



AUTOSAR 4.1.1의 최신 기술과 이해

Part 4 : AUTOSAR MCAL

2014년 01월호 지면기사 / 글 | 김 현 구 <heari@infobank.net> , Infobank

’0&f_size=13)  [프린트](#)

:/articleView.asp%3Fidx=1270&t=AUTOSAR%204.1.1의%20최신%20기술과%20이해) 

 (https://story.kakao.com/s/share?

3D1270&text=AUTOSAR%204.1.1%EC%9D%98%20%EC%B5%9C%EC%8B%A0%20%EA%B8%B0%EC%88%A0%EA%B3%BC%20%EC%9D%B4%ED%95%B4%20-CAL&kakao_agent=sdk%2F1.37.2%20os%2Fjavascript%20lang%2Fko-KR%20device%2FWin32%20origin%2Fhttp%253A%252F%252Fwww.autoelectronics.co.kr)

채승엽, 박대영, 이우승, 이현철 등 인포뱅크(Infobank) AUTOSAR 사업부 멤버가 AUTOSAR 최신 기술에 대한 아티클을 6회에 걸쳐 연재한다. 인포뱅크 사업부는 AUTOSAR 추세와 요구사항에 맞게 AUTOSAR Ethernet Solution, AUTOSAR 플랫폼 교육, SWC 개발 Tool과 같은 AUTOSAR 전반에 걸친 업무를 수행하고 있다. 네 번째 회는 ‘AUTOSAR MCAL’이다.

1. AUTOSAR의 실무 이해
2. AUTOSAR OS
3. AUTOSAR RTE
4. AUTOSAR MCAL
5. AUTOSAR CAN
6. AUTOSAR FlexRay

지난 호에 이어 이번 연재에서는 차량용 반도체 회사에서 반드시 지원해야 하는 AUTOSAR용 Device Driver인 MCAL(Microcontroller Abstraction Layer)을 주제로 설명하고자 한다.

먼저 MCAL에 대해 간략히 소개하고, 종류별로 어떠한 기능에 각각 사용하는지 설명한다. MCAL의 종류가 많은 관계로 상세히 하나하나 설명하기에는 무리가 있다. 그래서 실무에서 사용하는 간단한 DIO Control하기 위한 시스템 설계를 해 ARXML을 작성하고 작성된 ARXML을 Generate해서 각각의 함수 호출관계를 도식화해 설명할 예정이다. 시스템 설계단계는 AUTOSAR 개발 단계의 1단계에 해당해 MCAL에 대해 전혀 고려해야할 대상이 아닌 것 같지만, 시스템 설계 단계에서 MCAL의 제어를 위한 Interface 과정에서 함수인자(Argument)를 MCAL 제어를 위해 동일한 데이터 형(Data Type)으로 정의를 해야지만 이상동작 없이 제어(Control)할 수 있다.

본 연재에 소개될 예제에서는 MCAL을 시스템 설계 단계에서부터 설명하기 위해서 참고용으로 설계한 것이니 참고하기 바란다. 그리고, MCAL에서 생성된 소스코드는 사용할 Generator Tool과 독자 Parameter에 따라서 소스가 상이할 수 있으니 주의하기 바란다.

MCAL이란 무엇인가

한마디로 말하면 Device Driver라고 할 수 있다. MCAL로 제공된 Device Driver는 OS가 제공되지 않는 Embedded 환경에서도 동작이 가능하다는 것이다.

즉, Firmware Level에서는 Device Driver를 제어(Control)할 수 있도록 제공되는 것이 Driver인데 AUTOSAR용 MCAL은 AUTOSR에서 규정한 API 이름을 사용해 Device Driver를 제어할 수 있도록 제공한다. 차량용 반도체 회사에서는 AUTOSAR에서 규정한 API만으로는 각 MCU의 고유한 특징을 제어하기에 부족할 경우에는 독자적인 Parameter를 제공할 하고 있다. AUTOSAR용 MCAL의 가장 큰 특징은 XML 형식을 이용한 소스코드 자동 생성이므로, MCU Vendor에 의존한 독자 Parameter를 제공할 경우 XML 형식으로 독자 Parameter를 설정할 수 있어야 하며 자동으로 독자 Parameter의 확장 API 소스코드를 생성할 수 있도록 제공해야만 한다.

MCAL은 AUTOSAR Software Architec ture상 다음 그림처럼 Microcontroller 위에 위치 해 Microcontroller를 직접적으로 컨트롤할 수 있다.

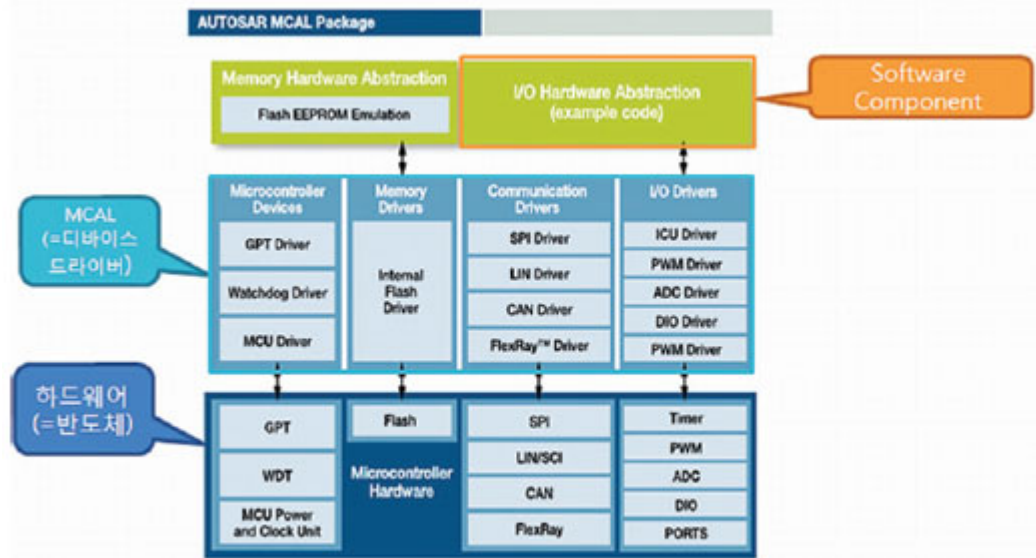


그림 1 | AUTOSAR Software Architecture – MCAL Layer

MCU Vendor에서 제공하는 독자 Parameter와 AUTOSAR에서 규정한 Parameter를 구분해 분석하기 위해서 인포뱅크는 자체 개발한 Tool을 이용한다. 그림 2와 같이 ARXML을 엑셀로 자동 변환하면 쉽게 독자 Parameter에 이해를 높일 수 있다.

독자 Parameter와 AUTOSAR에서 규정한 Parameter를 구분하는 Tag는 그림 2에서 Origin이다. 필드 값이 AUTOSAR_ECUC이면 AUTOSAR에서 제공하는 규정된 Parameter이며, 독자 Parameter는 MCUVendor의 이름(예, Freescale)으로 표기된다.

MCAL은 아래와 같이 크게 4가지로 분류된다.

1. Microcontroller Devices
2. Memory Driver
3. Communication Driver
4. I/O Driver

위의 4가지 분류에는 각각의 MCAL이 존재하는데 MCAL의 종류에 대해서 하나씩 살펴보도록 하겠다.

1. Microcontroller Devices:

① GPT(General Purpose Timer)는 MCU driver의 prescaler와 PLL을 사용해 HW timer(OS사용)로 사용되면 time에 관련 된 interrupt, wakeup에 사용할 수가 있다. GPT 사용을 위해서는 MCU의 Parameter 설정과정에서 설정해줘야만 하는 Parameter들이 있다. MCU Vendor에서 제공하는 독자 Parameter 부분도 상당부분이 존재한다.



Gpt_SetMode를 사용해 Stop 또는 Active 상태로 상태를 천이하고 Gpt_GetTime Elapsed를 사용해 주기적인 시간 기준으로 경과 시간을 알아낼 수 있다. Gpt_GetTime Remaining을 사용하면 주기적인 시간 기준으로 남은 시간에 대해서 알아낼 수 있다.

(http://ssl.logger.co.kr/tracker_ad.tsp?u=37061&m=C&adCode=57236)

(http://ssl.logger.co.kr/tracker_ad.tsp?u=37061&m=C&adCode=77860)

과월호 e-Book 보기

(/ebook/list.asp)



(/ebook/list.asp)(/ebook/list.asp)

News & Analysis

2륜차 리어램프에 최적! 4ch 리니어 LED 드라이버

(/article/articleView.asp?idx=3473)

다쏘시스템, 조메트리와 파트너십 체결...

(/article/articleView.asp?idx=3472)

포레스터 리서치, 산업용 IoT SW 플랫폼 리더로 마인드스피어 선정

(/article/articleView.asp?idx=3471)

[보도자료]ST, IIoT와 자동차 애플리케이션을 위한 안전한 셀룰러 연결 제공

(/article/articleView.asp?idx=3470)

뤼츠 시스템 솔루션즈, 자동차 이더넷 테스트(ATE) 발표

(/article/articleView.asp?idx=3469)

(https://www.drivingthenation.com)

(http://smartn.co.kr/book/book_detail.asp?p_no=B00159)

[Can Configuration parameter]									
Index	Parameter	Value	Unit	Parameter	Value	Unit	Parameter	Value	Unit
1	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
2	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
3	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
4	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
5	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
6	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
7	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
8	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
9	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
10	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
11	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
12	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
13	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
14	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
15	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
16	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
17	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
18	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
19	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
20	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
21	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
22	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
23	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
24	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
25	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
26	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
27	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
28	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
29	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
30	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
31	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
32	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
33	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
34	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
35	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
36	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
37	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
38	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
39	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
40	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
41	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
42	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
43	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
44	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
45	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
46	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
47	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
48	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
49	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
50	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
51	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
52	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
53	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
54	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
55	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
56	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
57	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
58	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
59	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
60	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
61	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
62	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
63	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
64	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
65	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
66	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
67	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
68	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
69	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
70	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
71	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
72	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
73	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
74	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
75	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
76	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
77	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
78	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
79	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
80	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
81	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
82	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
83	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
84	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
85	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
86	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
87	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
88	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
89	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
90	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
91	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
92	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
93	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
94	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
95	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
96	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
97	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
98	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
99	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t
100	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t	CanControllerId	0	uint8_t

그림 2 | AUTOSAR MCU Vendor Parameter – MCAL CAN

② WDT는 SW가 Logic상의 Error 등으로 인해 무한 루프 등에 빠지면 정상적인 동작을 못할 경우를 감시해 system reset 등을 수행하도록 한다. 종류는 원하던 상태로 Active되었는지를 Monitoring하는 WdgMAliveSupervision, 원하던 시간 안에 Logic이 처리되었는지를 Monitoring하는 WdgMDeadlineSupervision, 설계한 Logic대로 동작하는지를 Monitoring하는 WdgMExternalLogical Supervision 등이 있다.

③ MCU는 Clock, PLL을 초기화하고, RAM 영역의 Section들도 초기화한다. MCU는 절전모드를 활성화할 수 있으며 Reset을 할 수 있도록 한다. 그리고 Reset 원인에 대해서도 API로 제공한다. MCU는 Microcontroller 동작을 위해서 최우 선적으로 설정을 해야만 하는 Device Driver이다. MCU의 초기화 함수인 Mcu_Init은 AUTOSAR 표준을 준수한다면 EcuM.c의 EcuM_Init함수에서 호출돼야 한다.

2. Memory Driver:

① EEPROM, Flash Driver는 Memory를 control하기 위한 NvM(NV RAM Manager), MemIf(Memory Interface), Ea(EEPROM Abstraction), Fee(Flash EEPROM Emulation), Eep(EEPROM), Fls(Flash)에 관련된 Device Driver이다.

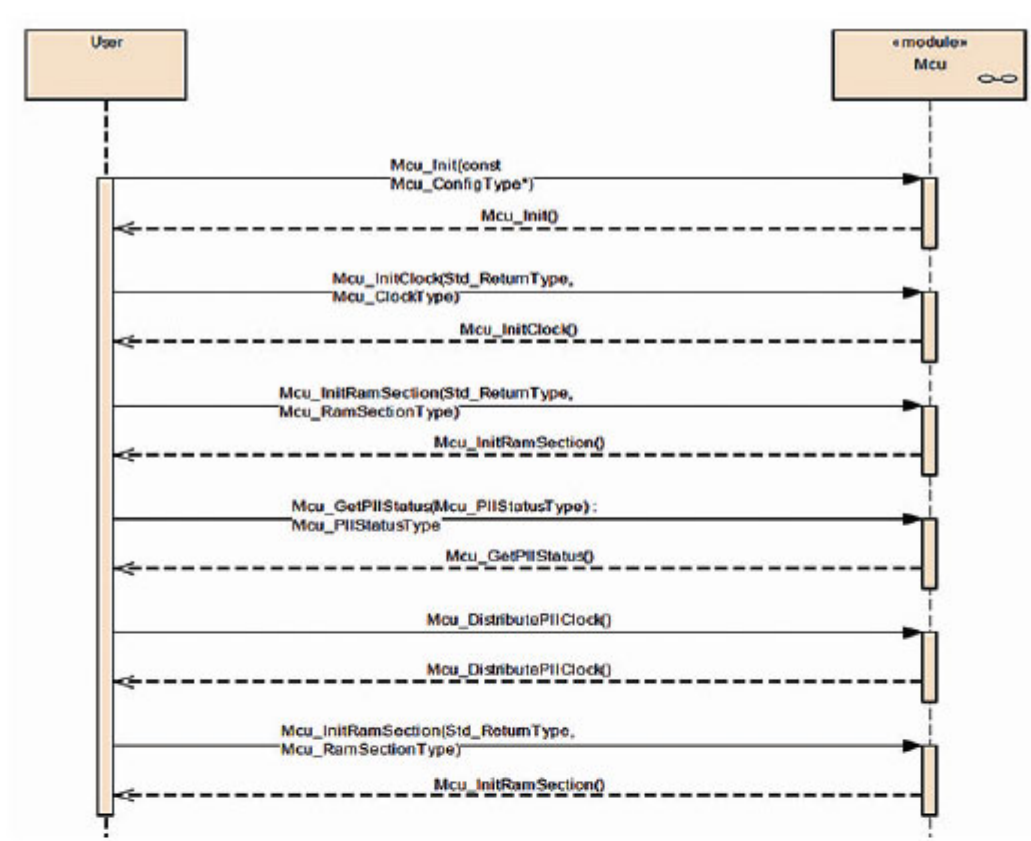


그림 3 | AUTOSAR MCAL MCU Initialization Sequence

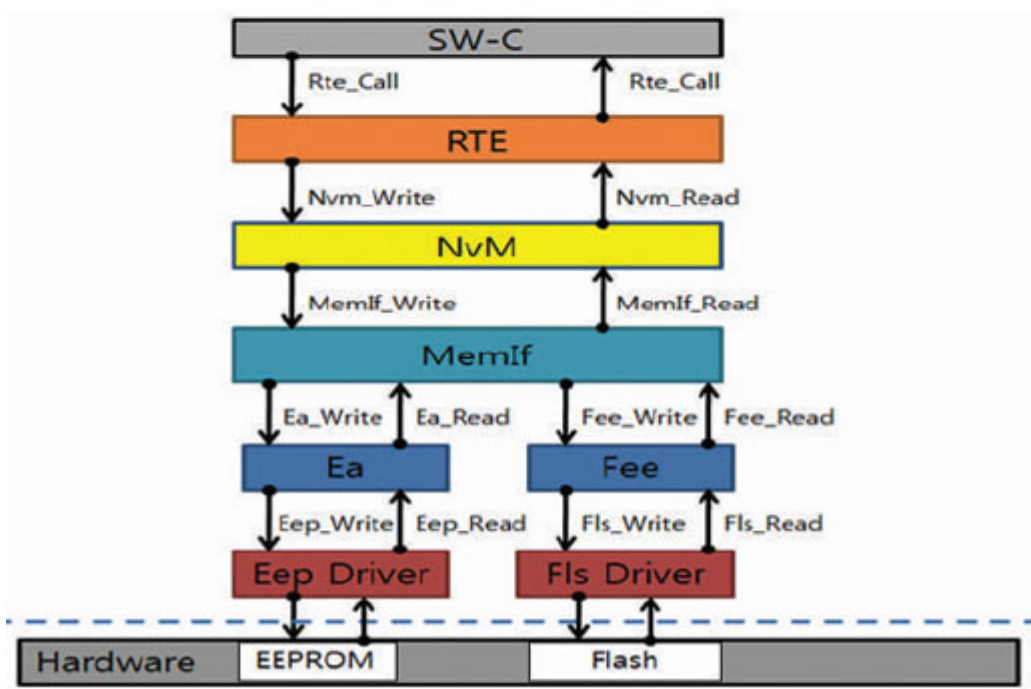


그림 4 | AUTOSAR Memory 전체 흐름

SW-C에서는 Memory와 관련해 NvM(NV RAM Manager), MemIf(Memory Interface)을 통해서만이 Memory Control을 할 수 있다. Memory 처리의 전체 흐름은 다음 그림과 같이 처리된다.

3. Communication Driver

① SPI는 확장성이 매우 높아 UART, PLC 등 외부 chip과 통신에서도 사용이 가능한 Device Driver이다. 가장 구현이 어려운 Device Driver이며, Tool 회사가 어디까지 지원하는지 반드시 확인하고 나서 개발이 시작돼야 한다. SPI는 100% 자동 생성으로 처리 되지 않음으로 제어 Logic을 I/O Hardware Abstraction SW-C, CDD(Complex Device Driver)로 구현해야만 한다.

4. I/O Driver

① ICU(Input Capture Unit)는 신호의 Rising edge Falling edge의 Signal을 설정하고 Notification을 받아내고 이를 분석해 사용하도록 하는 Device Driver이다.

Sequence는 Icu_Init(ConfigPtr)→Icu_

EnableWakeup(Ch1)→Icu_SetActivation Condition(Ch1)→Icu_SetActiation Condition(Ch1,

ICU_FALLING_EDGE)→Icu_EnableNotificaion(Ch1) 설정하고 나면 Falling edge가 발생하면 Icu_SignalNofiaion _Ch1이 호출된다.

호출된 함수에 Logic을 추가해서 기능을 구현하면 된다.

② PWM(Pulse width modulation)은 일정한 주기 내에서 Duty비를 변화시켜서 평균전압을 제어해 DC모터의 속도제어나 LED 등 헤드램프의 광량을 조절하는데 많이 사용된다. Duty Cycle은 1초 동안 반복되는 주기이며, 그 한 주기에 “HIGH Value를 유지한 %가 얼마인지가 Duty rate이다. Duty ratio가 30%일 때는 시간당 가해지는 신호의 평균값은 DC 신호의 30%가 된다. 즉, 30%의 DC신호가 가해지는 것과 거의 유사한 효과를 내는 것이다.

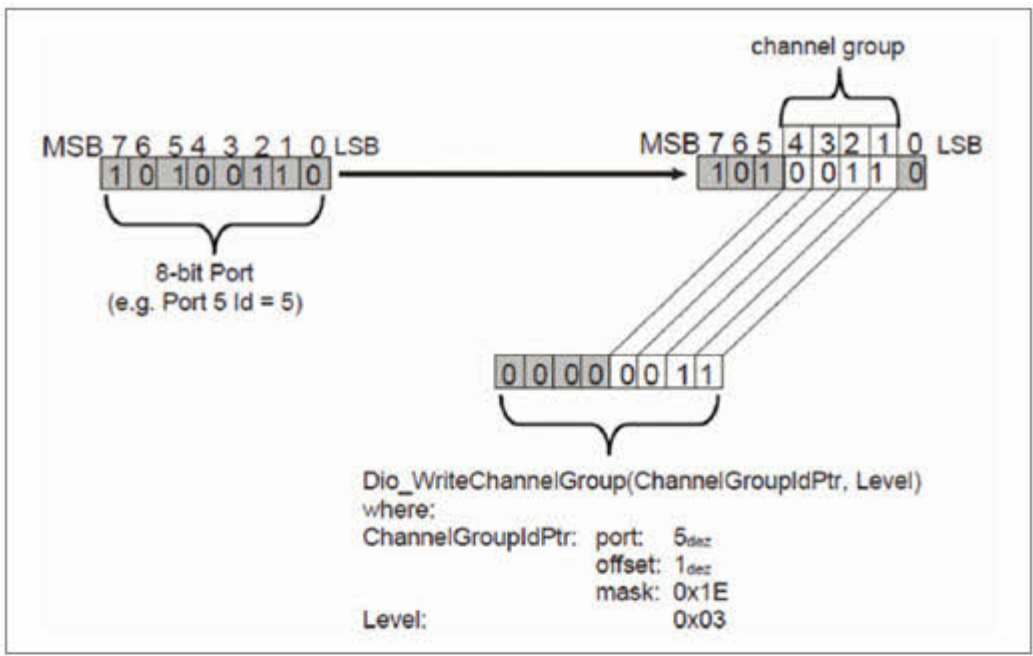


그림 5 | AUTOSAR MCAL DIO

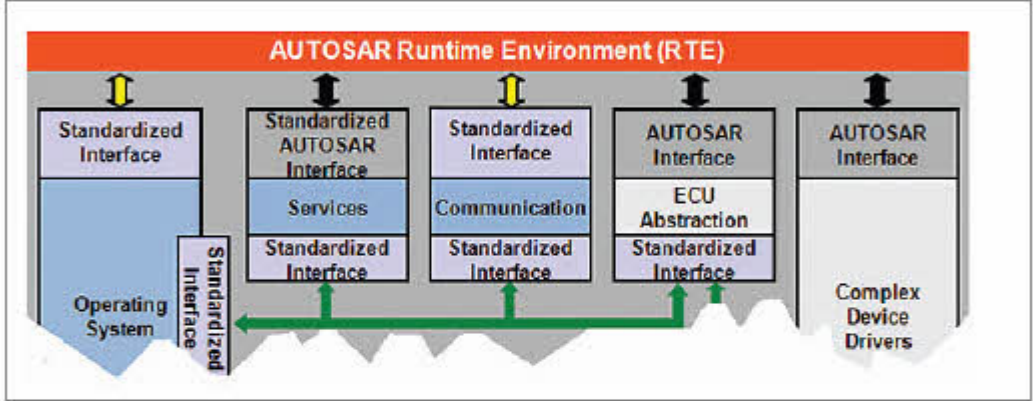


그림 6 | AUTOSAR I/O Hardware Abstraction-Interface



그림 7 | AUTOSAR 시스템 설계

Duty cycle설정에 사용되는 API는 Pwm_SetDutyCycle(ChannelNumber, DutyCycle)이며, 주기와 Duty설정 API는 Pwm_SetPeriodAndDuty(Pwm_ChannelType, Pwm_PeriodType, uint16)를 사용한다.

③ ADC(Analogue Digital Converter)는 Analogue 신호를 Digital 신호로 바꾸어서 읽어 들이도록 하는 Device Driver이다. ADC Channel과 ADC Channel Group으로 나뉘서 값을 읽어낼 수 있는데, ADC Channel은 ADC port 1개에 대해서만 읽어낼 때 사용되고, ADC Channel Group은 여러 개의 ADC Port가 하나의 기능을 위해 묶음으로 돼 있는 것을 읽어낼 때 사용된다.

④ DIO(Digital Input output)는 DIO channel, DIO port, DIO channel group으로 나뉘 pin에 대한 input output을 설정하도록 하는 Device Driver이다.

⑤ PORT는 MCU의 기본설정 이후에 Board상의 Pin을 어떤 목적으로 사용할지를 결정하고, Pin direction(Input /output), Pin의 Level 초기값을 설정하고 실행 중에 Pin direction 변경을 위한 API를 생성할 것인지 또는 실행 중에 Port Mode의 변경을 위한 API를 생성할 것인지를 설정하는 Device Driver이다.

이제 실무에서 직접적으로 사용할 수 있는 DIO Control을 위한 ARXML을 간단히 설계해서 실제 MCAL이 어떻게 사용되는지에 대해 알아보도록 하겠다.

DIO Control을 직접 제어하기 위해서는 시스템 설계에서 일반적으로 I/O Hardware Abstraction을 이용해 Client/Server Interface에서 Server Port만 사용해야 한다. 주의 사항은 I/O Hardware Abstraction은 Sender/ Receiver Interface를 사용할 수 없다. I/O Hardware Abstraction은 MCAL의 함수들을 직접(direct) 호출할 수 있도록 설계된다. I/O Hardware Abstraction은 EcuAbstractionSwComponent Type으로 설계되며, I/O Hardware Abstraction을 통해서 제어 가능한 MCAL에는 GPT Driver, SPI Driver, ICU Driver, PWM Driver, ADC Driver, DIO Driver, Port Driver 이상 총 7가지가 있다. 그리고 상위 SWC(Application SWC와 SensorAcuator SWC) 2개 중에서 EcuAbstraction SWC와 Port간 통신이 가능한 것은 Sensor Acuator SWC만 가능한 것이 AUTOSAR 표준이다. Sensor Acuator SWC에서는 Logic을 구현하고 EcuAbstraction SWC는 MCAL API의 직접 호출하도록 기능이 분리돼 있다.

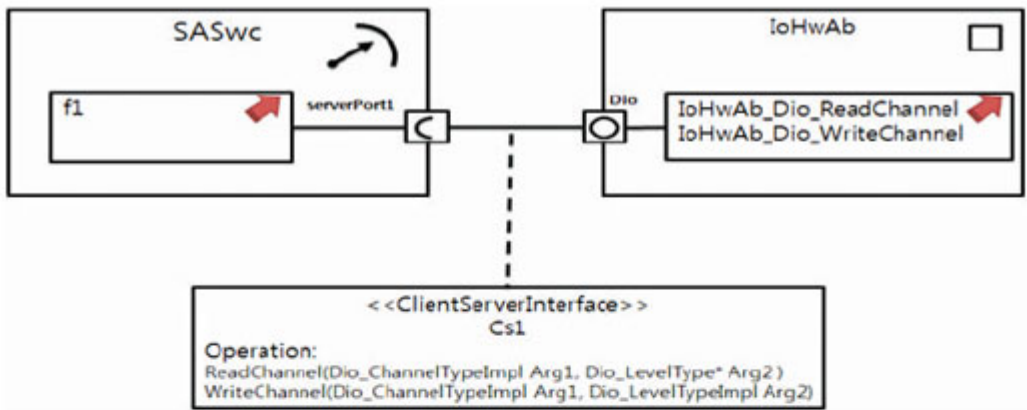


그림 8 | I/O Hardware Abstraction VFB 도식화

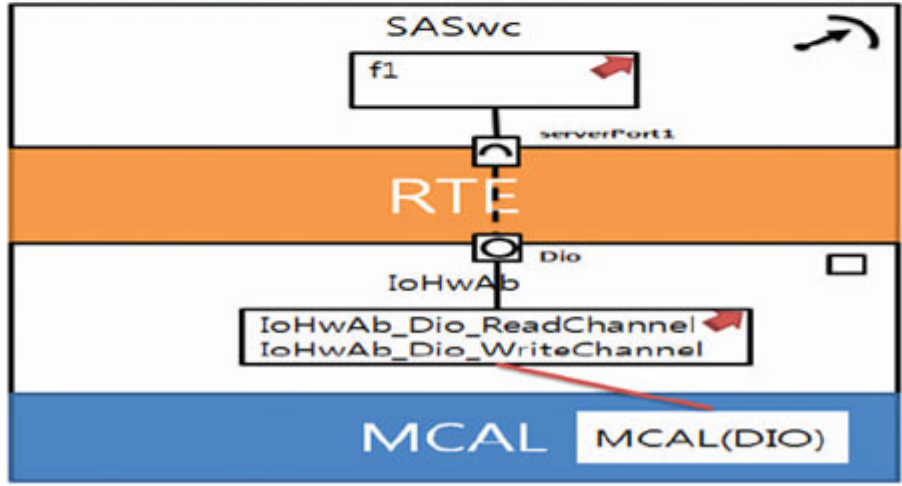


그림 9 | 시스템 설계 도식화

ARXML은 다음 그림과 같이 간단하게 DIO에 대한 값을 읽고 쓰기 위한 설계를 한 것이다.

위의 시스템 설계 ARXML은 Dio_Level Type Dio_ReadChannel(Dio_ChannelType ChannelId), void Dio_WriteChannel (Dio_ChannelType ChannelId, Dio_Level Type Level) 이렇게 2개의 DIO API만을 Control하기 위해서 설계를 한 것이다. 시스템 설계 과정에서 Client/Server의 Port(server Port1)에서의 Queue Length는 반드시 1이상으로 설정을 해야만 하는 필수 사항을 참고 하기 바란다.

설계된 ARXML은 그림과 같이 VFB로 도식화 될 수 있다.

위에서 설계한 내용을 수직관계로 도식화하면 그림 8, 9와 같다.

위의 시스템 설계 ARXML을 Generator해서 동작을 확인하기 위해서는 OS Task를 생성, Task에 RTE Event를 Mapping하고 MCAL의 기본인 MCU를 먼저 설정하고 어떤 용도로 Pin을 사용할지 정의하는 PORT를 설정한 이후에 DIO MCAL을 설정하도록 한다.

각각의 파일에서 생성되고 호출되는 함수들은 다음과 같다.

상위 App인 SASwc에서 Button의 상태를 체크해서 상태 값을 Led를 Control하기 위한 값으로 사용해서 Button의 상태에 따라서 Led가 On/Off하도록 하는 시스템 설계에 대한 전체적인 흐름에 대해서 살펴보았다. 앞에서도 언급했지만 시스템 설계단계의 Interface 과정에서 함수인자(Argument)는 MCAL 제어를 위해서 동일한 Data Type으로 정의돼야 한다는 것을 명심하기 바란다. 위의 예제에서는 Dio_LevelType, Dio_ChannelType이 사용됐다.

```
#include "Rte_SASwc.h"
#define STD_LOW 0
#define STD_HIGH 1
#define Button_69 69
#define Led_70 70
void f1(void)
{
    Dio_LevelType Arg2=STD_LOW;
    Rte_Call_SASwc_Dio_ReadChannel(Button_69,&Arg2);
    if(Arg2==STD_LOW){
        Rte_Call_SASwc_Dio_WriteChannel(Led_70,STD_LOW);
    }else{
        Rte_Call_SASwc_Dio_WriteChannel(Led_70,STD_HIGH);
    }
}
```

그림 10 | SASwc.c file 내용

```
#include "Rte.h"
#include "Os.h"
#include "Rte_Type.h"
#include "Rte_Main.h"
#include "Rte_Cbk.h"
#include "Rte_SASwc.h"
#include "Rte_IoHwAb.h"
FUNC(STD_ReturnType, RTE_CODE)Rte_Call_SASwc_Dio_ReadChannel(IN Dio_ChannelTypeImpl Arg1,OUT P2VAR(Dio_LevelType, AUTOMATIC, RTE_APPL_DATA) Arg2)
{
    Rte_GddQueueCS_IoHwAb_Dio_ReadChannel.Arg1 = Arg1;
    Rte_GddQueueCS_IoHwAb_Dio_ReadChannel.Arg2 = Arg2;
}
FUNC(STD_ReturnType, RTE_CODE)Rte_Call_SASwc_Dio_WriteChannel(IN Dio_ChannelTypeImpl Arg1,IN Dio_LevelTypeImpl Arg2)
{
    Rte_GddQueueCS_IoHwAb_Dio_WriteChannel.Arg1 = Arg1;
    Rte_GddQueueCS_IoHwAb_Dio_WriteChannel.Arg2 = Arg2;
}
```

그림 11 | Rte.c file 내용

```
#include "Rte_IoHwAb.h"
#include "Dio.h"
void IoHwAb_Dio_WriteChannel( Dio_ChannelType Arg1, Dio_LevelType Arg2)
{
    Dio_WriteChannel(Arg1,Arg2);
}
void IoHwAb_Dio_ReadChannel( Dio_ChannelType Arg1, Dio_LevelType* Arg2)
{
    *Arg2=Dio_ReadChannel(Arg1);
}
```

그림 12 | oHwAb.c file 내용

```
#include "Dio.h"
FUNC(Dio_LevelType, DIO_CODE) Dio_ReadChannel(CONST(Dio_ChannelType, AUTOMATIC)ChannelId)
{
    VAR(Dio_LevelType, AUTOMATIC) value = (Dio_LevelType) STD_LOW;
    value = Dio_LLDD_ReadChannel(ChannelId);
    return value;
}
FUNC(void, DIO_CODE) Dio_WriteChannel(CONST(Dio_ChannelType, AUTOMATIC)ChannelId,CONST(Dio_LevelType, AUTOMATIC)Level)
{
    Dio_LLDD_WriteChannel(ChannelId, Level);
}
```

그림 13 | Dio.c file 내용

SW-C에서는 Rtc_Call_<port name>_<operation>호출해 사용하는 것이 표준이다. 그러나, Rte.c에서는 Rtc_Call_<swc name>_<port name>_<operation> 형태로 사용된 것을 확인할 수 있을 것이다. 이유는 Rte.c내의 RTE API AUTOSAR 표준은 SWC name을 포함해야 되며, SWC내의 RTE API 호출의 표준은 SWC name이 삭제된 상태에서 호출돼야 하는 것이 AUTOSAR 표준이다.

따라서, AUTOSAR 표준을 준수해 SWC 내의 RTE API 호출은 SWC name을 넣지 않도록 주의가 필요하다.

마치며

이번 호에는 AUTOSAR MCAL에서 가장 기본이 되는 DIO(Digital Input output) Control의 간단한 예제를 작성해서 MCAL에서 제공된 API를 사용해서 전체적인 큰 그림의 사용법에 대해서 알아보았다. 사실 MCAL은 MCU Vendor가 제공하는 독자 Parameter에 대해서 파악하고 사용방법을 익혀야 한다. MCAL 전체를 본다면 여기에 소개된 내용으로는 턱없이 부족한 내용들이지만, 이번 호에서는 실무에 바로 적용할 수 있고 이해에 도움이 될 수 있는 내용들을 위주로 설명해봤다.

Communication Driver인 LIN, CAN, FlexRay는 다음호의 연재에서 좀 더 자세히 다룰 것이다.

다음에 연재할 내용은 AUTOSAR CAN (Controller Area Network)에 대한 내용이다. CAN은 차량용 Network를 위해 개발된 Serial 통신 프로토콜인데 CAN은 여러 가지 ECU(Electronic Control Unit)들을 병렬로 연결해 통신한다.

<저작권자(c)스마트앤컴퍼니. 무단전재-재배포금지>

100자평 쓰기 [로그인](#)

로그인후 입력하세요

등록

[Advertising / Media Partnership / Sponsoring](#) (/member/inquiry.asp?sel_type=ad)

회사소개 (<http://www.smartn.co.kr>) 개인정보취급방침 (</member/protect.asp>) 이메일주소 무단수집 거부 (</member/noemailcollect.asp>)
온라인 문의 (</member/inquiry.asp>) 정기구독 신청 (http://www.smartn.co.kr/book/book_detail.asp?p_no=B00033)
정기구독 주소변경 (/member/subs_edit.asp)

스마트앤컴퍼니(주) 대표이사 : 박성규 사업자등록번호 : 108-81-64739 통신판매업신고 : 2019-서울구로-2138호

서울특별시 구로구 디지털로34길 43, 607호(구로동, 코오롱사이언스밸리1차) [P: \(Phone\) 02-841-0017](#) [F: \(Fax\) 02-841-0584](#) ✉ webmaster@smartn.co.kr

© Smart & Company