



AUTOSAR 4.1.1의 최신 기술과 이해


Part 3: AUTOSAR RTE

2013년 11월호 지면기사 / 글 | 이 우 승 <twowin@infobank.net>

32&f_size=13)  [프린트](#)

:/articleView.asp%3Fidx=1232&t=AUTOSAR%204.1.1의%20최신%20기술과%20이해) 

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

3D1232&text=AUTOSAR%204.1.1의%20최신%20기술과%20이해) 
RTE&kakao_agent=sdk%2F1.37.2%20os%2Fjavascript%20lang%2Fko-KR%20device%2FWin32%20origin%2Fhttp%253A%252F%252Fwww.autoelectronics.co.kr)

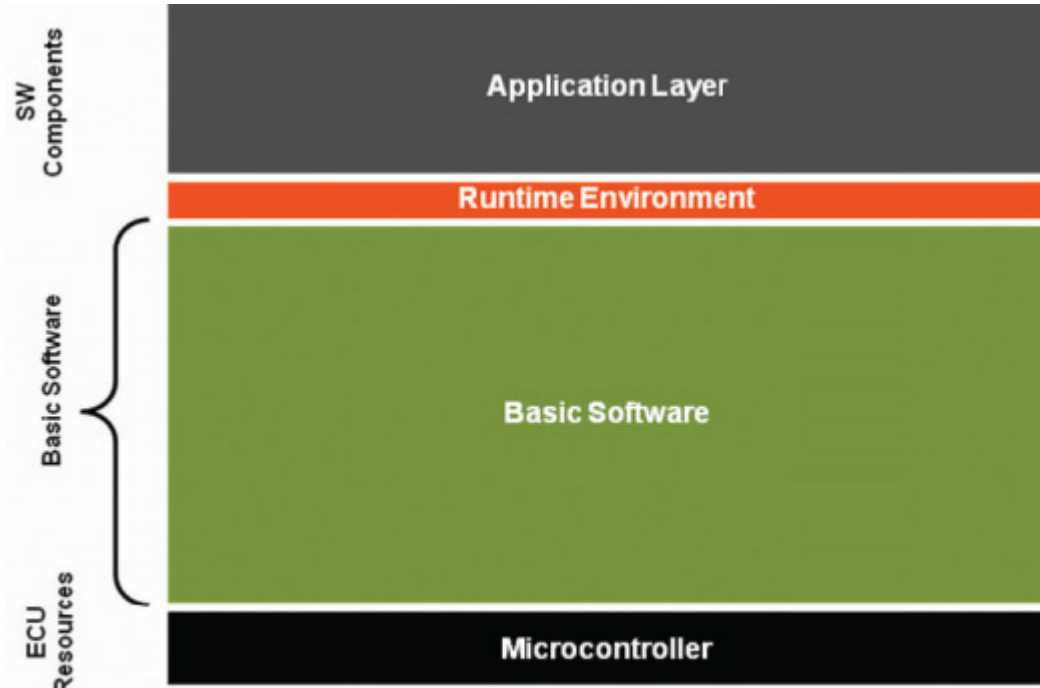
채승엽, 박대영, 이우승, 이현철 등 인포뱅크(Infobank) AUTOSAR 사업부 멤버가 AUTOSAR 최신 기술에 대한 아티클을 6회에 걸쳐 연재한다. 인포뱅크 사업부는 AUTOSAR 추세와 요구사항에 맞게 AUTOSAR Ethernet Solution, AUTOSAR 플랫폼 교육, SWC 개발 Tool과 같은 AUTOSAR 전반에 걸친 업무를 수행하고 있다. 세 번째 회는 'AUTOSAR RTE'다.

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

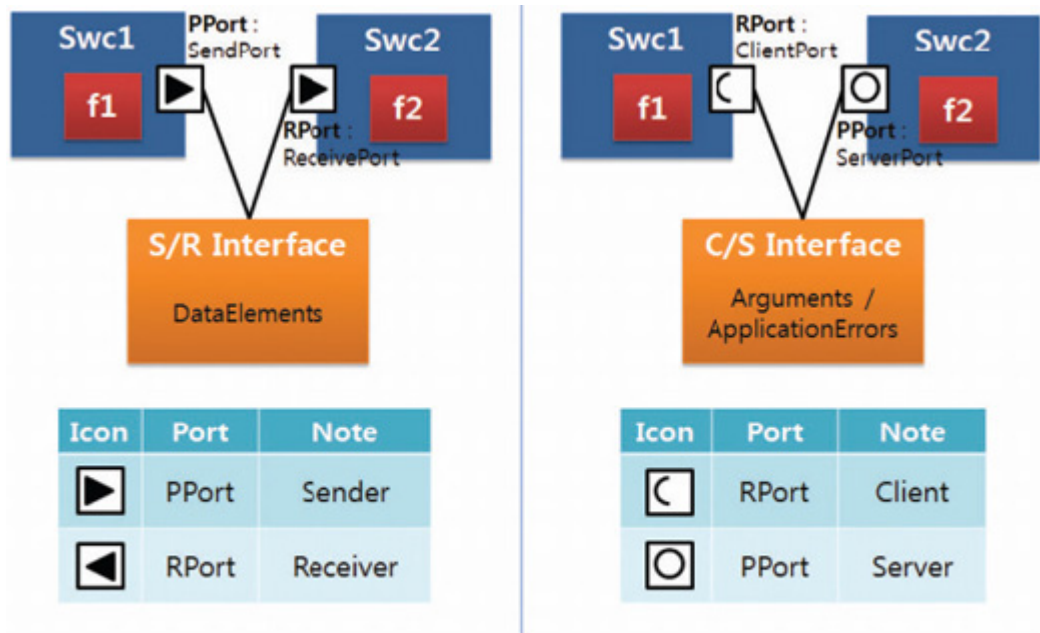


글 | 이 우 승 <twowin@infobank.net>

지난 호에 이어 이번에는 AUTOSAR RTE(Run-Time Environment, RTE)를 주제로 설명하고자 한다. RTE에 대해 간략히 소개하고 시스템 설계 단계에서 사용되는 RTE API 및 Event를 위주로 arxml을 작성하는 방법과 작성된 arxml이 코드로는 어떻게 생성되는지 설명할 예정이다. 참고로 본 연재에 소개될 소스는 개념 이해를 돕기 위한 요약된 소스로 실제 소스와는 다를 수 있으니 주의 바란다.



[그림 1] AUTOSAR Software Architecture · RTE Layer



[그림 2] S/R Interface와 C/S Interface

S/R interface와 C/S interface


SWC 간의 통신 역시 RTE를 통해 이뤄진다. SWC는 port를 가지고 있고 각 port를 연결하는 interface로는 Sender-Receiver(S/R) interface와 Client-Server(C/S) interface 등이 있다. S/R interface는 Sender가 receiver 측에 Data를 보내는 방식이고 C/S interface는 쉽게 말해 Client가 Server의 Operation 함수를 호출하는 방식이다. 그림 2는 S/R interface와 C/S interface 간의 차이를 도식화했다. 여기서 알 수 있듯이 두 interface 간의 큰 차이는 PPort와 RPort의 방향이 서로 다르다는 점이다. Arxml 작성 시 이 부분이 헷갈리지 않도록 유의할 필요가 있다.


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

(http://ssl.logger.co.kr/tracker_ad.tsp?u=37061&mode=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

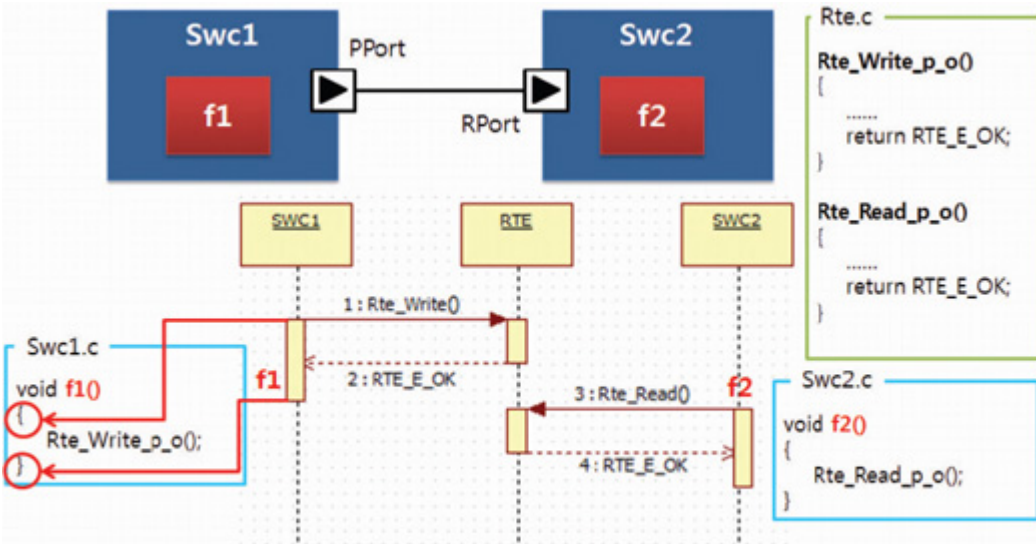
(/article/articleView.asp?idx=3470)

뤼츠 시스템 솔루션즈, 자동차 이더넷 테스터(ATE) 발표

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

(https://www.drivingthenation.com)

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

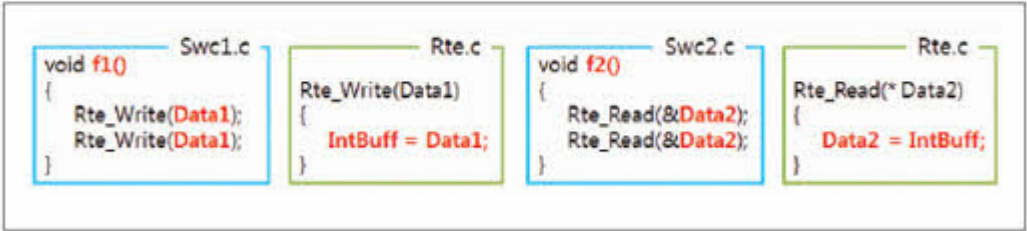


[그림 3] Sender-Receiver Interface

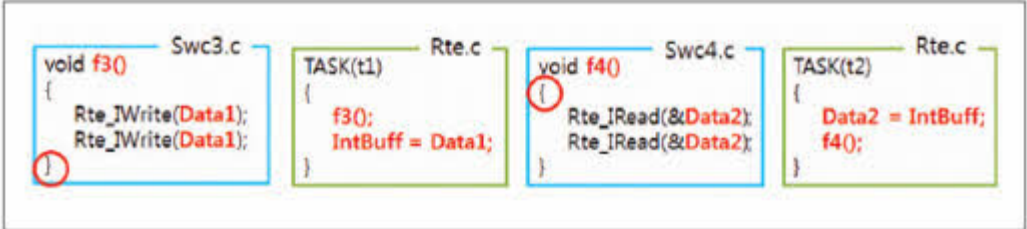
[표 1] S/R Interface를 사용하는 RTE API 비교

RTE API	DSP	DWA	Queue	Description
Rte_Write	O		FALSE	Queue를 사용하지 않고 Explicit하게 Data를 Write하는 API
Rte_IWrite		O	FALSE	Queue를 사용하지 않고 Implicit하게 Data를 Write하는 API
Rte_Send	O		TRUE	Queue를 사용하여 Explicit하게 Data elements를 Send하는 API

RTE API	DRP	DRA	Queue	Description
Rte_Read	O		FALSE	Queue를 사용하지 않고 Explicit하게 Data를 Read하는 API
Rte_IRead		O	FALSE	Queue를 사용하지 않고 Implicit하게 Data를 Read하는 API
Rte_Receive	O		TRUE	Queue를 사용하여 Explicit하게 Data를 Receive하는 API



[그림 4] Data Send Point / Data Receive Point



[그림 5] Data Write Access/Data Read Access

RTE API

Interface 구현을 위해서는 RTE API를 사용한다. 각각의 interface는 어떤 RTE API들이 사용되는지 알아보겠다.

S/R interface 동작은 그림 3에서와 같이 Swc1.c의 runnable인 f1()에서 Rte_Write를 사용하여 data를 전송하면 Swc2.c의 runnable인 f2()에서는 Rte_Read API를 사용해 data를 수신하는 방식이다. 자동 생성된 Rte.c 파일에서는 Rte_Write_p_o()/Rte_Read_p_o()와 같은 코드를 볼 수 있다. 여기서 <p>는 PortName이고 <o>는 DataPrototype으로 arxml 작성 시 설정한 name이 자동으로 들어가게 된다.

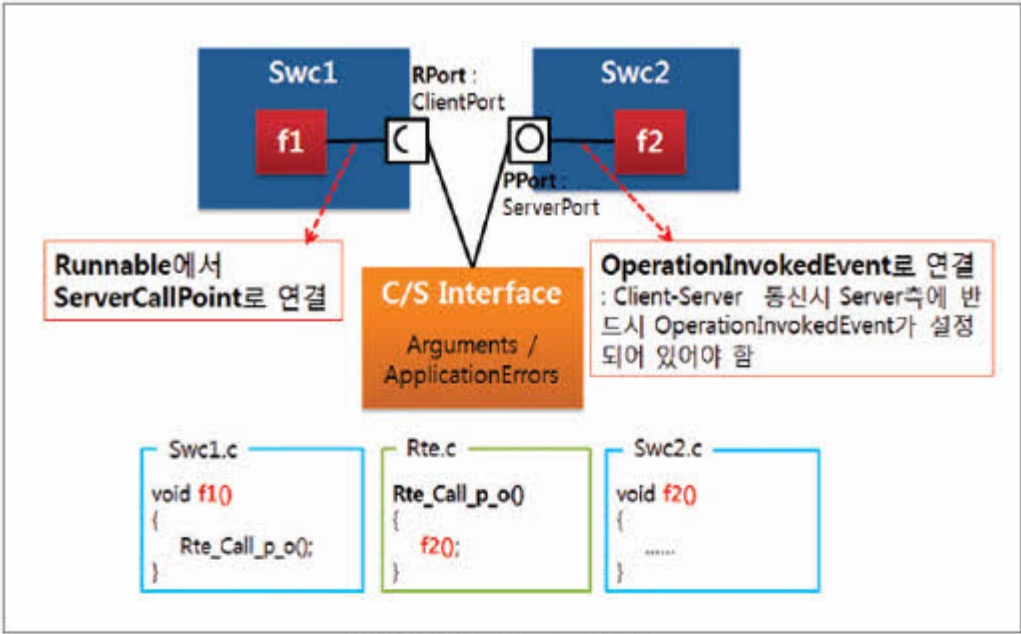
S/R interface에 주로 사용되는 RTE API에는 Rte_Write/Rte_Read, Rte_IWrite/ Rte_IRead, Rte_Send/Rte_Receive 등이 있다. Rte_Write, Rte_IWrite, Rte_Send가 PPort(Sender)에 해당하고 Rte_Read, Rte_IRead, Rte_Receive가 RPort(Receiver)에 해당한다.

각각의 API에 대해 좀 더 구체적으로 비교 설명하자면 표 1과 같다. Arxml 작성 시 SWC의 Runnable에서 Port와의 연결은 DSP/DWA/DRP/DRA로 지정할 수 있고 Port에서는 Queue 사용 여부를 결정할 수 있다.

- * DSP(DataSendPoint): API 호출 즉시 data를 전달 (Explicit)
- * DRP(DataReceivePoint): API 호출 즉시 data를 수신 (Explicit)
- * DWA(DataWriteAccess): Runnable 종료 후 data를 전달 (Implicit)
- * DRA(DataReadAccess): Runnable 시작 시 data를 수신 (Implicit)

여기서 Explicit 동작과 Implicit 동작의 차이를 설명하자면 Explicit하게 동작하는 Rte_Write와 Rte_Read의 경우 Rte_Write를 호출할 때마다 Data 값을 변경해 전송하면 내부 버퍼에 계속해서 변경된 값이 저장되고 Rte_Read가 호출될 때마다 변경된 Data도 값이 읽히게 된다. Implicit하게 동작하는 Rte_IWrite와 Rte_IRead의 경우 Rte_IWrite를 호출할 때마다 값을 변경해 Data를 여러 번 전송해도 Sender측 runnable이 종료된 후에 값을 전달하기 때문에 버퍼에는 가장 최근의 Data가 저장된다. 따라서 Rte_IRead는 여러 번 호출돼도 Receiver측 runnable 실행 시 한 번만 읽기 때문에 Data는 처음 읽었던 버퍼의 값이 저장된다. 그림 4와 그림 5의 코드를 보면 좀 더 이해가 갈 것이다.

이와 같이 API 마다 특성이 다르므로 개발자는 각 시스템의 성격에 맞게 원하는 API로 설계하면 된다.



[그림 6] Client-Server Interface

[표 2] RTE Event 종류

Abbreviation	Name	Description
T	TimingEvent	Software-Component description 내에 정의된 시간에 맞게 주기적으로 하나의 Runnable Entity를 작동시키는 EVENT.
DR	DataReceivedEvent	Sender-Receiver interface상에서 signal을 수신하고 처리하기 위한 Runnable Entity를 작동시키는 EVENT. (S/R Communication only)
DRE	DataReceiveErrorEvent	수신 쪽에서 data element의 error status를 수집하기 위해서 사용되는 Runnable Entity를 구동하기 위한 EVENT. (S/R Communication only)
DSC	DataSendCompletedEvent	Transmit acknowledgment Notification들을 수신하고 처리하기 위한 Runnable Entity를 작동시키는 EVENT. (Explicit S/R Communication only)
DWC	DataWriteCompletedEvent	Transmit acknowledgment Notification들을 수신하고 처리하기 위한 Runnable Entity를 작동시키는 EVENT. (Implicit S/R Communication only)
OI	OperationInvokedEvent	서버가 클라이언트로부터 Operation 실행을 위한 요청을 받았을 때 발생. (C/S Communication only)
ASCR	AsynchronousServer-CallReturnsEvent	비동기적인 Client-Server operation의 상태정보와 결과를 수집하기 위해 사용되는 Runnable Entity를 활성화시키는 EVENT. (C/S Communication only)
MS	SwcModeSwitchEvent	Mode변경의 결과로 Runnable Entity를 구동하기 위한 EVENT.
MSA	ModeSwitchedAckEvent	mode가 수신완료 또는 오류가 발생했을 경우 발생.
MME	SwcModeManagerErrorEvent	mode handling 동안 에러 발생시 RTE에 의해 발생.

C/S Interface의 주요 API로는 Rte_Call을 들 수 있다. Rte_Call은 Server의 Runnable을 바로 호출하는 API이다. Swc1.c의 runnable인 f1()에서 RTE API를 사용하는 것은 같으나 그림 6에서 보는 것과 같이 S/R Interface와는 다르게 Rte.c 파일의 Rte_Call_p_o() 안에서 Swc2.c의 runnable인 f2()를 바로 호출하는 것을 알 수 있다. 여기서 <p>는 PortName, <o>는 Operation이다.

Rte_Call API를 사용 시 Runnable에서 Port와의 연결은 ServerCallPoint(SCP)로 연결해야 한다. 그리고 또 한 가지 주의할 점은 Server측에 반드시 OperationInvokedEvent가 설정돼 있어야 한다는 것이다. 해당 Event는arxml에서 SWC Internal Behavior 안에서 설정할 수 있다.

RTE Event

Rte_Call과 같이 RTE API와 함께 RTE Event를 사용하는 경우가 많은데 지금부터는 RTE Event에 대해 설명하도록 하겠다. RTE Event는 RTE(Rte.c)에서 Runnable Entity를 activate하거나 wakeup하기 위해 사용한다. arxml에서 RTE Event를 설정하지 않을 수 있지만 그렇게 되면 Rte.c에서 봤을 때 API는 구현되어 있으나 해당 API가 실행되지는 않게 된다. 주요 RTE Event에 대한 설명은 표 2와 같다.

앞서 설명했던 S/R Inerface와 C/S Interface를 RTE API와 EVENT를 사용해 arxml로는 어떻게 작성하는지 예를 들어보겠다. Oxygen XML Editor를 사용했고 arxml의 내용이 길어 Editor의 outline을 보고 중요 내용인 SWC와 Interface 부분만 설명하도록 하겠다.



[그림 7] S/R Interface의 Rte_Write/Rte_Read



[그림 8] C/S Interface의 Rte_Call

S/R Interface 설계

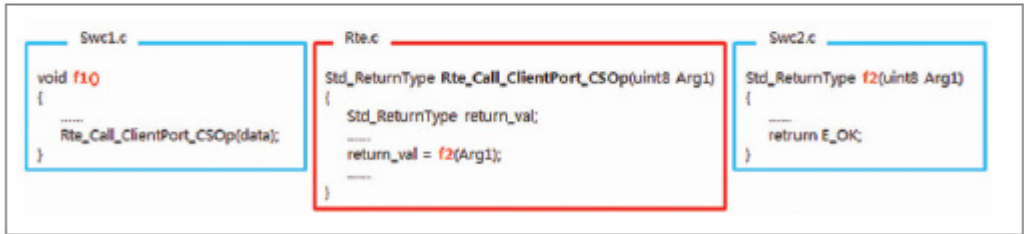
그림 7은 S/R Interface의 Rte_Write/Rte_Read를 설계한 내용이다. 먼저 ① Sender측인 Swc1의 Port를 보면 “PPort”로 정하고 PortName은 “SendPort”로 했고 ② Queue를 사용하지 않으므로 “Nonqueued Send Port”로 지정했으며 ③ Interface는 “SENDER-RECEIVER-INTERFACE”로 지정했다. Runnable은 ④ Name을 “f1”으로 정하고 ⑤ “DATA-SEND-POINTS”(DSP)로 설정했음을 알 수 있다. 다음으로 ⑥ Receiver측인 Swc2의 Port는 “RPort”로 정하고 PortName은 “Receive Port”로 했고 ⑦ Queue를 사용하지 않으므로 “Nonqueued SendPort”로 지정했으며 ⑧ Interface는 마찬가지로 “SENDER-RECEIVER-INTERFACE”로 지정했다. Runnable은 ⑨ Name을 “f2”로 정하고 ⑩ Port와의 연결은 “DATA-RECEIVE-POINT-BY-VALUES”(DRP)로 설정했다. Interface 부분의 설정을 보면 ⑪ “SENDER-RECEIVER-INTERFACE”로 지정했고 Name은 “SRIInterface”이다. ⑫ “VARIABLE-DATA-PROTOTYPE”의 Name은 “Data”이다. 해당 arxml로 generator를 거치면 Rte.c에는 Rte_Write_SendPort_Data, Rte_Read_ReceivePort_Data라는 API가 생성된다. Swc1.c의 runnable인 f1()에서 Rte_Write_SendPort_Data를 호출해 data를 전송하고 Swc2.c의 runnable인 f2()에서 Rte_Read_ReceivePort_Data를 호출해 수신된 data를 읽는다.

C/S Interface 설계

그림 8은 C/S Interface의 Rte_Call을 설계한 내용이다. 먼저 ① Client측인 Swc1의 Port를 보면 “RPort”로 지정했음을 알 수 있다. PortName은 “ClientPort”로 했고 ② Interface는 “CLIENT-SERVER-INTERFACE”로 지정했다. ③ Runnable Name은 “f1”이고 ④ Runnable과 RPort와의 연결은 “SERVER-CALL-POINTS”로 설정했음을 알 수 있다. 다음으로 ⑤ Server측인 Swc2를 보면, Port는 “PPort”로 정하고 PortName은 “ServerPort”로 했고 ⑥ Interface는 마찬가지로 “CLIENT-SERVER-INTERFACE”로 지정했다. ⑦ Runnable에서 Name은 “f2”로 했는데 여기서 중요한 점은 바로 Event 부분이다. Rte_Call에서는 Runnable과 PPort와의 연결 없이 Event가 설정된다. ⑧ Event를 보면 “OPERATION-INVOKED-EVENT”로 설정했음을 알 수 있다. Operation Invoked Event는 C/S 통신에서만 사용되는 RTE Event로 Rte_Call을 사용을 위해 반드시 설정해줘야 한다. Interface 부분을 보면 ⑩ “CLIENT-SERVER-INTERFACE”로 지정했고 Name은 “CSInterface”로 작성했다. 그리고 Rte_Call에서는 Operation과 Possible Errors를 설정해야 한다. ⑪ Operation Name은 “CSOp”로 했고 이 Operation Name이 Rte.c에서 API name으로 쓰이게 된다. Operation 작성 시 ⑫ Arguments를 추가하게 되는데 runnable에서 전달할 data가 된다. Arguments의 Name은 “Arg1”으로 했다. ⑬ Possible Errors는 runnable에서 return 값이라고 보면 된다. 이번 sample에서는 E_OK와 E_NOT_OK 두 가지로 정의했다. 해당 arxml로 generator를 거치면 Rte.c에는 Rte_Call_ClientPort_CSOp라는 API가 생성된다. 해당 API는 Swc1.c의 runnable인 f1()에서 호출하게 된다. Swc2.c의 runnable인 f2()는 runnable 작성 시 arguments와 possible errors를 모두 작성한 상태이기 때문에 return 값과 arguments를 모두 가지는 형태인 Std_ReturnType f2(uint8 Arg1) 형태로 선언된다. 여기서 Operation의 Arguments와 Possible Errors의 설정 여부에 따라 표 3과 같이 server측 runnable의 return 값과 parameter 유무가 결정된다. 해당 내용은 앞서 연재했던 “1. AUTOSAR의 실무 이해”에서도 언급했던 내용이다. 이렇게 생성된 f2()는 Rte.c의 Rte_Call_ClientPort_CSOp 내에서 바로 호출하는 형태가 된다. 그림 9를 보면 조금 더 이해가 잘 될 것이다. 앞의 arxml 설계에서 언급하지 않았던 RTE Event 중 몇 가지를 추가로 설명하자면 S/R Interface의 Sender측 runnable과 C/S Interface의 Client측 runnable에 Timing Event(T)를 설정할 수 있다. TimingEvent의 주기를 0.01로 설정할 경우 10ms마다 f1()을 주기적으로 실행할 수 있게 된다. arxml에서는 그림 9와 같이 작성할 수 있다.

[표 3] RTE Event 종류

Arguments	Possible Errors	Runnable
X	X	void f2()
X	O	Std_ReturnType f2()
O	O	Std_ReturnType f2(uint8 Arg1)



[그림 9] Rte_Call 예제



[그림 10] TimingEvent(T) 예제



[그림 11] DataReceivedEvent(DR) 예제

S/R Interface의 Receiver측에는 Data Received Event(DR)를 설정할 수 있다. Buffer에 write 작업이 완료될 때마다 Data Received Event가 발생돼 f2()를 실행하게 된다. 해당 RTE Event의 경우 S/R 통신에서만 사용할 수 있다. arxml에서는 그림 11과 같이 작성할 수 있다.

마치며

이번 호에는 AUTOSAR RTE API와 Event에 대해서 알아보았다. 사실 API와 Event들 외에도 RTE에 대한 내용들은 무수히 많다. RTE 전체를 본다면 여기에 소개된 내용으로는 턱없이 부족한 내용들이지만, 이번 호에서는 실무에 바로 적용할 수 있고 이해에 도움이 될 수 있는 내용들을 위주로 설명해보았다. 다음에 연재할 내용은 AUTOSAR MCAL (Microcontroller Abstraction Layer)에 대한 내용으로, MCAL은 차량용 반도체 회사에서 반드시 지원해야 하는 AUTOSAR용 Device Driver라고 할 수 있다. 다음 연재에는 MCAL에 대해 자세히 설명하도록 하겠다. AE

<저작권자(c)스마트엔컴퍼니. 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)

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

서울특별시 구로구 디지털로34길 43, 607호(구로동, 코오롱싸이언스밸리1차) P: (Phone) 02-841-0017 F: (Fax) 02-841-0584 ✉ webmaster@smartn.co.kr

© Smart & Company