SAE J1939 개발 도구

IXXAT J1939 Protocol Stack

SAE J1939 표준은 트럭, 건설 또는 농업용 차량과 기계들 같은 중장비 차량에서의 전자 제어 장치(ECU)들 간의 데이터 통신을 바탕으로 하는 CAN 을 정의합니다. 1998년에 최초로 발표되었지만, SAE J1939 표준은 최근 몇 년간 인기가 늘어나고 있습니다. 현재 디젤 엔진, 트랜스미션, 브레이크등을 위한 ECU 공급업체들은 그들의 시스템을 J1939를 기반으로 하는 통신으로 변환하지 않을 수 없게 되었습니다.

SAE J1939 프로토콜 스택에 대한 수요

J1939 프로토콜 스택을 ECU 같은 디바이스에 적용 시킬때, 가장 먼저 고려할 것은 이용할 수 있는 자원입니다. 오늘날의 ECU는 다소 제한된 자원을 가진 8비트 플랫폼부터 풍부한 메모리와 고성능 CPU 그리고 정교한 실시간 운영 시스템을 실행하는 32비트 시스템에 이르는 마이크로컨트롤러들에 장착되어 있습니다.

사용된 마이크로컨트롤러에 상관없이, 통신 스택은 항상 처리 시간 또는 메모리와 관련해서 자원을 그다지 많이 소모해서는 안 되는 액세서리로 간주됩니다. 특히 저가형 시스템의 경우 메모리 foot-print 가 중요합니다.

J1939 프로토콜의 구현에서 또 다른 중요한 측면은 애플리케이션의 실제 통신 요건들입니다. 어떤 애플리케이션들은 단지 최대 8 바이트 데이터 길이의 메시지 몇 개와 고정된 디바이스 어드레스 (네트워크 제어 축소)만으로 충족될 수 있습니다. 그러나 어떤 애플리케이션들은 동적 디바이스 주소 할당, 대량 데이터 패킷 (멀티-패킷) 전송과 SAE J1939/73에 따른 종합적인 진단 기능들을 필요로 하기도 합니다. 따라서 J1939 프로토콜 스택은 애플리케이션의 실제 요건에 맞춰 쉽게 응용(adaptable) 가능하고 매우 가변적(scalable)이어야 합니다.

㈜임베디드시스템코리아

애플리케이션과 프로토콜 소프트웨어의 인터페이스

우선, J1939 프로토콜 스택은 J1939메시지의 전송과 수신을 위한 기능을 제공해야 합니다. 애플리케이션과 프로토콜 소프트웨어의 인터페이스에는 일반적으로 두 가지 다른 방법들이 널리 사용됩니다:

- **Function Interface**: 이 인터페이스 유형에서는 애플리케이션과 프로토콜 소프트웨어가 해당 함수 호출에 의해 연결됩니다.
- **Data Interface**: 이 인터페이스 유형에서는 애플리케이션과 프로토콜 소프트웨어가 공유 메모리를 거쳐 직접 데이터 값(signals)을 교환합니다.

함수 인터페이스를 통해 데이터를 전송하기 위해서, 애플리케이션이 일체의 메시지를 전송하도록 전송 함수를 사용합니다. 수신된 메시지는 프로토콜 소프트웨어에서 애플리케이션으로 callback 함수를 통해 지연 없이 전송되거나 애플리케이션이 메시지를 읽을 수 있는, ring 버퍼에 임시 저장됩니다. 함수 인터페이스는 애플리케이션과 프로토콜 소프트웨어 사이의 엄격한 연결을 제공합니다.
데이터 인터페이스를 통한 데이터 전송은 독립적으로 애플리케이션과 프로토콜 소프트웨어에 의해 접속될 수 있는 공유 메모리를 매개로 실행됩니다. 그러므로 애플리케이션과 통신 소프트웨어는 가능한 최대 범위까지 결합이 해제(decouple)됩니다.
이 애플리케이션은 자체의 고유 처리 사이클을 실행할 수 있으며 데이터 인터페이스를 사용하여 비동기적으로 필요할 때만 데이터를 전송하고 수신합니다. 이 방법으로 메시지 데이터의 assembling 과 disassembling은 통신 스택의 작업이 됩니다. 데이터 인터페이스를 이용하여 애플리케이션은 직접 애플리케이션 데이터를 접속합니다.

표준(standard) 메시지와 특정(proprietary) 메시지

J1939 표준은 일반 메시지 포맷에서 지정된 message catalogue를 정의합니다. 이것은 프로토콜 스택에 의해 사용된 일반 메시지들을 위한 특정 메시지 pack() 과 unpack() 함수들을 위한 기점입니다.
또한, J1939는 특정 공급업체의 특정 메시지들간 교환을 허용합니다. 특정 메시지들의 disassembly와 assembly를 위해, 프로토콜 소프트웨어는 특정 메시지로부터 애플리케이션 데이터를 각각 demapping, mapping하기 위한 규칙들을 알아야 합니다. 따라서 프로토콜 소프트웨어는 특정 메시지와 관련되어 구성되어야 합니다.

Scalable J1939 프로토콜 스택

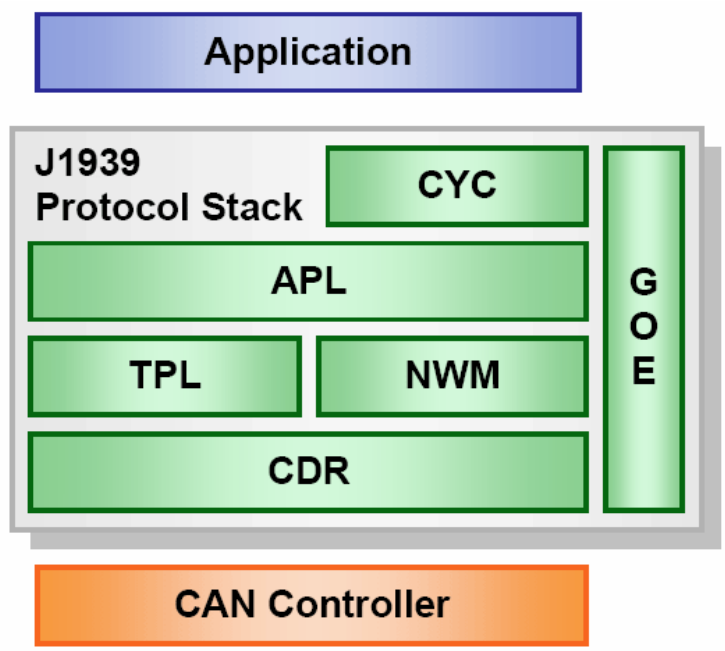
IXXAT의 J1939 프로토콜 스택은 애플리케이션의 실제 통신 요건에 맞추기 위한 다양한 옵션들을 제공하는 매우 가변적인 솔루션으로, 자원 소비는 줄이면서 간편한 소프트웨어 통합을 제공합니다.

J1939 프로토콜 소프트웨어는 다음 모듈들을 포함합니다.

- CAN 컨트롤러를 액세스 하기 위한 CAN 드라이버 (CDR)
- 세그먼트된 메시지의 전송을 위한 전송 계층 (TPL)
- 별도의 mapping/demapping 매크로들을 포함한 함수 지향 프로그래밍 인터페이스를 위한 애플리케이션 계층 (APL)
- 데이터 인터페이스를 제공하고 주기적으로 전송된 메시지의 처리와 감독을 위한 순환 모듈
- 디바이스 주소 청구 처리를 위한 네트워크 관리 (NWM)
- 운영 시스템 추상 계층 제공을 위한 일반적 운영 시스템 환경

IXXAT는 두 가지 버전의 J1939 프로토콜 소프트웨어를 제공합니다:

- 충분한데이터 메모리 (2kByte)가 있는 타겟 시스템을 위한 **standard version**.
- 제한된 데이터 메모리 (최소 200 bytes)의 시스템을 위한 **micro version**.



Block diagram of the J1939 protocol software

Standard version 은 실행되는 동안 function 인터페이스를 통한 SAE J1939 소프트웨어의 역동적인 구성이 가능합니다. 따라서, 예를 들어 수신 메시지들을 위한 필터 규칙들이 실행되는 동안에 수정될 수 있습니다. 또한 이 버전은 한 개 이상의 software instance (CAN 채널)을 지원하며 이 목적을 위해 특별히 제공된 추상 계층(abstraction layer)에 필요한 실시간 운영 시스템 환경에서의 사용을 위해 설계되었습니다. 이 표준 버전은 또한 운영 시스템 없이도 시스템에서 사용될 수 있습니다.

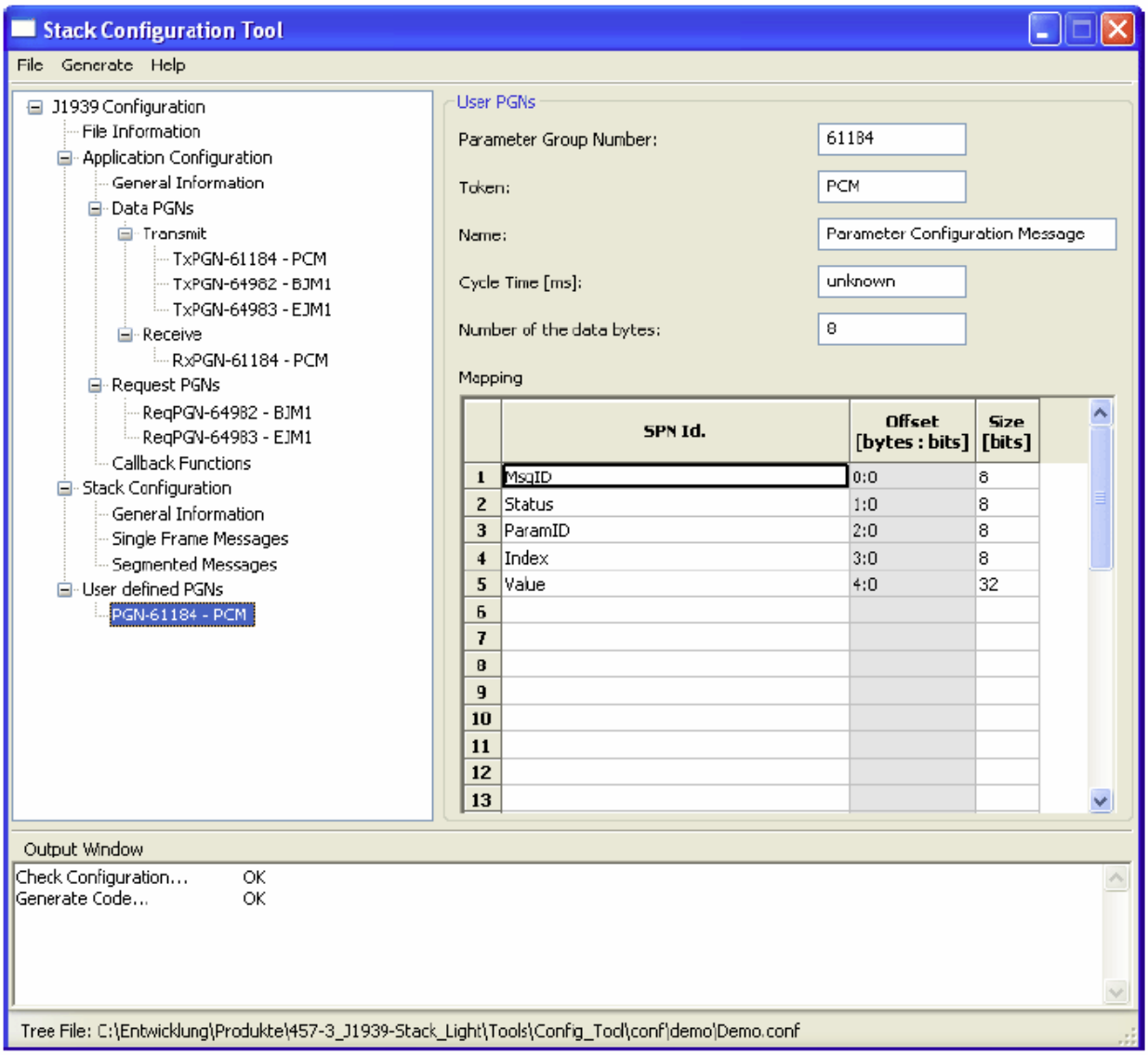
Micro version 은 제한된 성능과 메모리 자원을 가진 8/16-bit CPU 시스템에서의 사용에 최적화되어 있습니다. 이 경우 프로토콜 소프트웨어는 제한된 자원의 한정된 사용을 위해 컴파일하는 동안 구성되어야 합니다.
실행되는 동안 매개 변수들의 변경은 불가능합니다. 원하는 J1939 구성을 나타내는 C-file은 IXXAT J1939 구성 툴에 의해 자동으로 생성됩니다. 끝으로 일체의 구성이 standard version에서 사용된 동적 구성과 비교했을 때 RAM 요건이 상당히 축소된 코드에 통합됩니다.
양쪽 소프트웨어 버전들은 J1939 메시지의 전송과 수신을 위해 비슷한 function 인터페이스를 갖습니다. Standard 버전에서는, function 인터페이스가 실시간 실행 환경에서 한 개 이상의 작업에 의해 사용될 수 있는 방식으로 설계됩니다. 따라서 Standard 버전은 “thread-safe”로 언급될 수 있습니다.

메시지-지향 인터페이스를 제공하는 function interface 외에, IXXAT J1939 스택에서는 신호-지향 데이터 인터페이스가 추가로 제공됩니다. 이것은 메시지를 주기적으로 모니터링하고 전송하는 것 이외에도 프로토콜 스택과 애플리케이션 사이에 데이터 인터페이스를 제공하는 이른바 cycle 모듈에 의해 이루어집니다.
프로토콜 소프트웨어는 전송 프로토콜을 처리하고 time-out을 모니터하기 위해 주기적으로 호출되어야 합니다. 실시간 실행 시스템 (super loop)이 없는 시스템에서, 이것은 대개 주기적으로 처리되는 메인 프로그램에 있는 process() function을 호출하여 이루어집니다.

실시간 실행에 의해 제어되는 시스템에서는, process() 함수 호출을 위하여 별도의 순환 작업이 설치되어야 합니다.

J1939 프로토콜 소프트웨어 구성 툴

J1939 프로토콜 스택에서 제공되는 시각적 구성이 뛰어난 J1939 구성 툴은 micro와 standard version의 구성과 개별 맞춤 작업을 매우 수월하게 만듭니다.



J1939 Configuration Tool – Definition of user specific messages

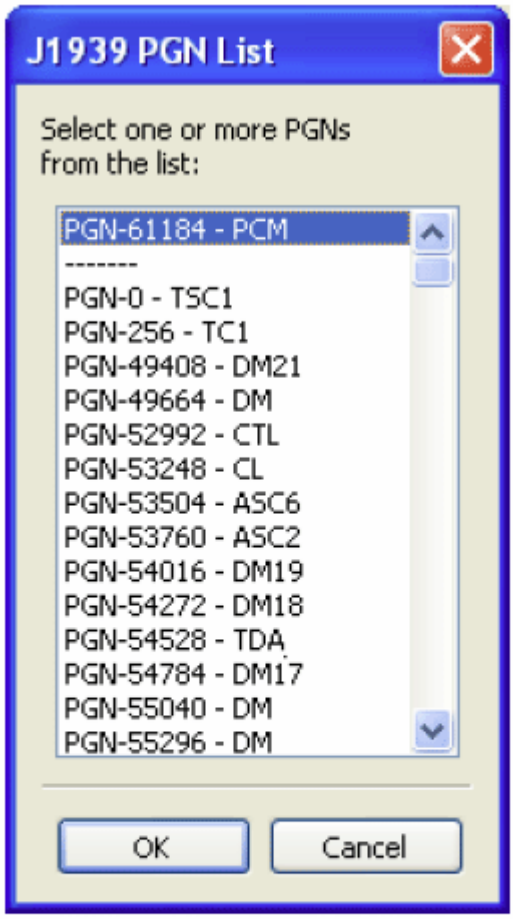
디바이스 이름과 ring 버퍼의 크기 같은 일반 매개변수들 설정 외에도, J1939 구성 툴은 또한 pre-defined message catalogue 에서의 일반 메시지 선택이 가능합니다. 프로토콜 소프트웨어에서 필터(filter)와 분배(distributor) 기능들이 구성되고 액세스 매크로들이 C 파일 형태로 생성되어 이러한 설정들을 바탕으로 하는 소프트웨어 프로젝트에 직접 통합됩니다. 이러한 액세스 매크로들은 J1939 메시지들과 처리 파라매터 또는 수신과 전송을 위해 애플리케이션에서 사용된 변수들 사이에 mapping 규칙을 정의합니다.

일반 메시지들의 구성 외에, J1939 구성 툴은 또한 독점 메시지들의 정의도 가능합니다.

J1939 구성 툴의 가장 중요한 특징은 하나의 툴에 의해 모든 구성이 생성된다는 사실이며 따라서 모든 파라매터들의 일관성이 보장됩니다.

㈜임베디드시스템코리아

그러므로 FIFO 메모리 사이즈의 부정확 같은 문제들을 피할 수 있습니다. 더 나아가 J1939 구성 툴은 J1939 애플리케이션을 위한 완벽한 C-code framework를 생성하는 옵션을 제공합니다. 이 기능은 특히 J1939 를 이용한 "첫 단계" 에서 유용한데, 이것이 J1939 통신으로의 매우 빠른 엔트리를 지원하기 때문입니다.



J1939 Configuration Tool – Standard message catalogue based on Parameter Group Numbers PGNs

예제 애플리케이션

다음의 코드 라인들은 J1939 프로토콜 소프트웨어의 초기화와 작동을 보여줍니다. 이 예에서, 수신된 파라매터는 데이터 인터페이스를 통해 버퍼에서 읽혀집니다.

이 메시지는 필수 파라매터(엔진 오일 압력, SPN100)가 포함된 수신 메시지로써 J1939 구성 툴을 이용하여 설정되어 있습니다.

프로토콜 소프트웨어의 구성은 J1939 구성 툴을 통해 수행되었으므로, function APL_Init() 호출에 의해 자동적으로 모든 구성 파라매터들이 참작됩니다. 단 milliseconds의 local time은 프로토콜 소프트웨어로 (function time() 을 통해) 패스되어야 합니다

```
void main(void)
{
   // Initialize the J1939
   // protocol software
   APL_Init(time());

   while(1)
   {
      // Keep the J1939 protocol
      // software running
      APL_Main(time());
      do_other_tasks();
   }
}
```

Cycle 모듈은 구성된 메시지의 수신을 모니터하고 버퍼에서 데이터를 저장하는데, 이것은 J1939 구성 툴에 의해 생성되었던 것입니다. 실제 파라매터는 J1939 구성 툴에 의해 생성되었던 매크로 UNPACK_EFL_P1_SPN100() 를 통한 애플리케이션에 의해 읽혀질 수 있습니다.

```
// read SPN100 (engine oil pressure)
// from PGN65263 (EFL/P1)
val = UNPACK_EFL_P1_SPN100();
```