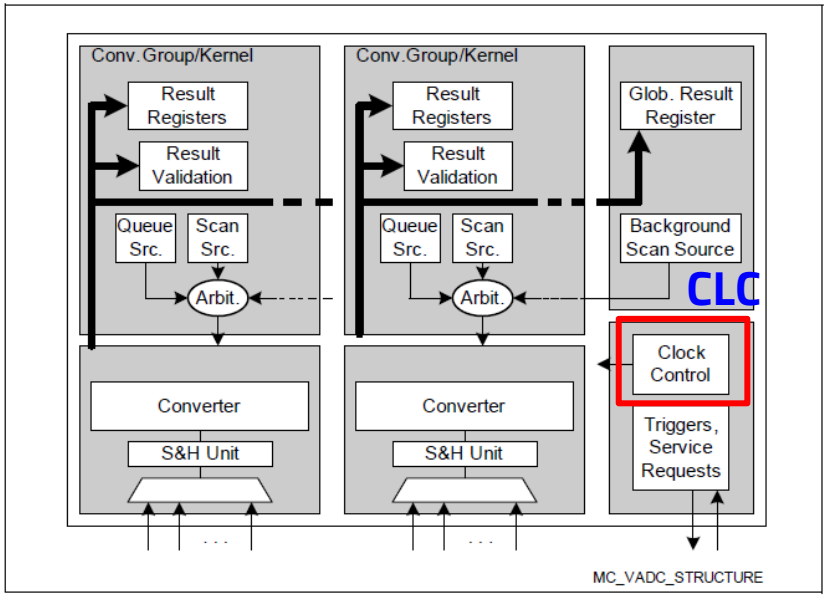


Figure 28-1 ADC Structure Overview

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.650

- "CE0"- writeable only when CPU0 ENDINIT bit is zero
- "CE1" - writeable only when CPU1 ENDINIT bit is zero
- "CE2" - writeable only when CPU2 ENDINIT bit is zero
- "E" - writeable when any (one or more) CPUx ENDINIT bit is zero
- "SE" - writeable only when Safety ENDINIT bit is zero
- None of the above - accessible at any time

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4022

Table 28-11 Registers Overview

Register Short Name	Register Long Name	Offset Addr.	Access Mode Read	Access Mode Write	Page Num.
ID	Module Identification Register	0008_H	U, SV	BE	28-14
CLC	Clock Control Register	0000_H	U, SV	SV, E, P	28-15
OCS	OCDS Control and Status Register	0028_H	U, SV	SV P	28-16

SCU 레지스터 설정 – WDTCPU0CON0

: VADC 모듈 사용 설정 과정을 보호하는 레지스터

1. SCU 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address)
   - **= 0xF0036000**

Reserved	F003 5200$_H$ - F003 5FFF$_H$	–	SPBBE	SPBBE
System Control Unit (SCU)	F003 6000$_H$ - F003 63FF$_H$	Kbyte	access	access
Reserved	F003 6400$_H$ F003 67FF$_H$	–	SPBBE	SPBBE
Safety Management Unit (SMU)	F003 6800	2	access	access

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.230

2. 사용할 레지스터의 주소 찾기
   - **WDTCPU0CON0** 의 Offset Address = **0x100**

→ WDTCPU0CON0 레지스터 주소 = 0xF0036000 + 0x100 = **0xF0036100**

Table 7-28 Register Overview of SCU (Offset from Main Register Base)

Short Name	Long Name	Offset Addr. 1)	Access Mode Read	Access Mode Write	Reset	Description See
EMSR	Emergency Stop Register	0FC$_H$	U, SV	SV, SE, P	Application Reset	Page 7-291
WDTCPU0CON0	CPU0 WDT Control Register 0	100$_H$	U, SV	U, SV, 32,(CPU 0[2)])	Application Reset	Page 7-276
WDTCPU0CON1	CPU0 WDT Control	104$_H$	U, SV	SV,	Application	Page

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.697

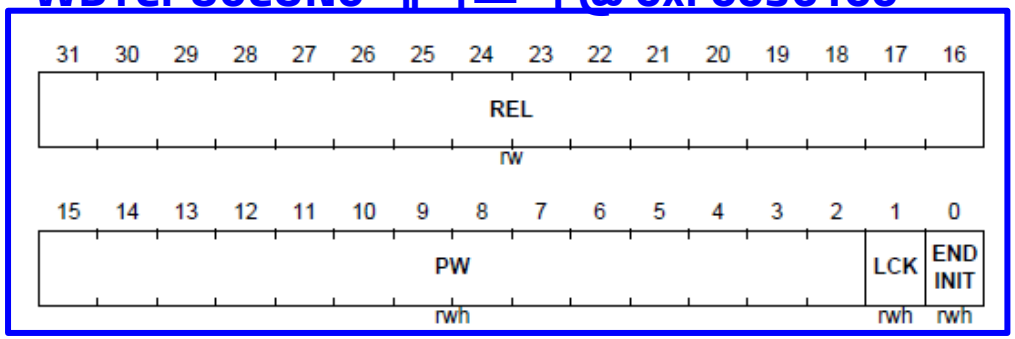
SCU 레지스터 설정 – WDTCPU0CON0

: VADC 모듈 사용하도록 설정 위해 보호 레지스터(ENDINIT) 잠금 해제

3. Password Access 를 통해 WDTCPU0CON0 레지스터의 Lock 상태를 해제해야 함

1) WDTCPU0CON0 레지스터를 읽어 **WDTCPU0CON0.REL, WDTCPU0CON0.PW** 영역의 값을 확인

WDTCPU0CON0 레지스터 @ 0xF0036100



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.661

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.662

PW	[15:2]	rwh	**User-Definable Password Field for Access to WDTxCON0** This bit field is written with an initial password value during a Modify Access. A read from this bitfield returns this initial password, but bits [7:2] are inverted (toggled) to ensure that a simple read/write is not sufficient to service the WDT.

단, PW 영역의 일부 PW[7:2] 는 반전해서 읽어야 함

2) Write 할 값은 1)에서 읽은 REL 과 PW 의 조합으로 결정

[31:16] = REL, [15:2] = PW, [1] = “0”, [0] = “1”

3) 결정한 32bit 값을 WDTCPU0CON0 레지스터에 한 번에 write

4) WDTCPU0CON0 레지스터의 1번째 bit LCK 를 읽어서 Lock 상태가 해제되었는지 확인

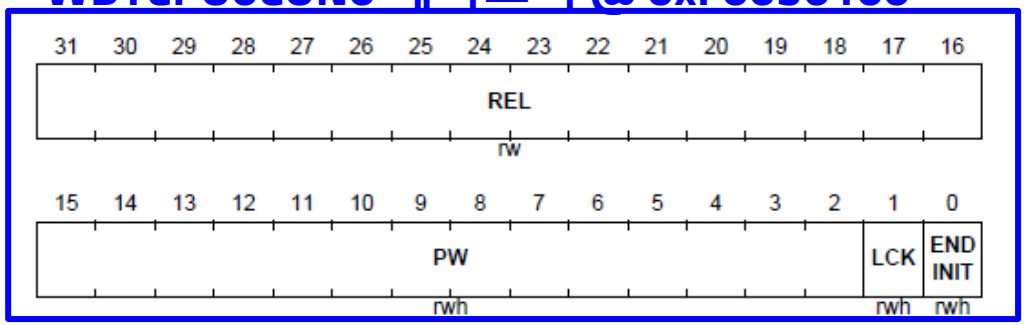
Lock 상태가 해제되면 LCK bit 가 0으로 읽힘

SCU 레지스터 설정 – WDTCPU0CON0

: 보호 레지스터 ENDINIT 해제후 다시 잠금 설정

4. **Modify Access** 를 통해 WDTCPU0CON0 레지스터의 CPU0 ENDINIT을 set/clear 함
   1) WDTCPU0CON0 레지스터를 읽어 **WDTCPU0CON0.REL, WDTCPU0CON0.PW** 영역의 값을 확인

WDTCPU0CON0 레지스터 @ 0xF0036100



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.661

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.662

PW	[15:2]	rwh	**User-Definable Password Field for Access to WDTxCON0** This bit field is written with an initial password value during a Modify Access. ~~A read from this bitfield returns this initial password, but bits [7:2] are inverted (toggled) to ensure that a simple read/write is not sufficient to service the WDT.~~

단, PW 영역의 일부 PW[7:2] 는 반전해서 읽어야 함

   2) Write 할 값은 1)에서 읽은 REL 과 PW 의 조합으로 결정
      [31:16] = REL, [15:2] = PW, [1] = "1"
   3) 0번째 bit **ENDINIT을 설정하려면 [0] = "1", 해제하려면 "0" 값을 write**
   4) 결정한 32bit 값을 WDTCPU0CON0 레지스터에 한 번에 write
   5) WDTCPU0CON0 레지스터의 1번째 bit LCK 를 읽어서 Lock 상태가 설정되었는지 확인
      Lock 상태가 설정되면 LCK bit 가 1로 읽힘

5. **Modify Access 를 통해 CPU0 ENDINIT을 해제하면 레지스터 수정 후 반드시 CPU0 ENDINIT을 재설정해줘야 함**

Lab5: SCU 레지스터 설정 – WDTCPU0CON0
: VADC 모듈 사용하도록 설정 위해 보호 레지스터 ENDINIT Clear

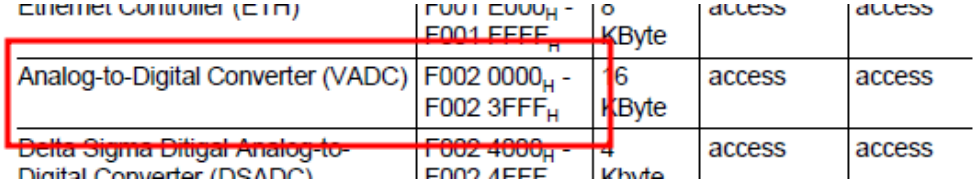
```
280
281 void initVADC(void)
282 {
283     // Password Access to unlock SCU_WDTSCON0
284     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
285     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
286
287     // Modify Access to clear ENDINIT
288     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
289     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
290
291     VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable VADC
292
293     // Password Access to unlock SCU_WDTSCON0
294     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
295     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
296
297     // Modify Access to set ENDINIT
298     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
299     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
```

Lock비트 해제

ENDINIT Clear후 다시 Lock

VADC 레지스터 설정 – CLC
: VADC 모듈 사용 설정 레지스터

1. VADC 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address)
   - **= 0xF0020000**



Ethernet Controller (ETH)	$F001\ E000_H$ - $F001\ FFFF_H$	8 KByte	access	access
Analog-to-Digital Converter (VADC)	$F002\ 0000_H$ - $F002\ 3FFF_H$	16 KByte	access	access
Delta Sigma Ditigal Analog-to-Digital Converter (DSADC)	$F002\ 4000_H$ - $F002\ 4FFF$	4 Kbyte	access	access

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.230

2. 사용할 레지스터의 주소 찾기
   - **CLC** 의 Offset Address = **0x00**

   → CLC 레지스터 주소 = 0xF0020000 + 0x00 = **0xF0020000**

Table 28-11 Registers Overview

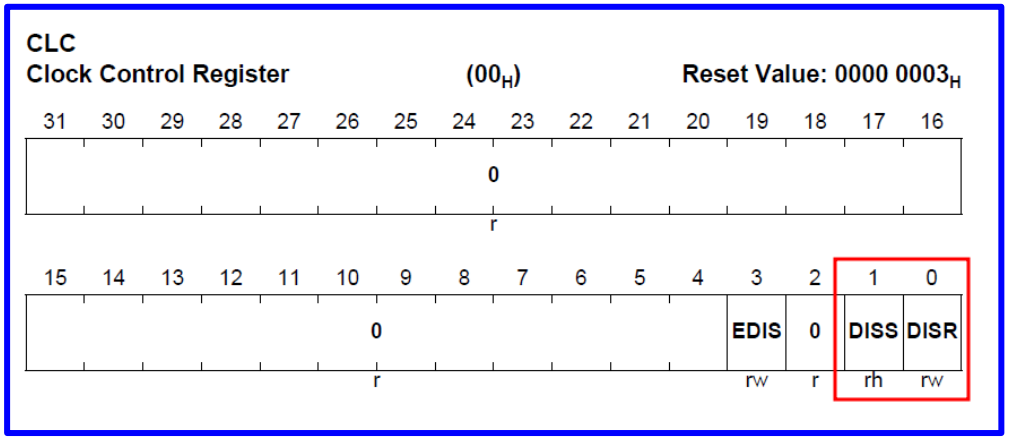
Register Short Name	Register Long Name	Offset Addr.	Access Mode Read	Access Mode Write	Page Num.
ID	Module Identification Register	0008_H	U, SV	BE	28-14
CLC	Clock Control Register	0000_H	U, SV	SV E, P	28-15
OCS	OCDS Control and Status Register	0028_H	U, SV	SV P	28-16

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4022

VADC 레지스터 설정 – CLC
: VADC 모듈 사용하도록 설정

3. 레지스터 write 값 결정
   - VADC 모듈을 enable 하기 위해 **DISR 영역에 0x0 write**
   - VADC모듈의 **DISS 영역의 값을 읽었을 때 "0"이면 모듈이 enable 되었음**
     - 만약 읽었을 때 "1"이라면 모듈이 **disable** 되어있다는 뜻

CLC 레지스터 @ 0xF0020000



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. Also the analog section is disabled by clearing ANONS. 0_B On request: enable the module clock 1_B Off request: stop the module clock
DISS	1	r	**Module Disable Status Bit** 0_B Module clock is enabled 1_B Off: module is not clocked

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3889

Lab6: 헤더 작성

: VADC 레지스터의 각 영역(필드) LSB 비트 시작 위치 define 정의

- 레지스터에 값을 write할 때 shift되는 offset을 쉽게 사용하기 위한 define 작성

```
30
31  #include "IfxCcu6_reg.h"
32  #include "IfxVadc_reg.h"
33
34  // Port registers
35  #define PC1_BIT_LSB_IDX             11
36  #define PC2_BIT_LSB_IDX             19
37  #define PC3_BIT_LSB_IDX             27
38  #define PC5_BIT_LSB_IDX             11
39  #define PC7_BIT_LSB_IDX             27
40  #define P1_BIT_LSB_IDX              1
41  #define P2_BIT_LSB_IDX              2
42  #define P3_BIT_LSB_IDX              3
43  #define P5_BIT_LSB_IDX              5
44  #define P7_BIT_LSB_IDX              7
45
46  // SCU registers
47  #define LCK_BIT_LSB_IDX             1
48  #define ENDINIT_BIT_LSB_IDX         0
49  #define EXIS0_BIT_LSB_IDX           4
50  #define FEN0_BIT_LSB_IDX            8
51  #define EIEN0_BIT_LSB_IDX           11
52  #define INP0_BIT_LSB_IDX            12
53  #define IGP0_BIT_LSB_IDX            14
54
55  // SRC registers
56  #define SRPN_BIT_LSB_IDX            0
57  #define TOS_BIT_LSB_IDX             11
58  #define SRE_BIT_LSB_IDX             10
59
60  // CCU60 registers
61  #define DISS_BIT_LSB_IDX            1
62  #define DISR_BIT_LSB_IDX            0
63  #define CTM_BIT_LSB_IDX             7
64  #define T12PRE_BIT_LSB_IDX          3
65  #define T12CLK_BIT_LSB_IDX          0
66  #define T12STR_BIT_LSB_IDX          6
67  #define T12RS_BIT_LSB_IDX           1
68  #define INPT12_BIT_LSB_IDX          10
69  #define ENT12PM_BIT_LSB_IDX         7
70
```

헤더 파일 참조 추가

Port 레지스터 bit shift offset

```
74  #define ANONC_BIT_LSB_IDX           0
75  #define ASEN0_BIT_LSB_IDX           24
76  #define CSM0_BIT_LSB_IDX            3
77  #define PRIO0_BIT_LSB_IDX           0
78  #define CMS_BIT_LSB_IDX             8
79  #define FLUSH_BIT_LSB_IDX           10
80  #define TREV_BIT_LSB_IDX            9
81  #define ENGT_BIT_LSB_IDX            0
82  #define RESPOS_BIT_LSB_IDX          21
83  #define RESREG_BIT_LSB_IDX          16
84  #define ICLSEL_BIT_LSB_IDX          0
85  #define VF_BIT_LSB_IDX              31
86  #define RESULT_BIT_LSB_IDX          0
87  #define ASSCH7_BIT_LSB_IDX          7
88
```

VADC 레지스터 bit shift offset

Lab7: VADC 레지스터 설정 – CLC
: VADC 모듈 사용하도록 설정

```
280
281 void initVADC(void)
282 {
283     // Password Access to unlock SCU_WDTSCON0
284     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
285     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
286
287     // Modify Access to clear ENDINIT
288     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
289     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
290
291     VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable VADC
292
293     // Password Access to unlock SCU_WDTSCON0
294     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
295     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
296
297     // Modify Access to set ENDINIT
298     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
299     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
300
301
302     // VADC configurations
303     while((VADC_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until VADC module enabled
304
305     VADC_G4_ARBPR.U |= 0x3 << PRIO0_BIT_LSB_IDX;         // highest priority for Request Source 0
306     VADC_G4_ARBPR.U &= ~(0x1 << CSM0_BIT_LSB_IDX);       // Wait-for-Start Mode
307     VADC_G4_ARBPR.U |= 0x1 << ASEN0_BIT_LSB_IDX;         // Arbitration Source Input 0 Enable
308
309     VADC_G4_QMR0.U &= ~(0x3 << ENGT_BIT_LSB_IDX);
310     VADC_G4_QMR0.U |= 0x1 << ENGT_BIT_LSB_IDX;          // enable conversion request
311     VADC_G4_QMR0.U |= 0x1 << FLUSH_BIT_LSB_IDX;         // clear ADC queue
312
313     VADC_G4_ARBCFG.U |= 0x3 << ANONC_BIT_LSB_IDX;       // ADC normal operation
314
315     VADC_G4_ICLASS0.U &= ~(0x7 << CMS_BIT_LSB_IDX);     // Class 0 Standard Conversion (12-bit)
316
317
318     // VADC Group 4 Channel 7 configuration
319     VADC_G4_CHCTR7.U |= 0x1 << RESPOS_BIT_LSB_IDX;      // result right-aligned
320     VADC_G4_CHCTR7.U &= ~(0xF << RESREG_BIT_LSB_IDX);   // store result @ Result Register G4RES0
321     VADC_G4_CHCTR7.U &= ~(0x3 << ICLSEL_BIT_LSB_IDX);   // Class 0
322
323     VADC_G4_CHASS.U |= 0x1 << ASSCH7_BIT_LSB_IDX;
324 }
325
```

Lab8: SCU 레지스터 설정 – WDTCPU0CON0

: Lock해제 후 ENDINIT Set 한 다음 다시 Lock

```
280
281 void initVADC(void)
282 {
283     // Password Access to unlock SCU_WDTSCON0
284     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
285     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
286
287     // Modify Access to clear ENDINIT
288     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
289     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
290
291     VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable VADC
292
293     // Password Access to unlock SCU_WDTSCON0
294     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
295     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
296
297     // Modify Access to set ENDINIT
298     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
299     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
300
301
302     // VADC configurations
303     while((VADC_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until VADC module enabled
304
305     VADC_G4_ARBPR.U |= 0x3 << PRIO0_BIT_LSB_IDX;         // highest priority for Request Source 0
306     VADC_G4_ARBPR.U &= ~(0x1 << CSM0_BIT_LSB_IDX);       // Wait-for-Start Mode
307     VADC_G4_ARBPR.U |= 0x1 << ASEN0_BIT_LSB_IDX;         // Arbitration Source Input 0 Enable
308
309     VADC_G4_QMR0.U &= ~(0x3 << ENGT_BIT_LSB_IDX);
310     VADC_G4_QMR0.U |= 0x1 << ENGT_BIT_LSB_IDX;          // enable conversion request
311     VADC_G4_QMR0.U |= 0x1 << FLUSH_BIT_LSB_IDX;         // clear ADC queue
312
313     VADC_G4_ARBCFG.U |= 0x3 << ANONC_BIT_LSB_IDX;       // ADC normal operation
314
315     VADC_G4_ICLASS0.U &= ~(0x7 << CMS_BIT_LSB_IDX);     // Class 0 Standard Conversion (12-bit)
316
317
318     // VADC Group 4 Channel 7 configuration
319     VADC_G4_CHCTR7.U |= 0x1 << RESPOS_BIT_LSB_IDX;      // result right-aligned
320     VADC_G4_CHCTR7.U &= ~(0xF << RESREG_BIT_LSB_IDX);   // store result @ Result Register G4RES0
321     VADC_G4_CHCTR7.U &= ~(0x3 << ICLSEL_BIT_LSB_IDX);   // Class 0
322
323     VADC_G4_CHASS.U |= 0x1 << ASSCH7_BIT_LSB_IDX;
324 }
325
```

Lock비트 해제

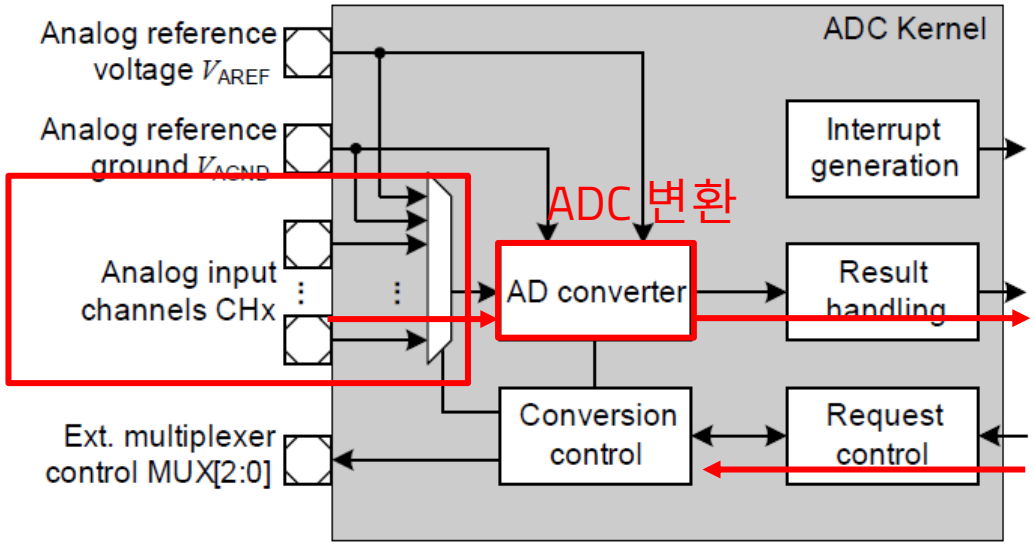
ENDINIT Set 후에 다시 Lock

VADC 변환 flow 및 원리

: ADC 변환 과정에 관여하는 Unit

- ADC Kernel 에 Conversion Request 가 입력되면, 아날로그 입력 Channel 중에서 변환 대상을 선택

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3878



(3) Analog Input Channel 선택

(2) ADC 변환 결과는 Result Handling으로 전달
저장될 위치 설정 필요

(1) Conversion Request 입력

VADC 사용을 위한 레지스터 설정
: 확장보드 아날로그 입력 연결되는 핀 확인

- **SAR4_7** 핀의 의미

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.227

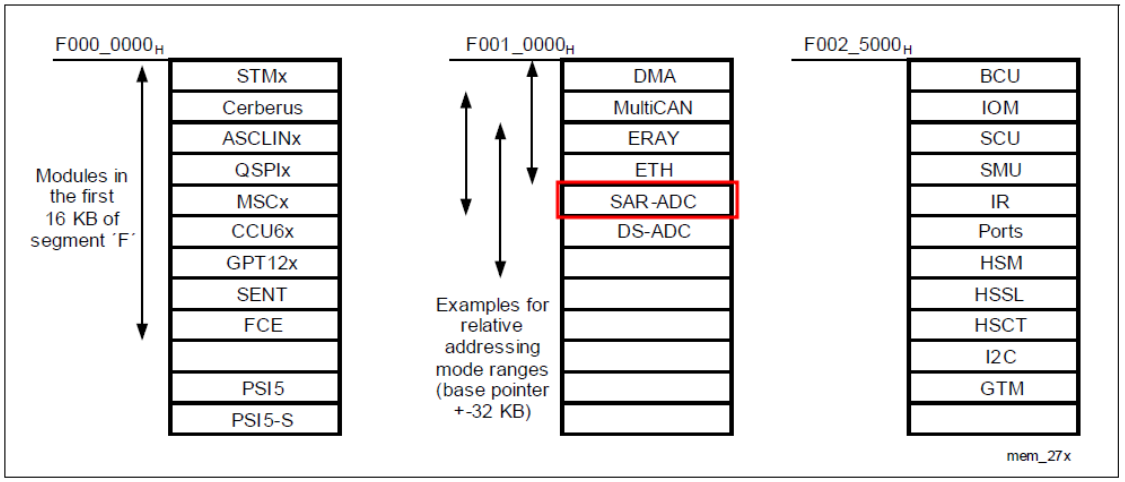


Figure 3-1 Segment F Structure

TC275 MCU 에서 **SAR-ADC 는 VADC 와 동일**

noAltref)			
G4CH4	I	AN36, P40.6 (X)	analog input channel 4 of group 4
G4CH5	I	AN37, P40.7 (X)	analog input channel 5 of group 4
G4CH6	I	AN38, P40.8 (X)	analog input channel 6 of group 4
G4CH7	I	AN39, P40.9 (X)	analog input channel 7 of group 4
G5CH0 (AltRef)	I	AN40	analog input channel 0 of group 5
G5CH1 (MD)	I	AN41	analog input channel 1 of group 5

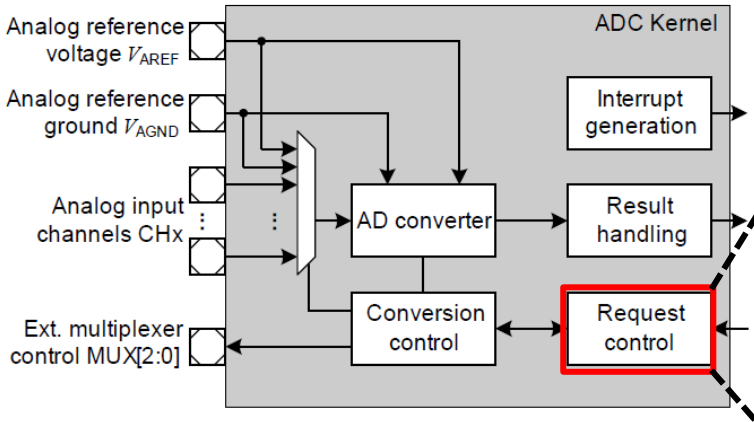
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4028

해당 핀은 아날로그 입력으로만 사용
Channel 7, Group 4 에 mapping 되어 있음
→ **AN39**

VADC 변환 flow 및 원리

: ADC 변환 요청 신호 생성 – Arbiter가 중재하여 Converter로 전달

- ADC 변환을 요청하는 Conversion Request 생성



여러 가지 요소로부터 Request 를 생성 가능
(Timer, External, Software)

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3880

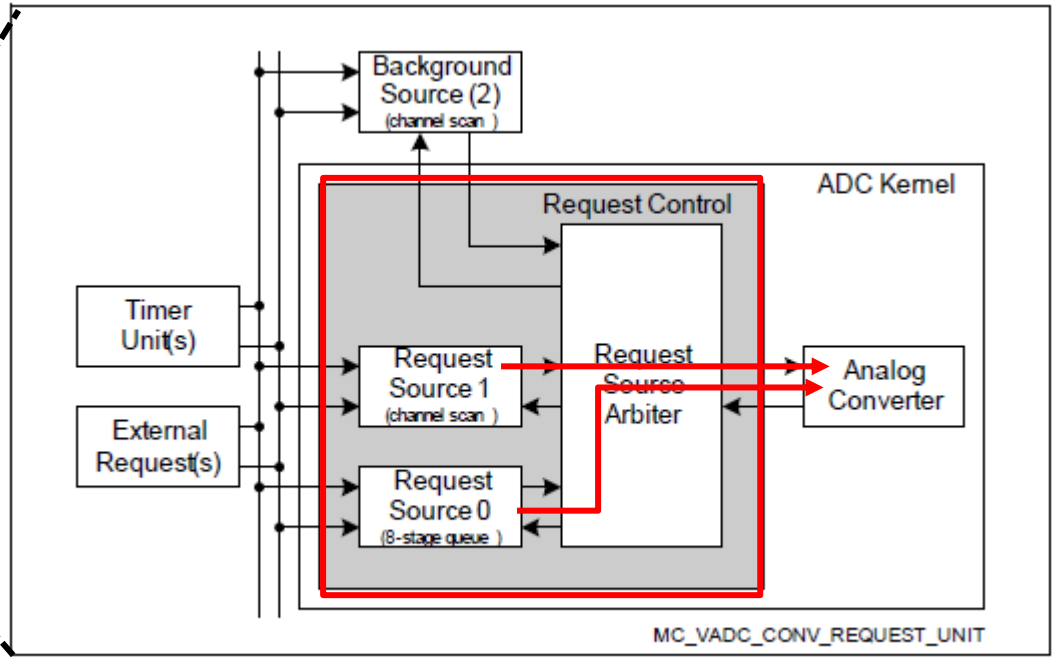


Figure 28-3 Conversion Request Unit

2개의 Request Source 에서 생성된 Conversion Request 들이 Analog Converter로 전달됨
=> 우리는 Request Source 0을 사용하기로 함

VADC Trigger 생성을 위한 Arbiter 설정 : 설정 레지스터

- VADC 레지스터 항목에서 (group = 4)
  - **G4ARBPR** 레지스터 설정
  - **G4QMR"0"** 레지스터 설정 (Request Source = 0 이므로)
  - **G4ARBCFG** 레지스터 설정
  - **G4ICLASS"0"** 레지스터 설정 (Input Class = 0 이므로)

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3880

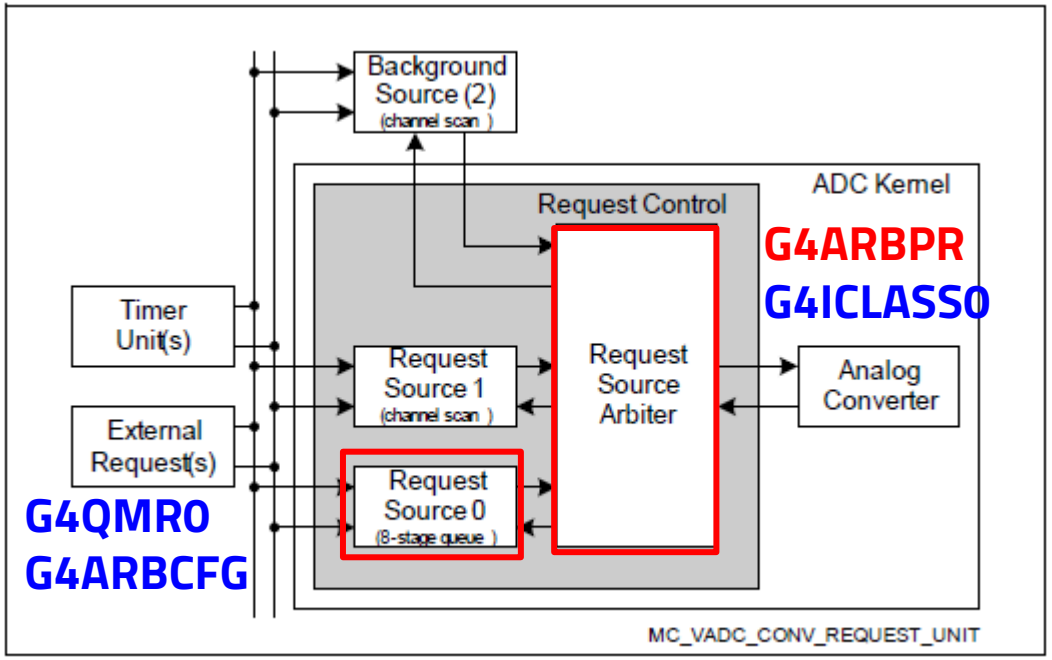


Figure 28-3 Conversion Request Unit

VADC 레지스터 설정 – G4ARBPR

: Arbitration Priority

1. VADC 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0020000**

2. 사용할 레지스터의 주소 찾기 **– SAR4_7핀이 Group 4에 할당되었음을 상기**
   - **G4ARBPR** 의 Offset Address (group = x = 4) = (x * 0x400) + 0x484 = **0x1484**

   → G4ARBPR 레지스터 주소 = 0xF0020000 + 0x1484 = **0xF0021484**

				P	
GxARBPR	Arbitration Priority Register	$X484_H$	U, SV	U, SV P	28-71
GxCHASS	Channel Assignment Register, Group x	$X488_H$	U, SV	U, SV P	28-25

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4023

GxARBPR (x = 0 - 7)
Arbitration Priority Register **Group x** **(x * 0400_H + 0484_H)** **Reset Value: $0000\ 0000_H$**

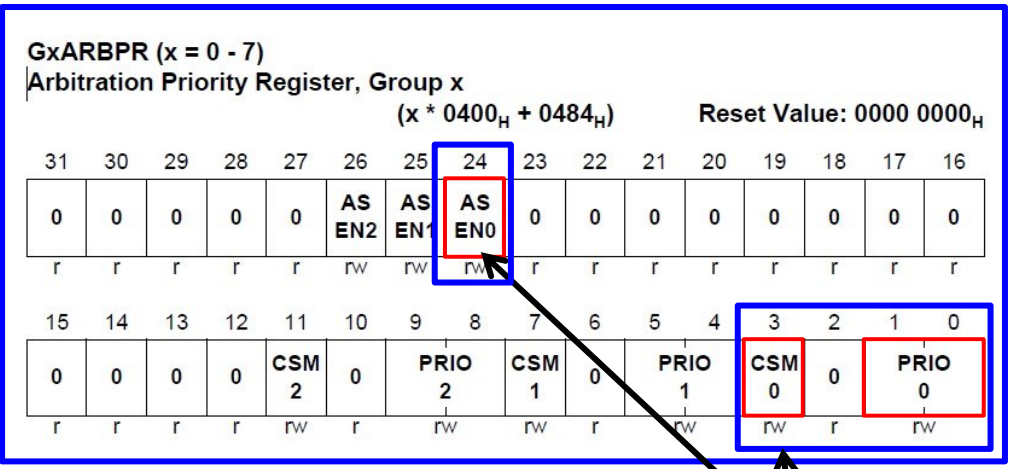
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3945

VADC 레지스터 설정 – G4ARBPR

: Trigger를 위한 Priority 설정

3. 레지스터 write 값 결정
   - Request Source 0 의 우선순위를 가장 높게 하기 위해 **PRIO0 영역에 0x3 write**
   - Conversion Request 가 현재 수행하는 conversion이 끝나고 실행되도록 **CSM0 영역에 0x0 write**
   - Request Source 0 을 enable 하기 위해 **ASEN0 영역에 0x1 write**

G4ARBPR 레지스터 @ 0xF0021484



Request Source 0 을 위한 설정

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3945

Field	Bits	Type	Description
PRIO0, PRIO1, PRIO2	[1:0], [5:4], [9:8]	rw	**Priority of Request Source x** Arbitration priority of request source x (in slot x) 00_B Lowest priority is selected. ... 11_B Highest priority is selected.
CSM0, CSM1, CSM2	3, 7, 11	rw	**Conversion Start Mode of Request Source x** 0_B Wait-for-start mode 1_B Cancel-inject-repeat mode, i.e. this source can cancel conversion of other sources.
0	2, 6, 10, [23:12]	r	**Reserved, write 0, read as 0**
ASENy (y = 0 - 2)	24 + y	rw	**Arbitration Slot y Enable** Enables the associated arbitration slot of an arbiter round. The request source bits are not modified by write actions to ASENR. 0_B The corresponding arbitration slot is disabled and considered as empty. Pending conversion requests from the associated request source are disregarded. 1_B The corresponding arbitration slot is enabled. Pending conversion requests from the associated request source are arbitrated.

Lab9: VADC 레지스터 설정 – G4ARBPR

: VADC 모듈에 입력되는 변환 요청 신호에 대한 설정

```
280
281 void initVADC(void)
282 {
283     // Password Access to unlock SCU_WDTSCON0
284     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
285     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
286
287     // Modify Access to clear ENDINIT
288     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINI
289     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
290
291     VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable VADC
292
293     // Password Access to unlock SCU_WDTSCON0
294     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINI
295     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
296
297     // Modify Access to set ENDINIT
298     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_
299     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
300
301
302     // VADC configurations
303     while((VADC_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until VADC module enabled
304
305     VADC_G4_ARBPR.U |= 0x3 << PRIO0_BIT_LSB_IDX;          // highest priority for Request Source 0
306     VADC_G4_ARBPR.U &= ~(0x1 << CSM0_BIT_LSB_IDX);        // Wait-for-Start Mode
307     VADC_G4_ARBPR.U |= 0x1 << ASEN0_BIT_LSB_IDX;          // Arbitration Source Input 0 Enable
308
309     VADC_G4_QMR0.U &= ~(0x3 << ENGT_BIT_LSB_IDX);
310     VADC_G4_QMR0.U |= 0x1 << ENGT_BIT_LSB_IDX;          // enable conversion request
311     VADC_G4_QMR0.U |= 0x1 << FLUSH_BIT_LSB_IDX;         // clear ADC queue
312
313     VADC_G4_ARBCFG.U |= 0x3 << ANONC_BIT_LSB_IDX;       // ADC normal operation
314
315     VADC_G4_ICLASS0.U &= ~(0x7 << CMS_BIT_LSB_IDX);     // Class 0 Standard Conversion (12-bit)
316
317
318     // VADC Group 4 Channel 7 configuration
319     VADC_G4_CHCTR7.U |= 0x1 << RESPOS_BIT_LSB_IDX;      // result right-aligned
320     VADC_G4_CHCTR7.U &= ~(0xF << RESREG_BIT_LSB_IDX);   // store result @ Result Register G4RES0
321     VADC_G4_CHCTR7.U &= ~(0x3 << ICLSEL_BIT_LSB_IDX);   // Class 0
322
323     VADC_G4_CHASS.U |= 0x1 << ASSCH7_BIT_LSB_IDX;
324 }
325
```

```
71 // VADC registers
72 #define DISS_BIT_LSB_IDX          1
73 #define DISR_BIT_LSB_IDX          0
74 #define ANONC_BIT_LSB_IDX         0
75 #define ASEN0_BIT_LSB_IDX         24
76 #define CSM0_BIT_LSB_IDX          3
77 #define PRIO0_BIT_LSB_IDX         0
78 #define CMS_BIT_LSB_IDX           8
79 #define FLUSH_BIT_LSB_IDX         10
80 #define TREV_BIT_LSB_IDX          9
81 #define ENGT_BIT_LSB_IDX          0
82 #define RESPOS_BIT_LSB_IDX        21
83 #define RESREG_BIT_LSB_IDX        16
84 #define ICLSEL_BIT_LSB_IDX        0
85 #define VF_BIT_LSB_IDX            31
86 #define RESULT_BIT_LSB_IDX        0
87 #define ASSCH7_BIT_LSB_IDX        7
```

VADC Trigger 생성을 위한 Arbiter 설정 : 설정 레지스터

- VADC 레지스터 항목에서 (group = 4)
  - **G4ARBPR** 레지스터 설정
  - **G4QMR"0"** 레지스터 설정 (Request Source = 0 이므로)
  - **G4ARBCFG** 레지스터 설정
  - **G4ICLASS"0"** 레지스터 설정 (Input Class = 0 이므로)

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3880

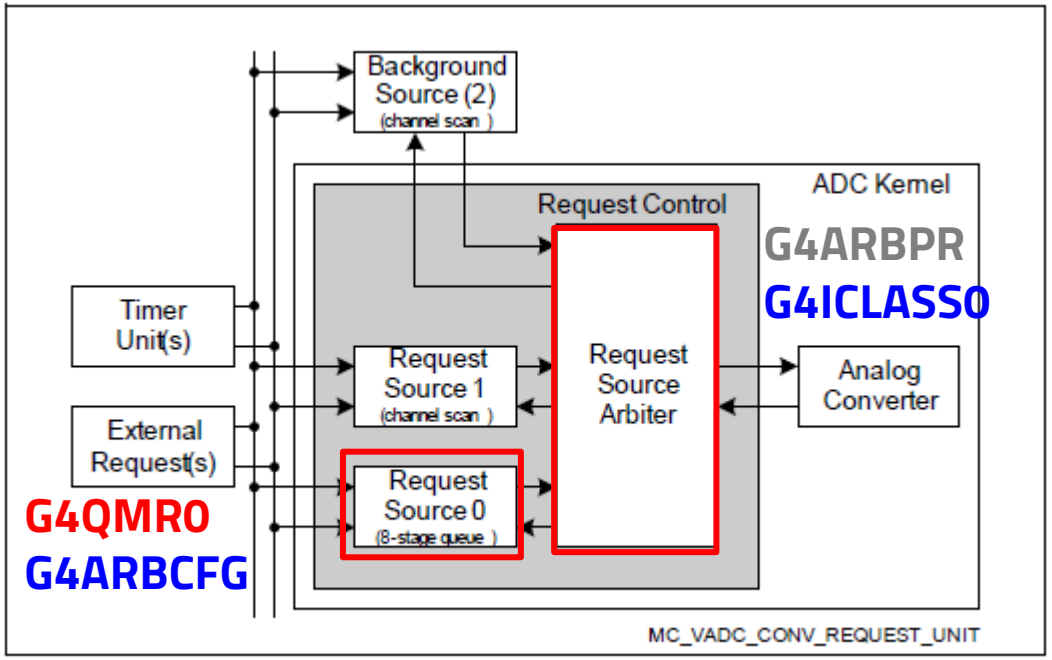


Figure 28-3 Conversion Request Unit

VADC 레지스터 설정 – G4QMR0

: Queue Mode Register

1. VADC 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0020000**

2. 사용할 레지스터의 주소 찾기
   - **G4QMR0** 의 Offset Address (group = x = 4) = (x * 0x400) + 0x504 = **0x1504**

   → G4QMR0 레지스터 주소 = 0xF0020000 + 0x1504 = **0xF0021504**

GxQCTRL0	Queue 0 Source Control Register, Group x	$X500_H$	U, SV	U, SV P	28-38
GxQMR0	Queue 0 Mode Register, Group x	$X504_H$	U, SV	U, SV P	28-40
GxQSR0	Queue 0 Status Register, Group x	$X508_H$	U, SV	U, SV P	28-42

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4023

GxQMR0 (x = 0 - 7)
Queue 0 Mode Register, Group x
($x * 0400_H + 0504_H$)

Reset Value: $0000\ 0000_H$

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3914

VADC 레지스터 설정 – G4QMR0

: Queue Mode Register

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3908

3. 레지스터 write 값 결정
   - Request Source 0 에서 request gating 기능을 사용하지 않도록 하기 위해 **ENGT** 영역에 **0x1 write**

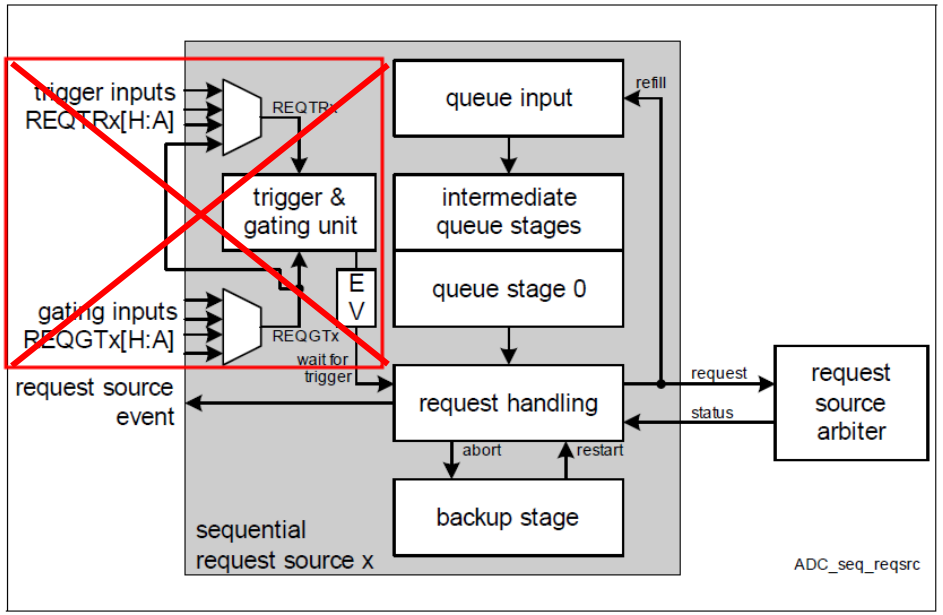


Figure 28-8 Queued Request Source

G4QMR0 레지스터 @ 0xF0021504

GxQMR0 (x = 0 - 7)
Queue 0 Mode Register, Group x
(x * 0400_H + 0504_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RPT DIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	CEV	FLU SH	TR EV	CLR V	0	0	0	0	0	EN TR	ENGT	
r	r	r	r	w	w	w	w	r	r	r	r	r	rw	rw	

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3914

Field	Bits	Type	Description
ENGT	[1:0]	rw	**Enable Gate** Selects the gating functionality for source 0/2. 00_B No conversion requests are issued 01_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register 10_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register and REQGTx = 1 11_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register and REQGTx = 0 *Note: REQGTx is the selected gating signal.*

VADC 레지스터 설정 – G4QMR0

: VADC에 시작을 위한 Trigger 신호 선택 설정

3. 레지스터 write 값 결정
   - Conversion Request 를 Software 에서 발생시키려면 **TREV 영역에 0x1 값을 write 해야 함** → ADC 변환을 시작하는 Trigger 역할이므로 원하는 시점에 set 해야 함 (Polling방식)
   - Request Source 0 에 의한 Conversion Request Queue 를 clear 하기 위해 **FLUSH 영역에 0x1 write을 쓰면 Queue가 Clear됨**

G4QMR0 레지스터 @ 0xF0021504

GxQMR0 (x = 0 - 7)
Queue 0 Mode Register, Group x
(x * 0400_H + 0504_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RPT DIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	CEV	FLU SH	TR EV	CLR V	0	0	0	0	0	EN TR	ENGT	
r	r	r	r	w	w	w	w	r	r	r	r	r	rw	rw	

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3914

TREV	9	w	**Trigger Event** 0_B No action 1_B Generate a trigger event by software
FLUSH	10	w	**Flush Queue** 0_B No action 1_B Clear all queue entries (including backup stage) and the event flag EV. The queue contains no more valid entry.

Lab10: VADC 레지스터 설정 – G4QMR0

: VADC에 입력되는 요청 신호 선택 설정

```
280
281 void initVADC(void)
282 {
283     // Password Access to unlock SCU_WDTSCON0
284     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
285     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
286
287     // Modify Access to clear ENDINIT
288     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINI
289     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
290
291     VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable VADC
292
293     // Password Access to unlock SCU_WDTSCON0
294     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINI
295     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
296
297     // Modify Access to set ENDINIT
298     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_
299     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
300
301
302     // VADC configurations
303     while((VADC_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until VADC module enabled
304
305     VADC_G4_ARBPR.U |= 0x3 << PRIO0_BIT_LSB_IDX;         // highest priority for Request Source 0
306     VADC_G4_ARBPR.U &= ~(0x1 << CSM0_BIT_LSB_IDX);       // Wait-for-Start Mode
307     VADC_G4_ARBPR.U |= 0x1 << ASEN0_BIT_LSB_IDX;         // Arbitration Source Input 0 Enable
308
309     VADC_G4_QMR0.U &= ~(0x3 << ENGT_BIT_LSB_IDX);
310     VADC_G4_QMR0.U |= 0x1 << ENGT_BIT_LSB_IDX;          // enable conversion request
311     VADC_G4_QMR0.U |= 0x1 << FLUSH_BIT_LSB_IDX;         // clear ADC queue
312
313     VADC_G4_ARBCFG.U |= 0x3 << ANONC_BIT_LSB_IDX;       // ADC normal operation
314
315     VADC_G4_ICLASS0.U &= ~(0x7 << CMS_BIT_LSB_IDX);     // Class 0 Standard Conversion (12-bit)
316
317
318     // VADC Group 4 Channel 7 configuration
319     VADC_G4_CHCTR7.U |= 0x1 << RESPOS_BIT_LSB_IDX;      // result right-aligned
320     VADC_G4_CHCTR7.U &= ~(0xF << RESREG_BIT_LSB_IDX);   // store result @ Result Register G4RES0
321     VADC_G4_CHCTR7.U &= ~(0x3 << ICLSEL_BIT_LSB_IDX);   // Class 0
322
323     VADC_G4_CHASS.U |= 0x1 << ASSCH7_BIT_LSB_IDX;
324 }
325
```

```
71 // VADC registers
72 #define DISS_BIT_LSB_IDX          1
73 #define DISR_BIT_LSB_IDX          0
74 #define ANONC_BIT_LSB_IDX         0
75 #define ASEN0_BIT_LSB_IDX         24
76 #define CSM0_BIT_LSB_IDX          3
77 #define PRIO0_BIT_LSB_IDX         0
78 #define CMS_BIT_LSB_IDX           8
79 #define FLUSH_BIT_LSB_IDX         10
80 #define TREV_BIT_LSB_IDX          9
81 #define ENGT_BIT_LSB_IDX          0
82 #define RESPOS_BIT_LSB_IDX        21
83 #define RESREG_BIT_LSB_IDX        16
84 #define ICLSEL_BIT_LSB_IDX        0
85 #define VF_BIT_LSB_IDX            31
86 #define RESULT_BIT_LSB_IDX        0
87 #define ASSCH7_BIT_LSB_IDX        7
```

VADC Trigger 생성을 위한 Arbiter 설정
: 설정 레지스터

- VADC 레지스터 항목에서 (group = 4)
  - **G4ARBPR** 레지스터 설정
  - **G4QMR"0"** 레지스터 설정 (Request Source = 0 이므로)
  - **G4ARBCFG** 레지스터 설정
  - **G4ICLASS"0"** 레지스터 설정 (Input Class = 0 이므로)

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3880

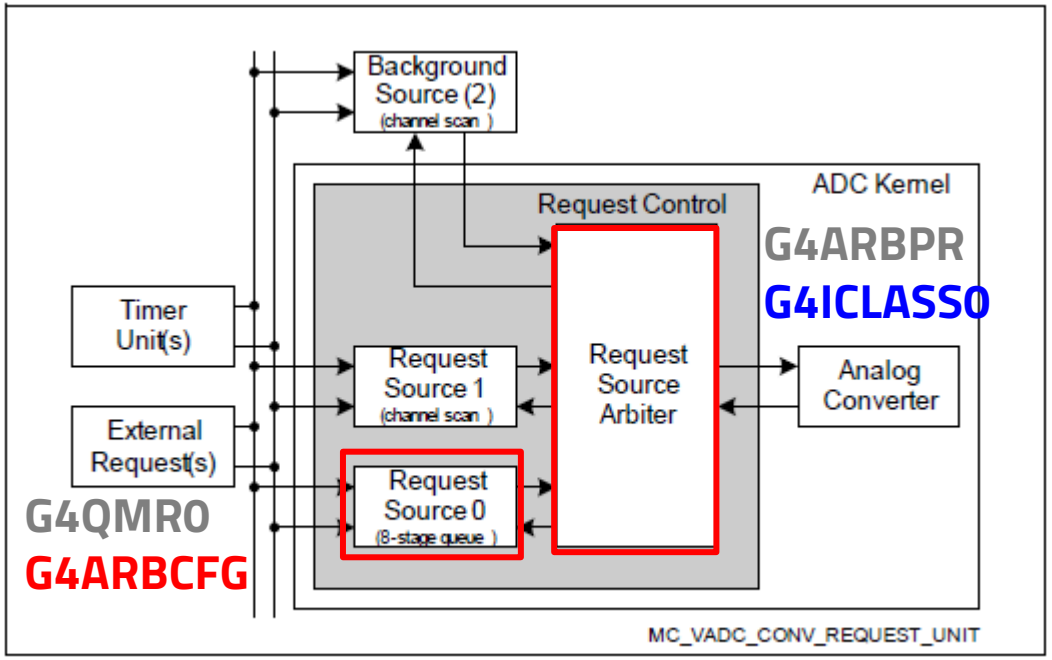


Figure 28-3 Conversion Request Unit

VADC 레지스터 설정 – G4ARBCFG

: Arbitration Configuration

1. VADC 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0020000**

2. 사용할 레지스터의 주소 찾기
   - **G4ARBCFG** 의 Offset Address (group = x = 4) = (x * 0x400) + 0x480 = **0x1480**

   → G4ARBCFG 레지스터 주소 = 0xF0020000 + 0x1480 = **0xF0021480**

ACCPROT1	Access Protection Register 1	$008C_H$	U, SV	SV, SE, P	28-28
GxARBCFG	Arbitration Configuration Register	$X480_H$	U, SV	U, SV P	28-68
GxARBPR	Arbitration Priority Register	$X484_H$	U, SV	U, SV P	28-71

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4023

GxARBCFG (x = 0 - 7)
Arbitration Configuration Register, Group x
($x * 0400_H + 0480_H$)

Reset Value: $0000\ 0000_H$

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3942

VADC 레지스터 설정 – G4ARBCFG
: VADC 동작 모드 설정

3. 레지스터 write 값 결정
   - ADC 를 Normal Operation Mode 로 동작 시키기 위해 **ANONC 영역에 0x3 write**

G4ARBCFG 레지스터 @ 0xF0021480

GxARBCFG (x = 0 - 7)
Arbitration Configuration Register, Group x
($x * 0400_H + 0480_H$) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAMPLE	BUSY	CALS	CAL	0	0	SYNRUN	CHNR					CSRC		ANONS	
rh	rh	rh	rh	r	r	rh	rh					rh		rh	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	ARBM	0	ARBRND		0	0	ANONC	
r	r	r	r	r	r	r	r	rw	r	rw		r	r	rw	

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3942

Field	Bits	Type	Description
ANONC	[1:0]	rw	**Analog Converter Control** Defines the value of bitfield ANONS in a stand-alone converter or a converter in master mode. Coding see ANONS or **Section 28.4.1**.
0	[3:2]	r	**Reserved, write 0, read as 0**

ANONS 값 사용

28.4.1 Analog Converter Control

The operating mode is determined by bitfield **GxARBCFG (x = 0 - 7)**.ANONS:

- ANONS = 11_B: **Normal Operation**
  The converter is active, conversions are started immediately.
  Requires no wakeup time.
- ANONS = 10 or 01 : **Reserved**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3904

Lab11 - VADC 레지스터 설정 – G4ARBCFG

: VADC 동작 모드 설정

```
280
281 void initVADC(void)
282 {
283     // Password Access to unlock SCU_WDTSCON0
284     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
285     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
286
287     // Modify Access to clear ENDINIT
288     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINI
289     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
290
291     VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable VADC
292
293     // Password Access to unlock SCU_WDTSCON0
294     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINI
295     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
296
297     // Modify Access to set ENDINIT
298     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_
299     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
300
301
302     // VADC configurations
303     while((VADC_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until VADC module enabled
304
305     VADC_G4_ARBPR.U |= 0x3 << PRIO0_BIT_LSB_IDX;         // highest priority for Request Source 0
306     VADC_G4_ARBPR.U &= ~(0x1 << CSM0_BIT_LSB_IDX);       // Wait-for-Start Mode
307     VADC_G4_ARBPR.U |= 0x1 << ASEN0_BIT_LSB_IDX;         // Arbitration Source Input 0 Enable
308
309     VADC_G4_QMR0.U &= ~(0x3 << ENGT_BIT_LSB_IDX);
310     VADC_G4_QMR0.U |= 0x1 << ENGT_BIT_LSB_IDX;          // enable conversion request
311     VADC_G4_QMR0.U |= 0x1 << FLUSH_BIT_LSB_IDX;         // clear ADC queue
312
313     VADC_G4_ARBCFG.U |= 0x3 << ANONC_BIT_LSB_IDX;       // ADC normal operation
314
315     VADC_G4_ICLASS0.U &= ~(0x7 << CMS_BIT_LSB_IDX);     // Class 0 Standard Conversion (12-bit)
316
317
318     // VADC Group 4 Channel 7 configuration
319     VADC_G4_CHCTR7.U |= 0x1 << RESPOS_BIT_LSB_IDX;      // result right-aligned
320     VADC_G4_CHCTR7.U &= ~(0xF << RESREG_BIT_LSB_IDX);   // store result @ Result Register G4RES0
321     VADC_G4_CHCTR7.U &= ~(0x3 << ICLSEL_BIT_LSB_IDX);   // Class 0
322
323     VADC_G4_CHASS.U |= 0x1 << ASSCH7_BIT_LSB_IDX;
324 }
325
```

```
71 // VADC registers
72 #define DISS_BIT_LSB_IDX          1
73 #define DISR_BIT_LSB_IDX          0
74 #define ANONC_BIT_LSB_IDX         0
75 #define ASEN0_BIT_LSB_IDX         24
76 #define CSM0_BIT_LSB_IDX          3
77 #define PRIO0_BIT_LSB_IDX         0
78 #define CMS_BIT_LSB_IDX           8
79 #define FLUSH_BIT_LSB_IDX         10
80 #define TREV_BIT_LSB_IDX          9
81 #define ENGT_BIT_LSB_IDX          0
82 #define RESPOS_BIT_LSB_IDX        21
83 #define RESREG_BIT_LSB_IDX        16
84 #define ICLSEL_BIT_LSB_IDX        0
85 #define VF_BIT_LSB_IDX            31
86 #define RESULT_BIT_LSB_IDX        0
87 #define ASSCH7_BIT_LSB_IDX        7
```

VADC Trigger 생성을 위한 Arbiter 설정
: 설정 레지스터

- VADC 레지스터 항목에서 (group = 4)
  - **G4ARBPR** 레지스터 설정
  - **G4QMR"0"** 레지스터 설정 (Request Source = 0 이므로)
  - **G4ARBCFG** 레지스터 설정
  - **G4ICLASS"0"** 레지스터 설정 (Input Class = 0 이므로)

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3880

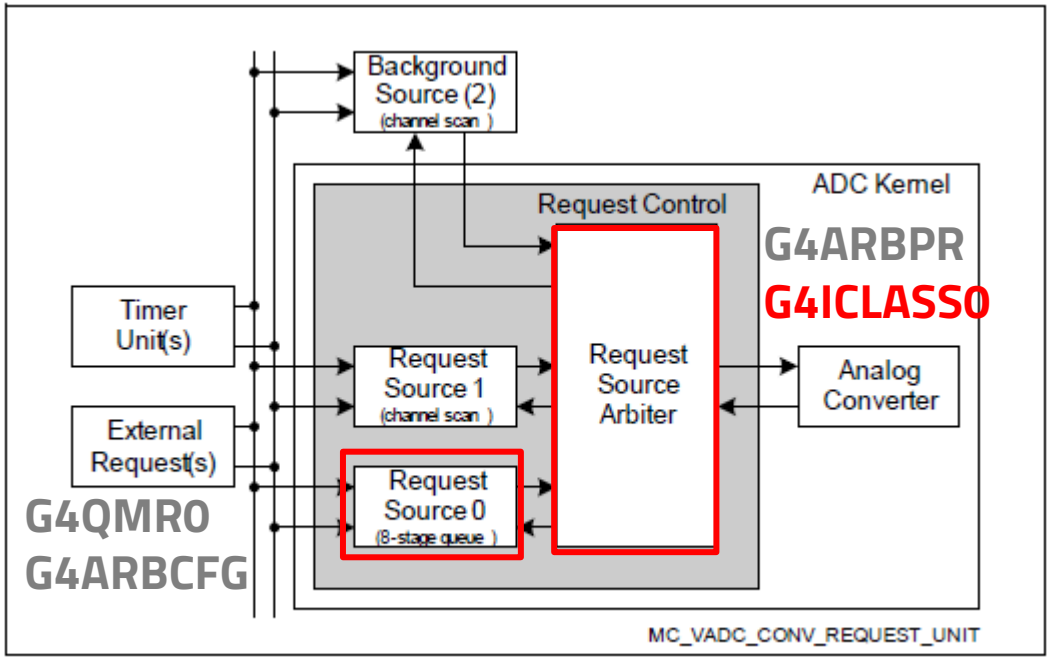


Figure 28-3 Conversion Request Unit

VADC 레지스터 설정 – G4ICLASS0

: Class0으로 설정된 입력에 대한 변환모드 결정

1. VADC 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0020000**

2. 사용할 레지스터의 주소 찾기 **(후술할 SAR4_7핀을 Class0에 할당함)**
   - **G4ICLASS0** 의 Offset Address (group = x = 4) = (x * 0x400) + 0x4A0 = **0x14A0**

   → G4ICLASS0 레지스터 주소 = 0xF0020000 + 0x14A0 = **0xF00214A0**

				P	28-75
GxICLASS0	Input Class Register 0, Group x	$X4A0_H$	U, SV	U, SV P	28-78
GxICLASS1	Input Class Register 1, Group x	$X4A4_H$	U, SV	U, SV P	28-78

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4024

GxICLASS0 (x = 0 - 7)
Input Class Register 0, Group x
($x * 0400_H + 04A0_H$)

Reset Value: $0000\ 0000_H$

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3952

VADC 레지스터 설정 – G4ICLASS0

: VADC 변환 결과가 어떤 형식으로 저장될 지 설정

3. 레지스터 write 값 결정
   - **후술할 SAR4_7핀을 Group 4 의 Input Class 를 0으로 설정함 (G4CHCTR7 레지스터에)**
   - 따라서 G4ICLASS"0" 레지스터에 설정해야 함.
   - 12bit 디지털 값으로 ADC 변환하는 Conversion Mode 사용 위해 **CMS 영역에 0x0 write**

G4ICLASS0 레지스터 @ 0xF0021480

GxICLASS0 (x = 0 - 7)
Input Class Register 0, Group x

$(x * 0400_H + 04A0_H)$ **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	CME			0	0	0	STCE				
r	r	r	r	r	rw			r	r	r	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	CMS			0	0	0	STCS				
r	r	r	r	r	rw			r	r	r	rw				

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3952

CMS	[10:8]	rw	**Conversion Mode for Standard Conversions** 000_B 12-bit conversion 001_B 10-bit conversion 010_B 8-bit conversion 011_B Reserved 100_B Reserved 101_B 10-bit fast compare mode 110_B Reserved 111_B Reserved

Lab12 - VADC 레지스터 설정 – G4ICLASS0

: VADC 변환 결과가 어떤 형식으로 저장될 지 설정

```
280
281 void initVADC(void)
282 {
283     // Password Access to unlock SCU_WDTSCON0
284     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
285     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
286
287     // Modify Access to clear ENDINIT
288     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINI
289     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
290
291     VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable VADC
292
293     // Password Access to unlock SCU_WDTSCON0
294     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINI
295     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
296
297     // Modify Access to set ENDINIT
298     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_
299     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
300
301
302     // VADC configurations
303     while((VADC_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until VADC module enabled
304
305     VADC_G4_ARBPR.U |= 0x3 << PRIO0_BIT_LSB_IDX;         // highest priority for Request Source 0
306     VADC_G4_ARBPR.U &= ~(0x1 << CSM0_BIT_LSB_IDX);       // Wait-for-Start Mode
307     VADC_G4_ARBPR.U |= 0x1 << ASEN0_BIT_LSB_IDX;         // Arbitration Source Input 0 Enable
308
309     VADC_G4_QMR0.U &= ~(0x3 << ENGT_BIT_LSB_IDX);
310     VADC_G4_QMR0.U |= 0x1 << ENGT_BIT_LSB_IDX;          // enable conversion request
311     VADC_G4_QMR0.U |= 0x1 << FLUSH_BIT_LSB_IDX;         // clear ADC queue
312
313     VADC_G4_ARBCFG.U |= 0x3 << ANONC_BIT_LSB_IDX;       // ADC normal operation
314
315     VADC_G4_ICLASS0.U &= ~(0x7 << CMS_BIT_LSB_IDX);     // Class 0 Standard Conversion (12-bit)
316
317
318     // VADC Group 4 Channel 7 configuration
319     VADC_G4_CHCTR7.U |= 0x1 << RESPOS_BIT_LSB_IDX;      // result right-aligned
320     VADC_G4_CHCTR7.U &= ~(0xF << RESREG_BIT_LSB_IDX);   // store result @ Result Register G4RES0
321     VADC_G4_CHCTR7.U &= ~(0x3 << ICLSEL_BIT_LSB_IDX);   // Class 0
322
323     VADC_G4_CHASS.U |= 0x1 << ASSCH7_BIT_LSB_IDX;
324 }
325
```

```
71 // VADC registers
72 #define DISS_BIT_LSB_IDX          1
73 #define DISR_BIT_LSB_IDX          0
74 #define ANONC_BIT_LSB_IDX         0
75 #define ASEN0_BIT_LSB_IDX         24
76 #define CSM0_BIT_LSB_IDX          3
77 #define PRIO0_BIT_LSB_IDX         0
78 #define CMS_BIT_LSB_IDX           8
79 #define FLUSH_BIT_LSB_IDX         10
80 #define TREV_BIT_LSB_IDX          9
81 #define ENGT_BIT_LSB_IDX          0
82 #define RESPOS_BIT_LSB_IDX        21
83 #define RESREG_BIT_LSB_IDX        16
84 #define ICLSEL_BIT_LSB_IDX        0
85 #define VF_BIT_LSB_IDX            31
86 #define RESULT_BIT_LSB_IDX        0
87 #define ASSCH7_BIT_LSB_IDX        7
```

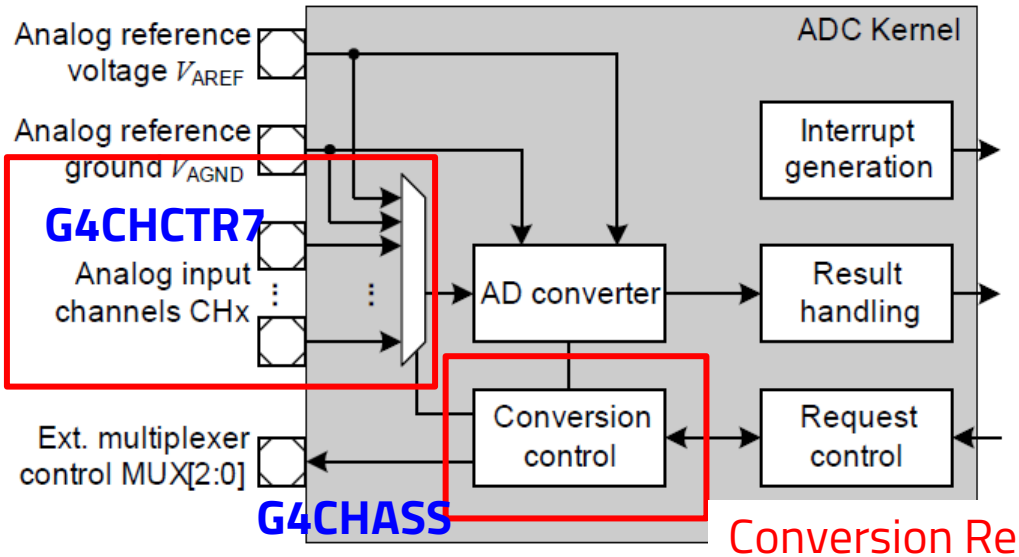
VADC 입력 선택 및 변환 제어 방식 설정
: 필요한 레지스터들

- 가변 저항이 연결된 핀 **AN39** 는 **Group 4, Input Channel 7**을 상기하자~
- Input Channel 에 대한 Input Class 지정, 변환 완료 후 저장될 레지스터 지정 등 설정 필요
- Conversion Request 발생하는 policy 설정 필요

- VADC 레지스터 항목에서
  - **G4CHCTR7** 레지스터 설정
  - **G4CHASS** 레지스터 설정
  - **G4QINR0** 레지스터 설정

  (4 = group, 7 = input channel, 0 = Request Source)

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3878



Analog Input Channel 관련 설정

Conversion Request 관련 설정

VADC 변환 flow 및 원리
: ADC 변환 결과 저장 위치 설정

- ADC 변환 결과가 저장될 위치를 결정하는 Result Handling

ADC 변환 결과는 Global Result Register 에 저장

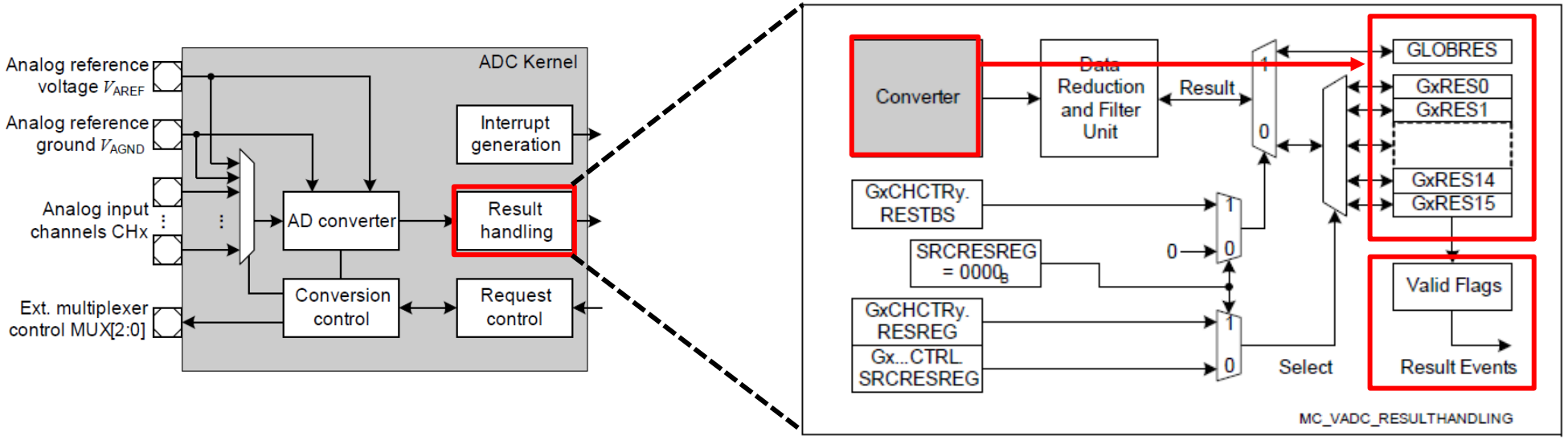


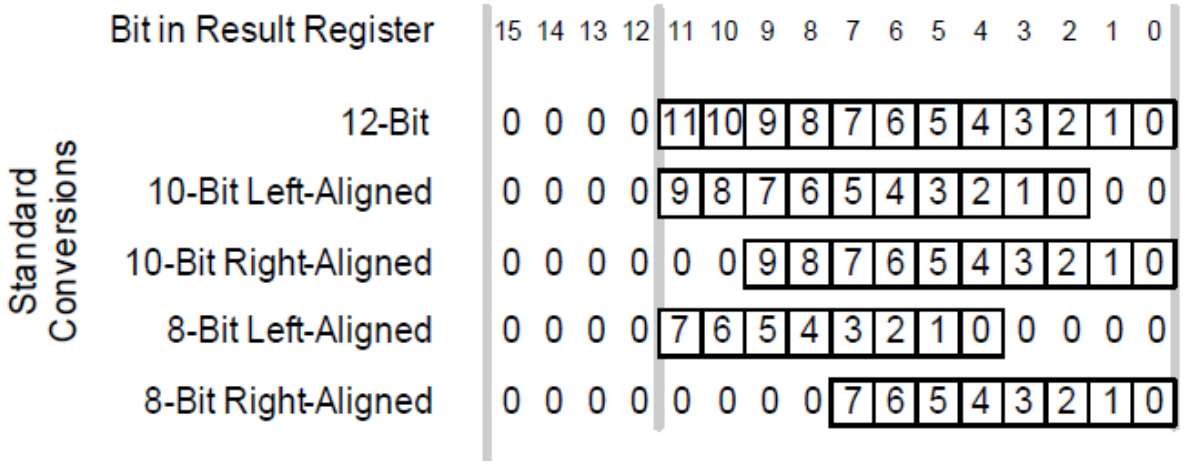
Figure 28-18 Conversion Result Storage

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3969

- 레지스터에 새로운 값이 저장되면 Valid Flag 가 set 됨
- 레지스터를 Software로 읽으면 자동으로 clear

VADC 변환 flow 및 원리
: ADC 변환 결과의 디지털 값 형식

- ADC 변환 결과의 format
  - ADC 변환된 결과인 디지털 값은 8bit 부터 12bit 까지의 크기를 가질 수 있음
  - 설정에 따라 Result Register에 Right-Aligned (오른쪽 정렬) 또는 Left-Aligned (왼쪽 정렬)로 저장됨
  - 필요로 하는 목적에 따라 ADC 변환 결과가 저장될 형식을 적절히 설정해야 함



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3980

VADC 레지스터 설정 – G4CHCTR7

Channel Control

1. VADC 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0020000**

2. **SAR4_7핀이 Group 4에, Channel 7에 할당되어 있음을 상기.**
   - **G4CHCTR7** 의 Offset Address (channel = y = 7) = 0x1600 + (y * 0x4) = **0x161C**

   → G4CHCTR7 레지스터 주소 = 0xF0020000 + 0x161C = **0xF002161C**

BRSPNDx	Background Request Source Channel Pending Register, Group x	$01CY_H$	U, SV	U, SV P	28-65
GxCHCTRy	Channel x Control Register	$X60Y_H$	U, SV	U, SV P	28-75
GxICLASS0	Input Class Register 0, Group x	$X4A0_H$	U, SV	U, SV P	28-78

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4024

G4CHCTRy (y = 0 - 7)
Group 4, Channel y Ctrl. Reg. $(1600_H + y * 0004_H)$ **Reset Value: 0000 0000$_H$**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3949

VADC 레지스터 설정 – G4CHCTR7

: 채널 class, 저장 위치, 정렬방식 설정

3. 레지스터 write 값 결정 –**G4CHCTR7은 Group4, Channel7에 대한 설정임.**
   - **SAR4_7핀을 Input Class 0**으로 설정하기 위해 **ICLSEL 영역에 0x0 write**
   - ADC 변환 결과를 Group 4 의 Result Register 0 에 저장하기 위해 **RESREG 영역에 0x0 write**
   - **ADC 변환 결과를 Right-Aligned (오른쪽 정렬)로 저장하기 위해 RESPOS 영역에 0x1 write**

G4CHCTR7 레지스터 @ 0xF002161C

G4CHCTRy (y = 0 - 7)
Group 4, Channel y Ctrl. Reg. (1600_H + y * 0004_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	BWDEN	BWDCH		0	0	0	0	0	0	RESPOS	RESTBS	RESREG			
r	rw	rw		r	r	r	r	r	r	rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDSELX				REFSEL	SYNC	CHEVMODE		BNDSELU		BNDSELL		0	0	ICLSEL	
rw				rw	rw	rw		rw		rw		r	r	rw	

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3949

Field	Bits	Type	Description
ICLSEL	[1:0]	rw	**Input Class Select** 00_B Use group-specific class 0 01_B Use group-specific class 1 10_B Use global class 0 11_B Use global class 1
RESREG	[19:16]	rw	**Result Register** 0000_BStore result in group result register GxRES0 ... 1111_BStore result in group result register GxRES15
RESPOS	21	rw	**Result Position** 0_B Store results left-aligned 1_B Store results right-aligned

Lab13 - VADC 레지스터 설정 – G4CHCTR7

: 변환결과 저장위치, Align방법, Class 종류 설정

```
280
281 void initVADC(void)
282 {
283     // Password Access to unlock SCU_WDTSCON0
284     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
285     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
286
287     // Modify Access to clear ENDINIT
288     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINI
289     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
290
291     VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable VADC
292
293     // Password Access to unlock SCU_WDTSCON0
294     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINI
295     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
296
297     // Modify Access to set ENDINIT
298     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_
299     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
300
301
302     // VADC configurations
303     while((VADC_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until VADC module enabled
304
305     VADC_G4_ARBPR.U |= 0x3 << PRIO0_BIT_LSB_IDX;         // highest priority for Request Source 0
306     VADC_G4_ARBPR.U &= ~(0x1 << CSM0_BIT_LSB_IDX);       // Wait-for-Start Mode
307     VADC_G4_ARBPR.U |= 0x1 << ASEN0_BIT_LSB_IDX;         // Arbitration Source Input 0 Enable
308
309     VADC_G4_QMR0.U &= ~(0x3 << ENGT_BIT_LSB_IDX);
310     VADC_G4_QMR0.U |= 0x1 << ENGT_BIT_LSB_IDX;          // enable conversion request
311     VADC_G4_QMR0.U |= 0x1 << FLUSH_BIT_LSB_IDX;         // clear ADC queue
312
313     VADC_G4_ARBCFG.U |= 0x3 << ANONC_BIT_LSB_IDX;       // ADC normal operation
314
315     VADC_G4_ICLASS0.U &= ~(0x7 << CMS_BIT_LSB_IDX);     // Class 0 Standard Conversion (12-bit)
316
317
318     // VADC Group 4 Channel 7 configuration
319     VADC_G4_CHCTR7.U |= 0x1 << RESPOS_BIT_LSB_IDX;      // result right-aligned
320     VADC_G4_CHCTR7.U &= ~(0xF << RESREG_BIT_LSB_IDX);   // store result @ Result Register G4RES0
321     VADC_G4_CHCTR7.U &= ~(0x3 << ICLSEL_BIT_LSB_IDX);   // Class 0
322
323     VADC_G4_CHASS.U |= 0x1 << ASSCH7_BIT_LSB_IDX;
324 }
325
```

```
71 // VADC registers
72 #define DISS_BIT_LSB_IDX          1
73 #define DISR_BIT_LSB_IDX          0
74 #define ANONC_BIT_LSB_IDX         0
75 #define ASEN0_BIT_LSB_IDX         24
76 #define CSM0_BIT_LSB_IDX          3
77 #define PRIO0_BIT_LSB_IDX         0
78 #define CMS_BIT_LSB_IDX           8
79 #define FLUSH_BIT_LSB_IDX         10
80 #define TREV_BIT_LSB_IDX          9
81 #define ENGT_BIT_LSB_IDX          0
82 #define RESPOS_BIT_LSB_IDX        21
83 #define RESREG_BIT_LSB_IDX        16
84 #define ICLSEL_BIT_LSB_IDX        0
85 #define VF_BIT_LSB_IDX            31
86 #define RESULT_BIT_LSB_IDX        0
87 #define ASSCH7_BIT_LSB_IDX        7
```

VADC 레지스터 설정 – G4CHASS

: channel assignment

1. VADC 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0020000**

2. 사용할 레지스터의 주소 찾기
   - **G4CHASS** 의 Offset Address (group = x = 4) = (x * 0x400) + 0x488 = **0x1488**

   → G4CHASS 레지스터 주소 = 0xF0020000 + 0x1488 = **0xF0021488**

GxARBPR	Arbitration Priority Register	$X484_H$	U, SV	U, SV P	28-71
GxCHASS	Channel Assignment Register, Group x	$X488_H$	U, SV	U, SV P	28-25
GxRRASS	Result Assignment Register, Group x	$X48C_H$	U, SV	U, SV	28-26

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4023

GxCHASS (x = 0 - 7)
Channel Assignment Register, Group x **(x * 0400_H + 0488_H)** **Reset Value: $0000\ 0000_H$**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3899

VADC 레지스터 설정 – G4CHASS
: VADC에 입력되는 요청 신호의 우선순위 설정

3. 레지스터 write 값 결정
   - Group 4 의 Input Channel 7 에서 발생하는 Conversion Request를 최우선 순위로 설정하기 위해 **ASSCH7 영역에 0x1 write**
   - Input Channel 7 이므로 ASSCH"7"

G4CHASS 레지스터 @ 0xF0021488

GxCHASS (x = 0 - 7)
Channel Assignment Register, Group x
($x * 0400_H + 0488_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ASSCH31	ASSCH30	ASSCH29	ASSCH28	ASSCH27	ASSCH26	ASSCH25	ASSCH24	ASSCH23	ASSCH22	ASSCH21	ASSCH20	ASSCH19	ASSCH18	ASSCH17	ASSCH16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASSCH15	ASSCH14	ASSCH13	ASSCH12	ASSCH11	ASSCH10	ASSCH9	ASSCH8	ASSCH7	ASSCH6	ASSCH5	ASSCH4	ASSCH3	ASSCH2	ASSCH1	ASSCH0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ASSCHy (y = 0 - 31)	y	rw	**Assignment for Channel y** 0_B Channel y can be a background channel converted with lowest priority 1_B Channel y is a priority channel within group x

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3899

Lab14 - VADC 레지스터 설정 – G4CHASS

: VADC에 입력되는 요청 신호의 우선순위 설정

```
280
281 void initVADC(void)
282 {
283     // Password Access to unlock SCU_WDTSCON0
284     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
285     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
286
287     // Modify Access to clear ENDINIT
288     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINI
289     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
290
291     VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable VADC
292
293     // Password Access to unlock SCU_WDTSCON0
294     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINI
295     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
296
297     // Modify Access to set ENDINIT
298     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_
299     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
300
301
302     // VADC configurations
303     while((VADC_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until VADC module enabled
304
305     VADC_G4_ARBPR.U |= 0x3 << PRIO0_BIT_LSB_IDX;         // highest priority for Request Source 0
306     VADC_G4_ARBPR.U &= ~(0x1 << CSM0_BIT_LSB_IDX);       // Wait-for-Start Mode
307     VADC_G4_ARBPR.U |= 0x1 << ASEN0_BIT_LSB_IDX;         // Arbitration Source Input 0 Enable
308
309     VADC_G4_QMR0.U &= ~(0x3 << ENGT_BIT_LSB_IDX);
310     VADC_G4_QMR0.U |= 0x1 << ENGT_BIT_LSB_IDX;          // enable conversion request
311     VADC_G4_QMR0.U |= 0x1 << FLUSH_BIT_LSB_IDX;         // clear ADC queue
312
313     VADC_G4_ARBCFG.U |= 0x3 << ANONC_BIT_LSB_IDX;       // ADC normal operation
314
315     VADC_G4_ICLASS0.U &= ~(0x7 << CMS_BIT_LSB_IDX);     // Class 0 Standard Conversion (12-bit)
316
317
318     // VADC Group 4 Channel 7 configuration
319     VADC_G4_CHCTR7.U |= 0x1 << RESPOS_BIT_LSB_IDX;      // result right-aligned
320     VADC_G4_CHCTR7.U &= ~(0xF << RESREG_BIT_LSB_IDX);   // store result @ Result Register G4RES0
321     VADC_G4_CHCTR7.U &= ~(0x3 << ICLSEL_BIT_LSB_IDX);   // Class 0
322
323     VADC_G4_CHASS.U |= 0x1 << ASSCH7_BIT_LSB_IDX;
324 }
325
```

```
71 // VADC registers
72 #define DISS_BIT_LSB_IDX          1
73 #define DISR_BIT_LSB_IDX          0
74 #define ANONC_BIT_LSB_IDX         0
75 #define ASEN0_BIT_LSB_IDX         24
76 #define CSM0_BIT_LSB_IDX          3
77 #define PRIO0_BIT_LSB_IDX         0
78 #define CMS_BIT_LSB_IDX           8
79 #define FLUSH_BIT_LSB_IDX         10
80 #define TREV_BIT_LSB_IDX          9
81 #define ENGT_BIT_LSB_IDX          0
82 #define RESPOS_BIT_LSB_IDX        21
83 #define RESREG_BIT_LSB_IDX        16
84 #define ICLSEL_BIT_LSB_IDX        0
85 #define VF_BIT_LSB_IDX            31
86 #define RESULT_BIT_LSB_IDX        0
87 #define ASSCH7_BIT_LSB_IDX        7
```

VADC 레지스터 설정 – G4QINR0

Queue에 연결할 Input 채널 설정

1. VADC 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0020000**

2. 사용할 레지스터의 주소 찾기
   - **G4QINR0** 의 Offset Address (group = x = 4) = (x * 0x400) + 0x510 = **0x510**

   → G4QINR0 레지스터 주소 = 0xF0020000 + 0x1510 = **0xF0021510**

				P	
GxQINR0	Queue 0 Input Register, Group x	$X510_H$	U, SV	U, SV P	28-44
GxQ0R0	Queue 0 Register 0, Group x	$X50C_H$	U, SV	U, SV P	28-46

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4023

GxQINR0 (x = 0 - 7)
Queue 0 Input Register, Group x
($x * 0400_H + 0510_H$)

Reset Value: 0000 0000$_H$

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3918

VADC 레지스터 설정 – G4QINR0

: Queue에 연결할 Input 채널 설정 -7번 채널 읽기

3. 레지스터 write 값 결정
   - Group 4 의 Request Source 0 을 사용하므로 G4QINR"0" 레지스터 설정 필요
   - Analog Input Channel 7 을 입력으로 사용하기 위해 **REQCHNR 영역에 0x7 write**

G4QINR0 레지스터 @ 0xF0021510

GxQINR0 (x = 0 - 7)
Queue 0 Input Register, Group x

$(x * 0400_H + 0510_H)$ Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	EXTR	ENSI	RF	REQCHNR				
r	r	r	r	r	r	r	r	w	w	w	w				

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3918

Input Channel 7

0x7 값 write

Field	Bits	Type	Description
REQCHNR	[4:0]	w	~~Request Channel Number~~ Defines the channel number to be converted

Lab15 - VADC 레지스터 설정 – G4QINR0

: Queue에 연결할 Input 채널 설정 – 7번채널

```
325
326 void VADC_startConversion(void)
327 {
328     VADC_G4_QINR0.U |= 0x07;                    // no. of Request Channel = 7
329
330     VADC_G4_QMR0.U |= 0x1 << TREV_BIT_LSB_IDX;  // Generate Conversion Start Trigger Event
331 }
332
333 unsigned int VADC_readResult(void)
334 {
335     unsigned int result;
336
337     while( (VADC_G4_RES0.U & (0x1 << VF_BIT_LSB_IDX)) == 0 );   // wait until read available
338     result = VADC_G4_RES0.U & (0xFFF << RESULT_BIT_LSB_IDX);    // read ADC value
339
340     return result;
341 }
342
```

```
71 // VADC registers
72 #define DISS_BIT_LSB_IDX            1
73 #define DISR_BIT_LSB_IDX            0
74 #define ANONC_BIT_LSB_IDX           0
75 #define ASEN0_BIT_LSB_IDX           24
76 #define CSM0_BIT_LSB_IDX            3
77 #define PRIO0_BIT_LSB_IDX           0
78 #define CMS_BIT_LSB_IDX             8
79 #define FLUSH_BIT_LSB_IDX           10
80 #define TREV_BIT_LSB_IDX            9
81 #define ENGT_BIT_LSB_IDX            0
82 #define RESPOS_BIT_LSB_IDX          21
83 #define RESREG_BIT_LSB_IDX          16
84 #define ICLSEL_BIT_LSB_IDX          0
85 #define VF_BIT_LSB_IDX              31
86 #define RESULT_BIT_LSB_IDX          0
87 #define ASSCH7_BIT_LSB_IDX          7
```

VADC 변환 결과 저장 레지스터 Read
: ADC 변환 결과를 읽어오기

- G4CHCTR7 레지스터에서 ADC 변환이 완료된 데이터를 저장할 Result Register를 선택하였음

→ **선택한 x번째 Result Register 에 ADC 변환 결과가 저장됨**

- VADC 레지스터 항목에서
  - **G4RESx** 레지스터 Read (x = 나중에 설정할 Result Register 번호)

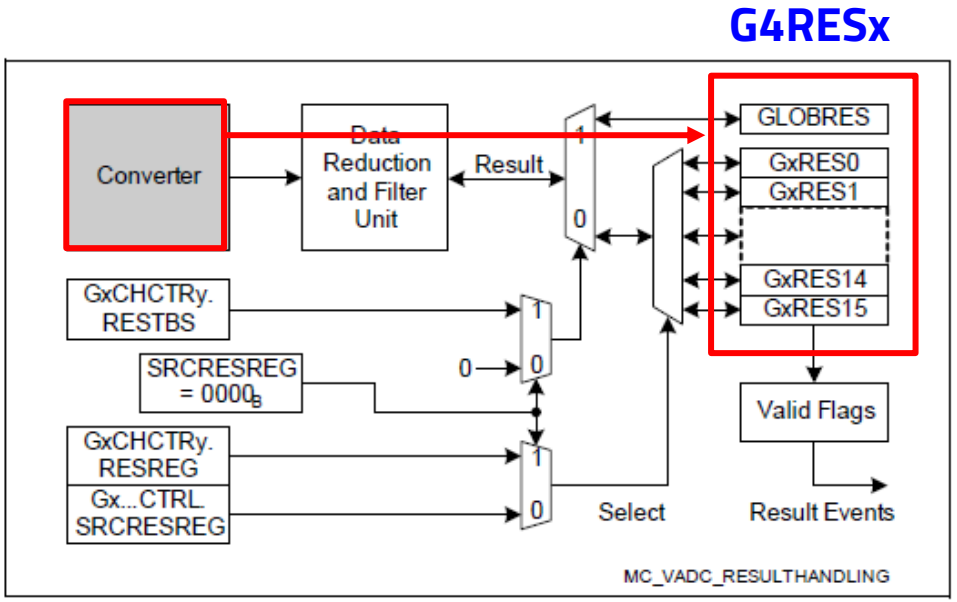


Figure 28-18 Conversion Result Storage

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3969

VADC 레지스터 Read – G4RES0

: VADC 변환 결과를 읽을 수 있는 레지스터

1. VADC 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0020000**

2. 사용할 레지스터의 주소 찾기
   - G4CHCTR7 레지스터에서 ADC 변환 결과를 **Result Register 0** 에 저장하도록 설정
   - **G4RES0** 의 Offset Address (Result Register = y = 0) = 0x1700 + (y * 0x4) = **0x1700**

   → G4RES0 레지스터 주소 = 0xF0020000 + 0x1700 = **0xF0021700**

Register Short Name	Register Long Name	Offset Addr.	Access Mode Read	Access Mode Write	Page Num.
GxRESy	Group x Result Register y	$X70Y_H$	U, SV	U, SV P	28-98
GxRESDy	Group x Result Register y (debug view)	$X70Y_H$	U, SV	U, SV P	28-100

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.4025

G4RESy (y = 0 - 15)

Group 4 Result Register y | **$(1700_H + y * 0004_H)$** | **Reset Value: $0000\ 0000_H$**

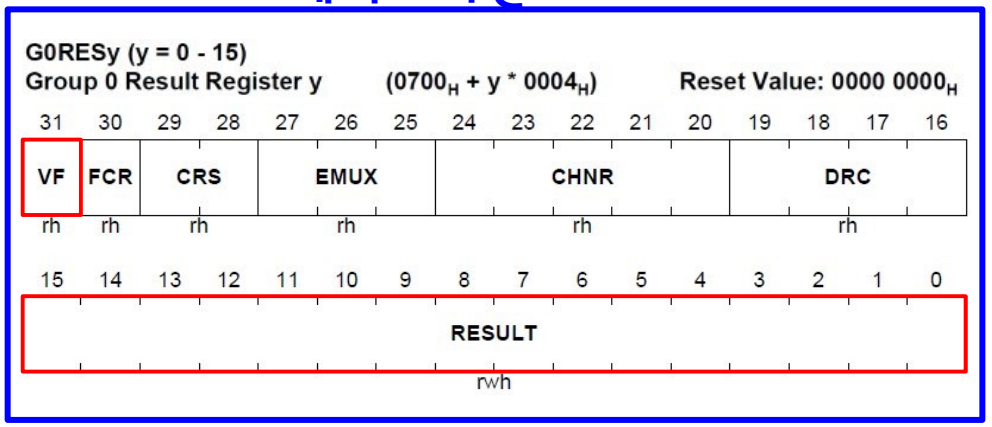
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3972

VADC 레지스터 Read – G4RES0

: VADC 변환 완료를 알려주는 flag 및 변환 결과 read

3. 레지스터 값 Read
   - G4CHCTR7 레지스터에서 Analog Input Channel 7에 대한 ADC 변환 결과를 Result Register 0 에 저장하도록 설정함
   - **변환이 끝나고 새로운 ADC 변환 결과가 레지스터에 저장되면 VF 영역의 값이 0x1 이 됨**
   - **변환된 ADC 값은 RESULT 영역에 존재 (Align 을 고려해야 함)**

G4RES0 레지스터 @ 0xF0021700



Field	Bits	Type	Description
RESULT	[15:0]	rwh	**Result of Most Recent Conversion** The position of the result bits within this bitfield depends on the configured operating mode. Refer to Section 28.7.2.
VF	31	rh	**Valid Flag** Indicates a new result in bitfield RESULT or bit FCR. 0_B No new result available 1_B Bitfield RESULT has been updated with new result value and has not yet been read, or bit FCR has been updated

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3972

Lab16: VADC 레지스터 Read – G4RES0

: VADC 변환 완료를 알려주는 flag 및 변환 결과 read

```
325
326 void VADC_startConversion(void)
327 {
328     VADC_G4_QINR0.U |= 0x07;                    // no. of Request Channel = 7
329
330     VADC_G4_QMR0.U |= 0x1 << TREV_BIT_LSB_IDX;  // Generate Conversion Start Trigger Event
331 }
332
333 unsigned int VADC_readResult(void)
334 {
335     unsigned int result;
336
337     while( (VADC_G4_RES0.U & (0x1 << VF_BIT_LSB_IDX)) == 0 );   // wait until read available
338     result = VADC_G4_RES0.U & (0xFFF << RESULT_BIT_LSB_IDX);    // read ADC value
339
340     return result;
341 }
342
```

```
71 // VADC registers
72 #define DISS_BIT_LSB_IDX            1
73 #define DISR_BIT_LSB_IDX            0
74 #define ANONC_BIT_LSB_IDX           0
75 #define ASEN0_BIT_LSB_IDX           24
76 #define CSM0_BIT_LSB_IDX            3
77 #define PRIO0_BIT_LSB_IDX           0
78 #define CMS_BIT_LSB_IDX             8
79 #define FLUSH_BIT_LSB_IDX           10
80 #define TREV_BIT_LSB_IDX            9
81 #define ENGT_BIT_LSB_IDX            0
82 #define RESPOS_BIT_LSB_IDX          21
83 #define RESREG_BIT_LSB_IDX          16
84 #define ICLSEL_BIT_LSB_IDX          0
85 #define VF_BIT_LSB_IDX              31
86 #define RESULT_BIT_LSB_IDX          0
87 #define ASSCH7_BIT_LSB_IDX          7
```

Lab17: Main 함수

- main 함수 작성

초기화 함수 호출

1. ADC 변환 request 를 발생시키고
2. ADC 변환이 완료되면 결과를 변수 *adcResult* 에 저장

가변 저항에서 읽은 ADC 값의 범위에 따라 RGB LED on/off 결정

if(adc값 >= 3096)
Red ON
else if(adc값 >= 2048)
Green ON
else if(adc값 >= 1024)
Blue ON
else
Red, Green, Blue ON

```
130     //initERU();
131     //initCCU60();
132     initLED();
133     initRGBLED();
134     initVADC();
135     //initButton();
136
137     unsigned int adcResult;
138
139     while(1)
140     {
141         VADC_startConversion();
142         adcResult = VADC_readResult();
143
144         if( adcResult >= 3096 )
145         {
146             P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
147             P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
148             P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
149         }
150         else if( adcResult >= 2048 )
151         {
152             P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
153             P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
154             P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
155         }
156         else if( adcResult >= 1024 )
157         {
158             P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
159             P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
160             P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
161         }
162         else
163         {
164             P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
165             P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
166             P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
167         }
168
169     }
170     return (1);
171 }
172
```

SW 프로그래밍

: VADC 레지스터의 각 영역(필드) LSB 비트 시작 위치 define 정의

- 레지스터에 값을 write할 때 shift되는 offset을 쉽게 사용하기 위한 define 작성

```
30
31 #include "IfxCcu6_reg.h"
32 #include "IfxVadc_reg.h"
33
34 // Port registers
35 #define PC1_BIT_LSB_IDX            11
36 #define PC2_BIT_LSB_IDX            19
37 #define PC3_BIT_LSB_IDX            27
38 #define PC5_BIT_LSB_IDX            11
39 #define PC7_BIT_LSB_IDX            27
40 #define P1_BIT_LSB_IDX             1
41 #define P2_BIT_LSB_IDX             2
42 #define P3_BIT_LSB_IDX             3
43 #define P5_BIT_LSB_IDX             5
44 #define P7_BIT_LSB_IDX             7
45
46 // SCU registers
47 #define LCK_BIT_LSB_IDX            1
48 #define ENDINIT_BIT_LSB_IDX        0
49 #define EXIS0_BIT_LSB_IDX          4
50 #define FEN0_BIT_LSB_IDX           8
51 #define EIEN0_BIT_LSB_IDX          11
52 #define INP0_BIT_LSB_IDX           12
53 #define IGP0_BIT_LSB_IDX           14
54
55 // SRC registers
56 #define SRPN_BIT_LSB_IDX           0
57 #define TOS_BIT_LSB_IDX            11
58 #define SRE_BIT_LSB_IDX            10
59
60 // CCU60 registers
61 #define DISS_BIT_LSB_IDX           1
62 #define DISR_BIT_LSB_IDX           0
63 #define CTM_BIT_LSB_IDX            7
64 #define T12PRE_BIT_LSB_IDX         3
65 #define T12CLK_BIT_LSB_IDX         0
66 #define T12STR_BIT_LSB_IDX         6
67 #define T12RS_BIT_LSB_IDX          1
68 #define INPT12_BIT_LSB_IDX         10
69 #define ENT12PM_BIT_LSB_IDX        7
70
```

헤더 파일 참조 추가

Port 레지스터 bit shift offset

```
71 // VADC registers
72 #define DISS_BIT_LSB_IDX           1
73 #define DISR_BIT_LSB_IDX           0
74 #define ANONC_BIT_LSB_IDX          0
75 #define ASEN0_BIT_LSB_IDX          24
76 #define CSM0_BIT_LSB_IDX           3
77 #define PRIO0_BIT_LSB_IDX          0
78 #define CMS_BIT_LSB_IDX            8
79 #define FLUSH_BIT_LSB_IDX          10
80 #define TREV_BIT_LSB_IDX           9
81 #define ENGT_BIT_LSB_IDX           0
82 #define RESPOS_BIT_LSB_IDX         21
83 #define RESREG_BIT_LSB_IDX         16
84 #define ICLSEL_BIT_LSB_IDX         0
85 #define VF_BIT_LSB_IDX             31
86 #define RESULT_BIT_LSB_IDX         0
87 #define ASSCH7_BIT_LSB_IDX         7
88
```

VADC 레지스터 bit shift offset

SW 프로그래밍
: RGB LED 사용 위한 GPIO 핀 초기화

- RGB LED 사용을 위한 초기화 함수 *initRGBLED()*

```
268
269  void initRGBLED(void)
270  {
271      // reset Port IOCR register
272      P02_IOCR4.U &= ~(0x1F << PC7_BIT_LSB_IDX);
273      P10_IOCR4.U &= ~(0x1F << PC5_BIT_LSB_IDX);
274      P10_IOCR0.U &= ~(0x1F << PC3_BIT_LSB_IDX);
275
276      // set Port as general purpose output (push-pull)
277      P02_IOCR4.U |= 0x10 << PC7_BIT_LSB_IDX;
278      P10_IOCR4.U |= 0x10 << PC5_BIT_LSB_IDX;
279      P10_IOCR0.U |= 0x10 << PC3_BIT_LSB_IDX;
280  }
281
```

SW 프로그래밍
: VADC 사용 위한 아날로그 입력 핀 초기화

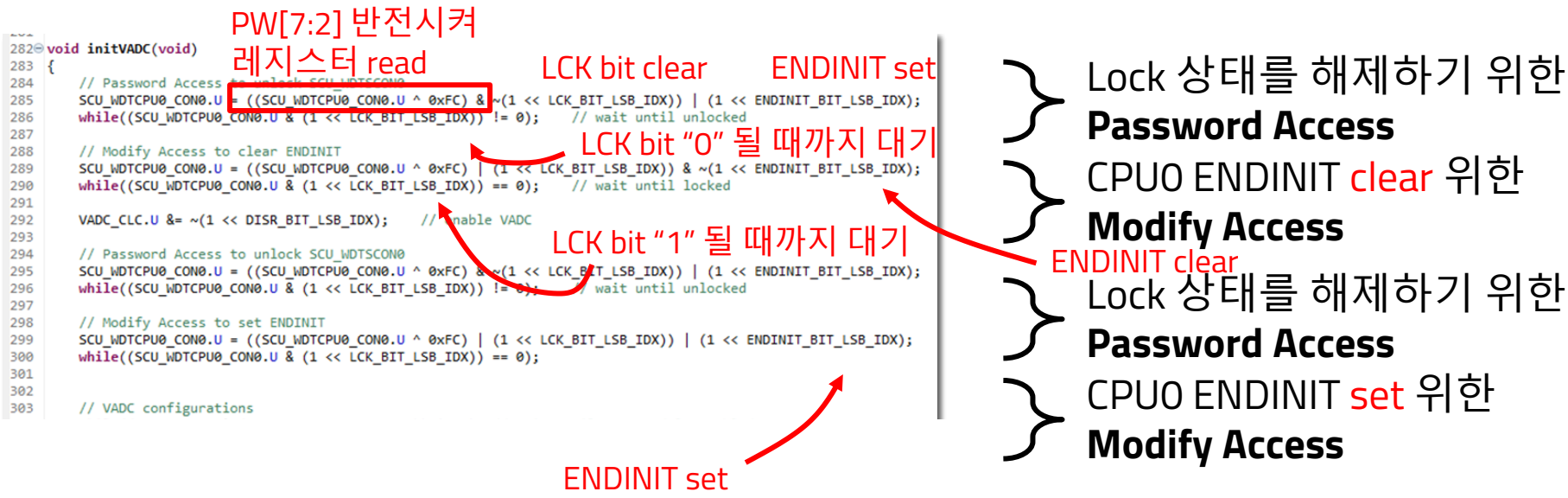
- VADC 모듈 사용을 위한 초기화 함수 *initVADC()*

```
282 void initVADC(void)
283 {
284     // Password Access to unlock SCU_WDTSCON0
285     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
286     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
287
288     // Modify Access to clear ENDINIT
289     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
290     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
291
292     VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable VADC
293
294     // Password Access to unlock SCU_WDTSCON0
295     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
296     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
297
298     // Modify Access to set ENDINIT
299     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
300     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
301
302
303     // VADC configurations
304     while((VADC_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until VADC module enabled
305
306     VADC_G4_ARBPR.U |= 0x3 << PRIO0_BIT_LSB_IDX;         // highest priority for Request Source 0
307     VADC_G4_ARBPR.U &= ~(0x1 << CSM0_BIT_LSB_IDX);       // Wait-for-Start Mode
308     VADC_G4_ARBPR.U |= 0x1 << ASEN0_BIT_LSB_IDX;         // Arbitration Source Input 0 Enable
309
310     VADC_G4_QMR0.U &= ~(0x3 << ENGT_BIT_LSB_IDX);
311     VADC_G4_QMR0.U |= 0x1 << ENGT_BIT_LSB_IDX;          // enable conversion request
312     VADC_G4_QMR0.U |= 0x1 << FLUSH_BIT_LSB_IDX;         // clear ADC queue
313
314     VADC_G4_ARBCFG.U |= 0x3 << ANONC_BIT_LSB_IDX;       // ADC normal operation
315
316     VADC_G4_ICLASS0.U &= ~(0x7 << CMS_BIT_LSB_IDX);     // Class 0 Standard Conversion (12-bit)
317
318
319     // VADC Group 4 Channel 7 configuration
320     VADC_G4_CHCTR7.U |= 0x1 << RESPOS_BIT_LSB_IDX;      // result right-aligned
321     VADC_G4_CHCTR7.U &= ~(0xF << RESREG_BIT_LSB_IDX);   // store result @ Result Register G4RES0
322     VADC_G4_CHCTR7.U &= ~(0x3 << ICLSEL_BIT_LSB_IDX);   // Class 0
323
324     VADC_G4_CHASS.U |= 0x1 << ASSCH7_BIT_LSB_IDX;
325 }
326
```

SW 프로그래밍

: VADC 모듈 사용 설정 전, 보호 레지스터 잠금 해제 필요
: VADC 모듈 사용 설정 후, 보호 레지스터 잠금 재설정 필요

- VADC 모듈 사용을 위한 초기화 함수 *initVADC()*




```
void initVADC(void)
{
    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked

    // Modify Access to clear ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked

    VADC_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // Enable VADC

    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked

    // Modify Access to set ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);


    // VADC configurations
```

SW 프로그래밍
: VADC 모듈 설정

- VADC 모듈 사용을 위한 초기화 함수 *initVADC()*

```
302
303     // VADC configurations
304     while((VADC_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until VADC module enabled
305
306     VADC_G4_ARBPR.U |= 0x3 << PRIO0_BIT_LSB_IDX;         // highest priority for Request Source 0
307     VADC_G4_ARBPR.U &= ~(0x1 << CSM0_BIT_LSB_IDX);       // Wait-for-Start Mode
308     VADC_G4_ARBPR.U |= 0x1 << ASEN0_BIT_LSB_IDX;         // Arbitration Source Input 0 Enable
309
310     VADC_G4_QMR0.U &= ~(0x3 << ENGT_BIT_LSB_IDX);
311     VADC_G4_QMR0.U |= 0x1 << ENGT_BIT_LSB_IDX;           // enable conversion request
312     VADC_G4_QMR0.U |= 0x1 << FLUSH_BIT_LSB_IDX;          // clear ADC queue
313
314     VADC_G4_ARBCFG.U |= 0x3 << ANONC_BIT_LSB_IDX;        // ADC normal operation
315
316     VADC_G4_ICLASS0.U &= ~(0x7 << CMS_BIT_LSB_IDX);      // Class 0 Standard Conversion (12-bit)
317
318
319     // VADC Group 4 Channel 7 configuration
320     VADC_G4_CHCTR7.U |= 0x1 << RESPOS_BIT_LSB_IDX;       // result right-aligned
321     VADC_G4_CHCTR7.U &= ~(0xF << RESREG_BIT_LSB_IDX);    // store result @ Result Register G4RES0
322     VADC_G4_CHCTR7.U &= ~(0x3 << ICLSEL_BIT_LSB_IDX);    // Class 0
323
324     VADC_G4_CHASS.U |= 0x1 << ASSCH7_BIT_LSB_IDX;
325 }
```

VADC 모듈 설정
- VADC 모듈 enable
- Request Source 0 의 우선 순위 설정
- Source Input 0 enable
- Conversion Request enable
- Clear ADC Request Queue
- ADC normal operation 설정
- ADC 변환 결과 right-align, 12-bit 길이 설정
- ADC 변환 결과 저장될 레지스터 설정
- Input Channel 7 최우선순위 설정

SW 프로그래밍
: VADC 변환을 요청(시작)

- VADC 모듈에 ADC 변환을 요청하는 함수 *VADC_startConversion()*

```
325
326 void VADC_startConversion(void)
327 {
328     VADC_G4_QINR0.U |= 0x07;                    // no. of Request Channel = 7
329
330     VADC_G4_QMR0.U |= 0x1 << TREV_BIT_LSB_IDX;  // Generate Conversion Start Trigger Event
331 }
332
```

아날로그 입력 중,
Input Channel 7 사용 설정

ADC에 변환 요청하는
request event 신호 발생
→ 변환 시작

SW 프로그래밍
: VADC 변환결과를 read 해서 변수에 저장

- ADC 변환 결과를 반환하는 함수 *VADC_readResult()*

```
332
333 unsigned int VADC_readResult(void)
334 {
335     unsigned int result;
336
337     while( (VADC_G4_RES0.U & (0x1 << VF_BIT_LSB_IDX)) == 0 );   // wait until read available
338     result = VADC_G4_RES0.U & (0xFFF << RESULT_BIT_LSB_IDX);    // read ADC value
339
340     return result;
341 }
342
```

ADC 변환이 완료될 때까지 대기

32bit 변수 *result* 에
ADC 변환 결과가 저장된
레지스터의 값을 저장

ADC 변환의 결과가 12bit 이므로
→ 0xFFF 로 bit masking

SW 프로그래밍

: VADC 모듈에서 변환된 디지털 값 사용 RGB LED 제어

- main 함수 작성

초기화 함수 호출

1. ADC 변환 request 를 발생시키고
2. ADC 변환이 완료되면 결과를 변수 *adcResult* 에 저장

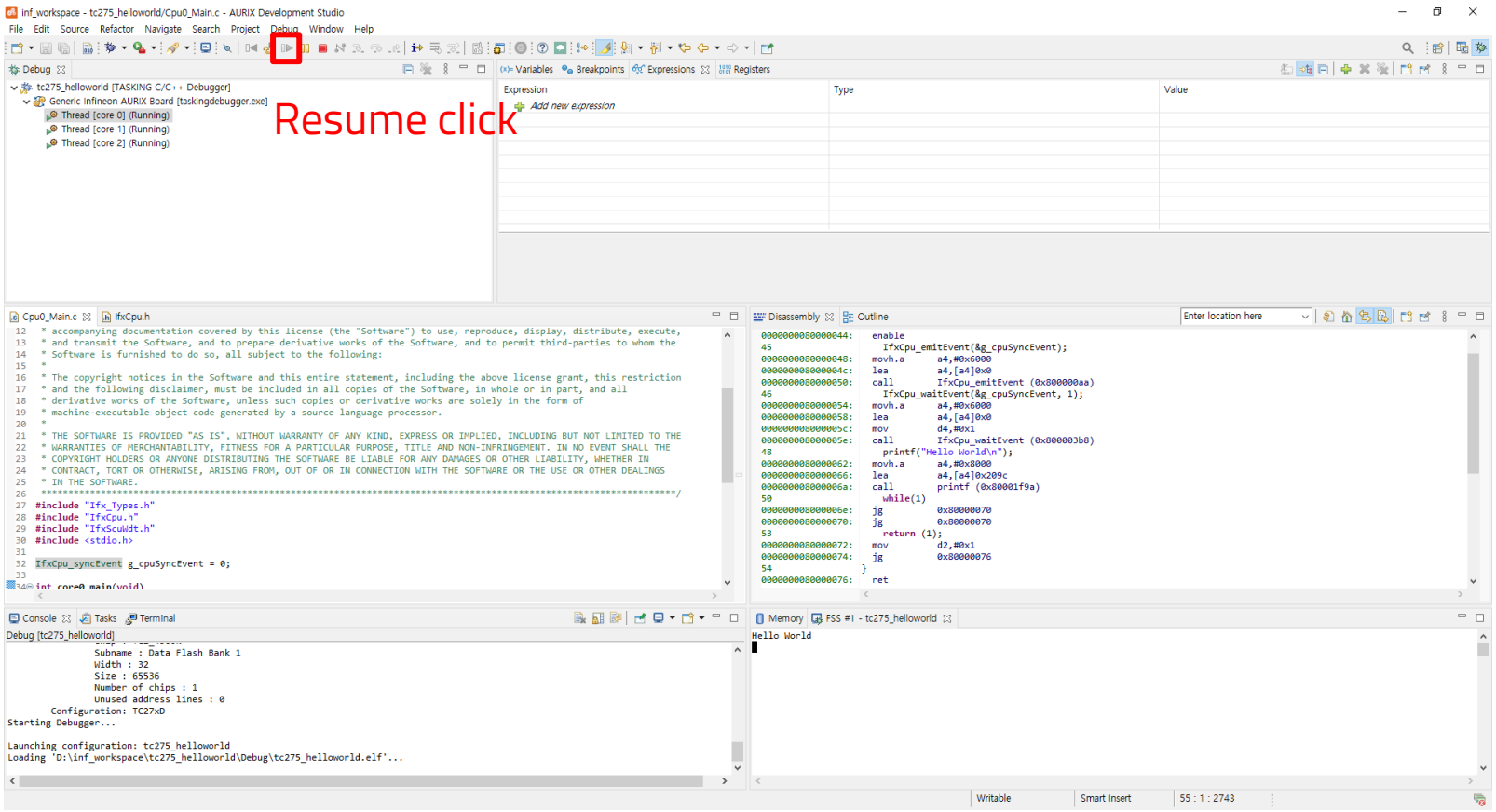
가변 저항에서 읽은
ADC 값의 범위에 따라
RGB LED on/off 결정

if(adc값 >= 3096)
Red ON
else if(adc값 >= 2048)
Green ON
else if(adc값 >= 1024)
Blue ON
else
Red, Green, Blue ON

```
130     //initERU();
131     //initCCU60();
132     initLED();
133     initRGBLED();
134     initVADC();
135     //initButton();
136
137     unsigned int adcResult;
138
139     while(1)
140     {
141         VADC_startConversion();
142         adcResult = VADC_readResult();
143
144         if( adcResult >= 3096 )
145         {
146             P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
147             P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
148             P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
149         }
150         else if( adcResult >= 2048 )
151         {
152             P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
153             P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
154             P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
155         }
156         else if( adcResult >= 1024 )
157         {
158             P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
159             P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
160             P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
161         }
162         else
163         {
164             P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
165             P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
166             P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
167         }
168
169     }
170     return (1);
171 }
172
```

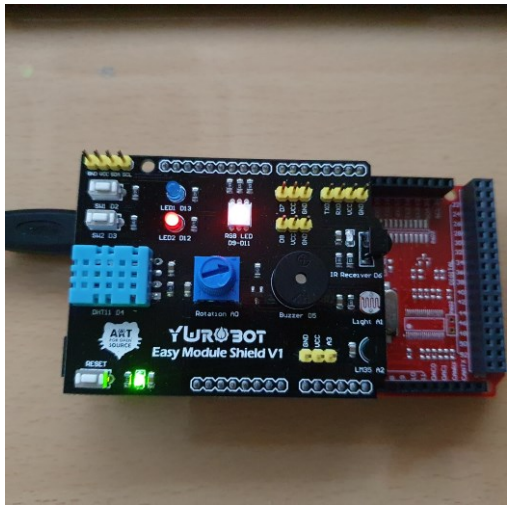
Build 및 Debug

- 프로젝트 빌드 (ctrl + b)
- 디버그 수행하여 보드에 실행 파일 flash

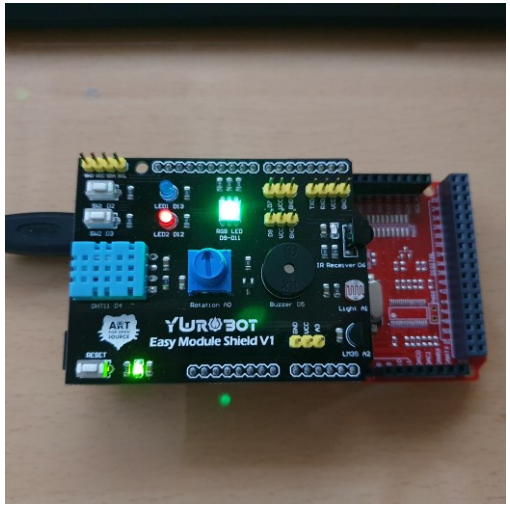


동작 확인

- 가변 저항을 돌리면 변화하는 ADC 값에 따라 on되는 RGB LED 색이 변하는 모습 확인 가능

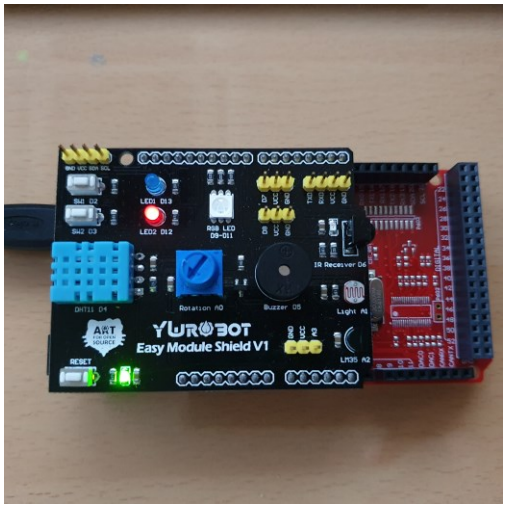


adcValue >= 3096



adcValue >= 2048

adcValue >= 1024



adcValue < 1024

감사합니다. 휴식~~

FAQ

- Open drain타입 통신이란, 그리고 사용이유?
- ADC센싱 특성이 아래쪽 위쪽이 좋지 않은 이유
- 샘플링 시작했을 때 실제 감지하는 시점은 언제인지?