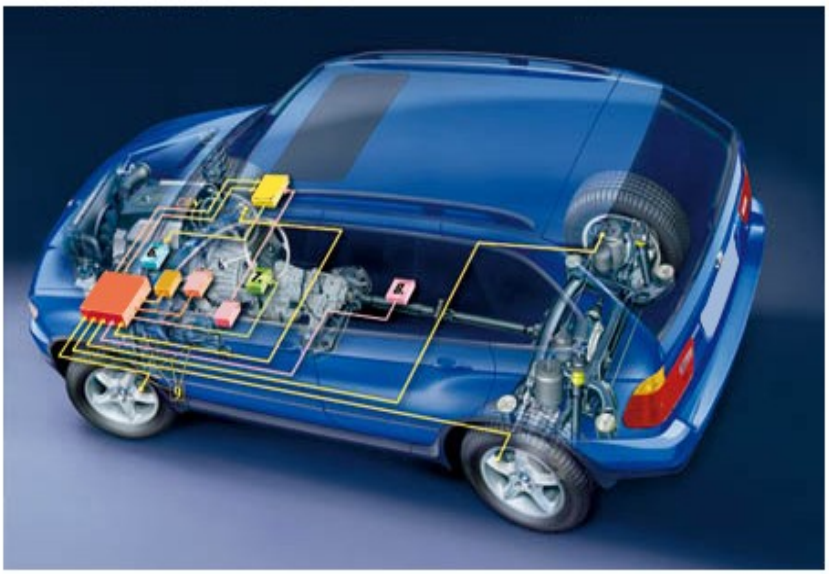
CCP/XCP 프로토콜 입문

배경



CCP (CAN Calibration Protocol) 는, 이름에서 알 수 있듯이, 전자 제어 유닛 (ECU) 에서 데이터를 수집하고 교정하기 위한 프로토콜입니다.

이 프로토콜은 일찍이 ASAP (*Arbeitskreis zur Standardisierung von Applikationssystemen)* 로 알려졌던, ASAM (*Association for Standarisation of Automation- and Measuring Systems*) 에 의해 정의되었습니다.
이것은 Audi, BMW, VW 같은 여러 주요 자동차 제조업체들로 구성된 국제적인 기구입니다. 과거에는 서로 다른 방법의 기술들이 ECU 하드웨어와 소프트웨어의 개발, 교정, 생산과 서비스에 사용되어 왔었습니다. CCP의 목표는 모든 단계의 ECU 개발을 위한 공통 툴을 만들고 다른 종류의 하드웨어 그리고 소프트웨어들과도 호환 되도록 하는 것입니다.

ASAM 그룹은 많은 표준들을 정의합니다. CCP와 XCP 표준은 subsection AE (Automotive Electronics) 에서 찾아보실 수 있으며 일명 **AE MCD 1** 이라 불리는 것으로 분류됩니다. CCP 규격의 현재 버전은 2.1 이며, 이것은 1999년 2월에 발표된 것입니다.

사용 분야

가장 흔히 사용되는 영역은 CAN이 자주 사용되는, 자동차 산업이지만, 그 이외의 다른 분야에서도 CAN 은 사용됩니다. 사용이 가능한 영역들은 다음과 같습니다.

- ECU 개발
- ECU의 기능과 환경 테스트를 위한 시스템
- 연소 기관, 변속 장치 또는 실내 온도 조절기를 위한 테스트 벤치.
- 사전 제작 차량에서의 측정과 교정
- 자동차 산업 이외의 일반적인 CAN 애플리케이션

요약

CCP 는 CAN 2.0B (11- or 29-bit CAN id) 의 애플리케이션 계층입니다. 이 프로토콜은 OSI 모델에 따른 최상 계층 (layer 7) 으로, 이것은 프로토콜이 비트와 바이트가 어떻게 생성되는지를 서술하는 것이 아니라 CAN 2.0B protocol physical, data link 와 네트워크 계층이 어떻게 사용되는지를 서술하는 것입니다.
CCP 는 다음 기능들을 지원합니다:

- ECU 메모리에 읽기(Read) 와 쓰기(Write).
- ECU로부터 동기적으로 순환 데이터 취득.
- CAN 버스에서 다중 노드 처리.
- 플래시 프로그래밍.
- 플러그 앤 플레이.
- 자원 보호(데이터 취득과 교정).

CCP의 세부사항

CCP 는 명령을 슬래이브 노드로 보내어 CCP 마스터가 통신을 시작하는 master/slave 애플리케이션에 기초합니다. CAN 버스에 연결된 슬래이브 노드는 여러 개가 될 수 있습니다. CCP는 교정을 위한 간단한 메모리 처리와 데이터 획득과 관련하여 일반 명령을 사용합니다. 이러한 두 개의 자원들은 독립적이며 따라서 동시에 사용될 수 있습니다.

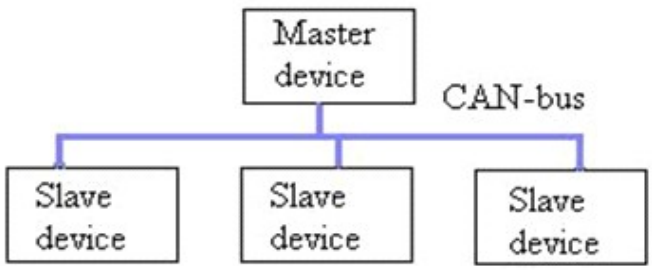


그림1: CCP 버스 연결.

CCP는 고성능 ECU와 소형 8비트 마이크로컨트롤러 모두의 제약과 요구를 처리하도록 설계되었습니다. ECU에는 추가 하드웨어가 전혀 연결되지 않습니다. CCP 드라이버는 소프트웨어에서 완벽하게 구현됩니다. CCP의 간소한 구현은 오직 이용 가능한 램, 롬, 그리고 실행을 위한 CPU 시간의 작은 부분만이 필요합니다. 간소한 구현은 단지 두 개의 CAN 메시지 식별자만을 필요로 하며, 이것은 통상적인 트래픽을 방해하지 않는 낮은 순위로 설정될 수 있습니다. CCP가 일반 PC에서 사용되어야 한다면, 마이크로컨트롤러에서 사용되는 것과 마찬가지로 간단하고 저렴한 비용의 CAN 인터페이스가 사용될 수 있습니다.

일반적인 명령

CCP 는 슬래이브 노드에서 서로 다른 기능들을 수행하기 위해, 노드에 한정된 것이 아닌, 일반 명령들을 사용합니다. 명령들이 일반적이기 때문에 모든 노드는 개별적인 위치 어드레스를 갖고 있어야 합니다. 마스터와 슬래이브 간의 논리적 연결은 어떤 명령이든 전송되기 전에 설정되어야 합니다. 이 연결은 마스터가 다른 슬래이브 노드와 연결하기로 결정할 때까지 또는 마스터가 disconnect 명령을 보낼 때까지 지속됩니다. 연결 후에는 마스터가 마스터와 슬래이브 사이의 모든 통신을 제어합니다. 마스터로부터의 모든 메시지는 데이터 또는 에러 코드를 포함하고 있는 슬래이브로 부터의 응답 메시지가 이어집니다.

CCP-특정 CAN 메시지

CCP는 CAN 2.0B 프로토콜을 기초로 합니다. 모든 메시지들은 8 바이트 길이입니다.
CAN 메시지의 단 두 종류만을 필요로 합니다, CRO 와 DTO, 각 방향으로 하나씩. CRO (Command Receive Object) 메시지는 마스터로부터 슬래이브로 전송되며 제어 명령을 포함하고 있습니다. DTO (Data Transmission Object) 메시지는 슬래이브에서 마스터로 전송됩니다. 슬래이브가 CRO 메시지를 수신했을 때 이것은 주어진 지시를 수행하고 그런 다음 CRM (Command Return Message) 를 포함하고 있는 DTO 메시지로 응답합니다. CRM 코드는 해당 제어 명령이 계획대로 수행되었는지 아닌지를 마스터에 알려줍니다.

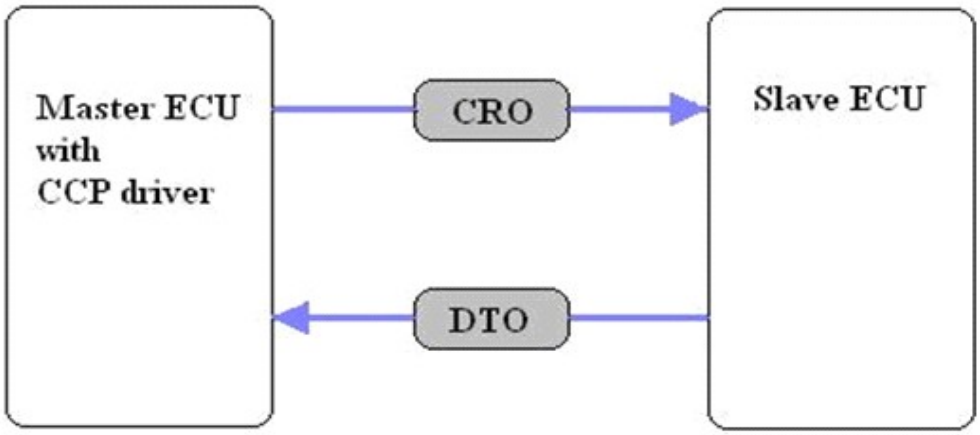


그림 2: CRO 와 DTO 메시지

CRO와 DTO 메시지에 사용되는 CAN 식별자들은 마스터를 구성하는데 사용되는 configuration file ("A2L file", defined by the **ASAM MCD 2MC/ASAP2** standard) 에 의해 결정됩니다. 이 구성 파일은 데이터 취득과 교정에 유용한 슬래이브 메모리 조직에 관한 정보도 포함할 수 있습니다. CAN 식별자가 메시지의 우선순위를 결정하므로 이것은 버스에서 통상적인 트래픽을 방해하지 않는 방식으로 선택 됩니다. CCP는 일반 데이터 전송에서 사용되는 바이트 순서 (Motorola 또는 인텔)를 결정하지 않습니다. 예외는 있는데 마스터와 슬래이브간의 연결 시 사용되는 station 주소의 바이트 순서는 Intel (LSB 우선)이 되어야 합니다.

CRO 메시지의 서술

CRO 메시지는 마스터에서 슬래이브로 전송되며 지시 사항들을 포함합니다. 첫 번째 바이트는 command code (CMD) 로 메시지의 목적을 서술합니다. 두 번째 바이트는 command counter (CTR) 로 통신의 추적을 유지하기 위해 사용됩니다. 또한 명령 카운터는 슬래이브에서 DTO 메시지의 리턴시 전송될 것입니다. 바이트 2-7은 명령 코드에 따라 데이터 파라매터들을 위해 유보됩니다. 메시지는 항상 8 바이트 길이이며 정의되지 않은 바이트들은 "개의치 않는 것"으로 간주됩니다.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CMD	CTR	Data	Data	Data	Data	Data	Data

그림 3: CRO 메시지의 구조.

DTO 메시지의 서술

DTO 메시지는 수신된 CRO 메시지를 받을 때 슬래이브에 의해 전송되며 데이터 취득을 위해서도 사용됩니다. 메시지에서의 첫 번째 바이트는 PID (Packet ID) 라 불립니다. PID의 값은 메시지 유형을 서술합니다. 메시지에는 세 가지 유형들이 있습니다:

- 0xFF, command return message (CRM), DTO가 CRO 메시지 수신으로 전송되는 경우
- 0xFE, event message, DTO가 오류 복구 또는 다른 서비스를 불러내기 위하여 내부 슬래이브 상태 변경을 보고하는 경우
- 0 - 0xFD, data acquisition message (DAQ). 이 PID는 나중에 서술되는, ODT (Object Descriptor Table)의 값을 가집니다

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CMD	ERR	CTR	Data	Data	Data	Data	Data

그림 4: DTO 메시지의 구조

데이터 취득 (DAQ)

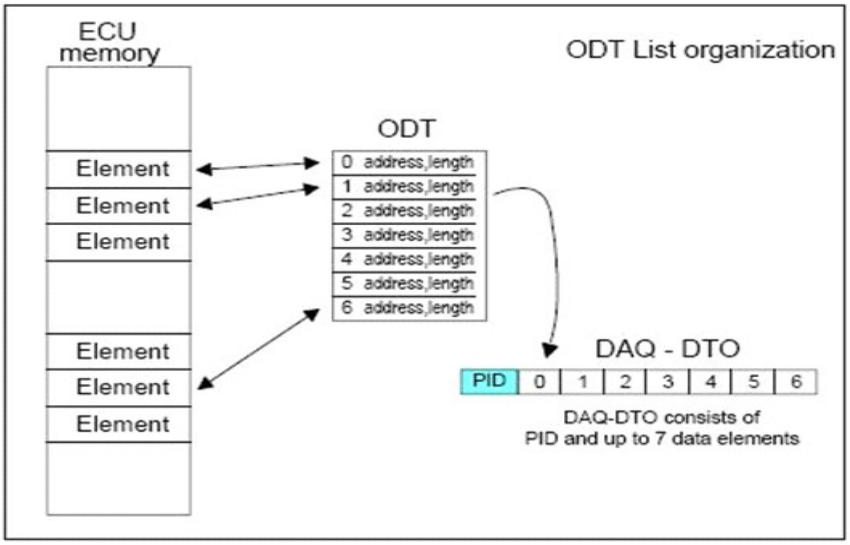


그림 5: ODT 리스트의 구조

마스터 디바이스가 초기화하여 슬래이브 디바이스로부터 데이터 취득을 시작할 수 있습니다. 데이터는 특수 DAQ-DTO와 함께 슬래이브에서 전송됩니다. 데이터 바이트는 수많은 ODT 리스트로 구성되어 있는 DAQ 리스트에서 조직됩니다. ODT 리스트는 데이터가 저장되는 ECU에 메모리 어드레스에 대한 최대 7개의 포인터들을 포함합니다. 메모리 어드레스에 대한 포인터 외에, ODT-list는 주소 확장과 보내질 바이트 수를 포함할 수 있습니다. 모든 슬래이브 디바이스들은 1 바이트 이상의 데이터 요소를 처리하지 않으며 마스터가 데이터를 단일 바이트로 쪼개어 이 작업을 해결할 수 있습니다.

DAQ-DTO 는 ODT 리스트의 메모리 포인터가 가리키는 데이터 요소들과 PID로 구성됩니다. PID 번호 (일반적으로 ODT 리스트와 동일) 는 0에서 253 사이의 값을 가지며 이는 254개의 ODT 리스트만이 동시에 있을 수 있다는 것을 의미합니다.

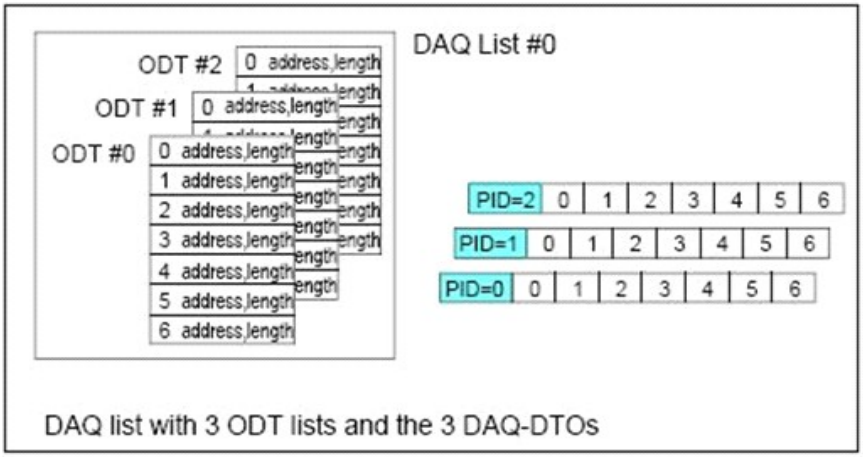


그림 6: DAQ 리스트의 구조

CCP 규격은 여러 개의 DAQ 리스트를 허용하며, 이것은 동시에 활성화될 수 있습니다. DAQ 리스트의 전송은 START_STOP 명령을 통해 마스터에 의해 시작됩니다. ODT 리스트의 데이터 바이트는 슬래이브 디바이스에서 샘플화되어 DAQ-DTO의 CAN 버스로 보내집니다. 새로운 START_STOP 명령이 진행중인 DAQ 사이클이 완료되기 전에 슬래이브에 의해 수신된다면, 두 가지 방식으로 반응합니다. 새로운 DAQ 명령이 시작되고 진행 중인 것이 종결되거나 진행중인 사이클이 완료되고 새로운 것은 무시됩니다. 양쪽 방식에는 장점과 단점이 있으며 CCP 규격은 어떤 것을 선택할지 정하지 않습니다.

명령(Command)

아래 명령들이 CCP 규격에 포함되어 있습니다. (그림 7). 교정 성능이 필요하지 않다면 모든 명령들이 구현될 필요는 없으며 따라서 아래 테이블 표에서 옵션으로 표기됩니다 (그림 7). GET_DAQ_SIZE, SET_DAQ_PTR, WRITE_DAQ, START_STOP (그리고 START_STOP_ALL) 는 DAQ 특성이며 이 자원들이 사용되지 않는 한 구현될 필요는 없습니다. GET_SEED 와 UNLOCK 은 key(암호) 로 보호된다면 데이터 취득과 교정 같은 CCP 자원들을 해체하는데 사용되며, 이것은 선택사항입니다.

Command	Code	TimeOut to ACK [ms]	Remark
CONNECT	0x01	25	
GET_CCP_VERSION	0x1B	25	
EXCHANGE_ID	0x17	25	
GET_SEED	0x12	25	optional command
UNLOCK	0x13	25	optional command
SET_MTA	0x02	25	
DNLOAD	0x03	25	
DNLOAD_6	0x23	25	optional command
UPLOAD	0x04	25	
SHORT_UP	0x0F	25	optional command
SELECT_CAL_PAGE	0x11	25	optional command
GET_DAQ_SIZE	0x14	25	
SET_DAQ_PTR	0x15	25	
WRITE_DAQ	0x16	25	
START_STOP	0x06	25	
DISCONNECT	0x07	25	
SET_S_STATUS	0x0C	25	optional command
GET_S_STATUS	0x0D	25	optional command
BUILD_CHKSUM	0x0E	30 000	optional command
CLEAR_MEMORY	0x10	30 000	optional command
PROGRAM	0x18	100	optional command
PROGRAM_6	0x22	100	optional command
MOVE	0x19	30 000	optional command
TEST	0x05	25	optional command
GET_ACTIVE_CAL_PAGE	0x09	25	optional command
START_STOP_ALL	0x08	25	optional command
DIAG_SERVICE	0x20	500	optional command
ACTION_SERVICE	0x21	5 000	optional command

그림 7: 명령

오류 처리

슬래이브에서의 오류 코드와 그것이 얼마나 중요한가에 따라, 마스터는 그림 8에 서술되어 있는 여러 가지 활동들을 실행합니다

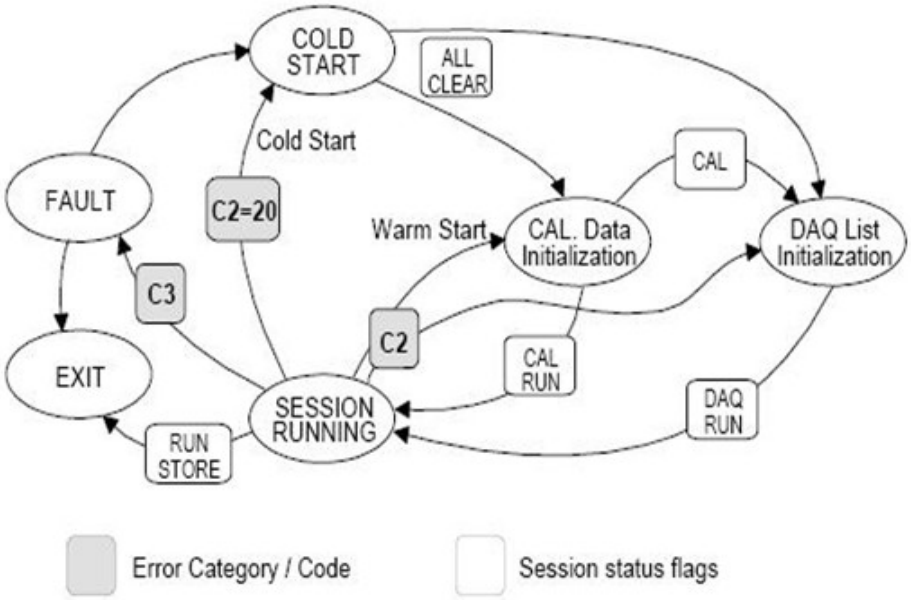


그림 8: 오류 처리

Error C0은 경고이며 아무 행동도 취하지 않습니다. 에러 C1이 발생한다면 통신 오류이거나 분주한 노드 전송에서의 오류입니다. 에러 C1에서 마스터는 그림 7에서 서술된 ACK 시간을 기다려야만 하며 그런 후 메시지를 재전송하려 시도해야 합니다. 이것은 2배가 될 것입니다. 에러 C2는 일시적 전력 손실이 될 수 있으며 재 초기화로 해결될 수 있습니다. 에러 C3은 해결될 수 없으며 마스터는 실행 세션을 종결시켜야 합니다.
Cold Start" 는 마스터와 슬래이브 사이에 새로운 논리 연결이 CONNECT 명령과 어떤 후속 초기화로 성립됨을 뜻합니다.

순차의 예

예제들에는 마스터로서, 기본 CCP 통신을 위하여, 사용할 명령들이 설명되어 있습니다.

로그인 세션 (Cold Start)
마스터와 슬래이브 사이의 전형적인 연결은 마스터에서의 CONNECT 명령을 포함하는 CRO로 시작합니다. 슬래이브는 이에 해당하는 DTO로 응답해야 합니다. 마스터와 슬래이브 모두 동일한 "언어"로 대화하는 것을 확실히 하기 위하여 GET_CCP_VERSION 명령이 예상했던 버전 번호와 같이 마스터에서 전송됩니다. 만약 슬래이브에서 반송되어 전달된 그 버전 번호가 일치한다면, 통신이 진행될 수 있습니다. "플러그 앤 플레이" 호환 노드들을 위해 EXCHANGE_ID 명령이 스테이션 주소에 따라, 자동 세션 구성을 위해 사용될 수 있습니다.

GET_SEED 명령에서 슬래이브 노드는 자원 (DAQ or Calibration) 의 보호 상태 (locked/unlocked) 에 관한 정보를 응답합니다. 만약, 어떤 이유로, 자원이 lock 된다면 이것은 사용되기 전에 unlock되어야 합니다.
자원을 해제(unlock)하기 위해서는 UNLOCK 명령이, GET_SEED DTO에서 수신 받은 "KEY" 와 함께 마스터로부터 전송되어야 합니다. 로그인 세션이 끝나기 전에 Status 비트의 초기화가 권장되는데, 이는 SET_S_STATUS 명령으로 실행됩니다.

교정 초기화 세션(Calibration init session)

이 세션 설명은 로그인 절차가 수행되었다는 것을 가정합니다. 그런 다음 첫 번째는 SET_S_STATUS 명령을 이용하여 교정에 대한 세션 상태 비트를 "off" 로 설정하는 것이 될 것입니다. 그 후 교환할 데이터를 포함하는 메모리 주소가 SET_MTA 명령으로 선택됩니다. 이 메모리 주소가 이용 가능하다는 것을 확실히 하기 위해 BUILD_CHKSUM 명령이 전송되고 이를 확인하는 슬래이브 노드로부터의 응답이 예정됩니다. 그러면 데이터 바이트를 선택된 주소로 다운로드 할 수 있습니다. 먼저 DOWNLOAD 명령이 데이터 바이트의 수와 각 바이트의 값과 더불어 전송됩니다. 데이터 교환을 실제로 수행하기 위해서 SELECT_CAL_PAGE 가 전송됩니다. 교정이 시작되었다는 것을 지시하기 위해서 교정을 위한 세션 상태 비트가 SET_S_STATUS 명령으로 "on" 으로 설정됩니다.

DAQ init 세션

이 세션 설명은 로그인 절차가 수행 되었다는 것을 가정합니다. DAQ의 세션 상태 비트는 SET_S_STATUS 로 "off"로 설정됩니다. DAQ 리스트는 GET_DAQ_SIZE 로 선택되며 슬래이브는 사용 가능한 ODT의 수로 응답합니다. SET_DAQ_POINTER 명령은 어떤 DAQ 리스트에, ODT의 element 와 ODT 테이블이 기록되어야 하는지를 선택합니다. 그러면 WRITE_DAQ 명령은 데이터 파라매터의 메모리 주소를 이전에 선택된 element에 할당합니다. 모든 DAQ 리스트들이 원하는 대로 채워지면 DAQ의 세션 상태 비트가 SET_S_STATUS 를 사용하여 "on"으로 설정됩니다. DAQ 리스트의 전송은 START_STOP 명령으로 시작됩니다. 여러 DAQ 리스트들이 동시에 시작되어야 하고 동기적으로 전송되어야 한다면 START_STOP_ALL 명령이 사용됩니다.