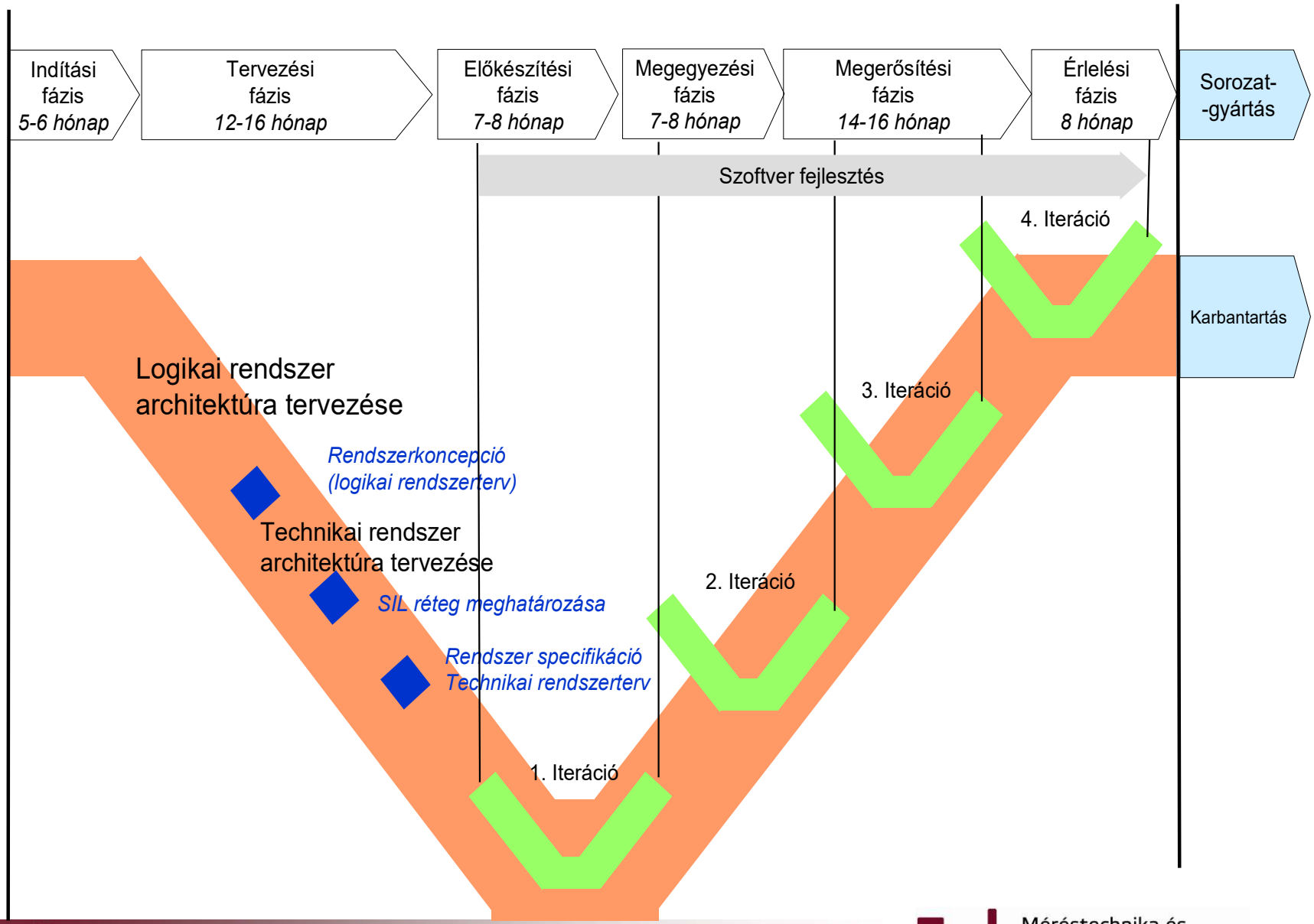
Diagnosztika

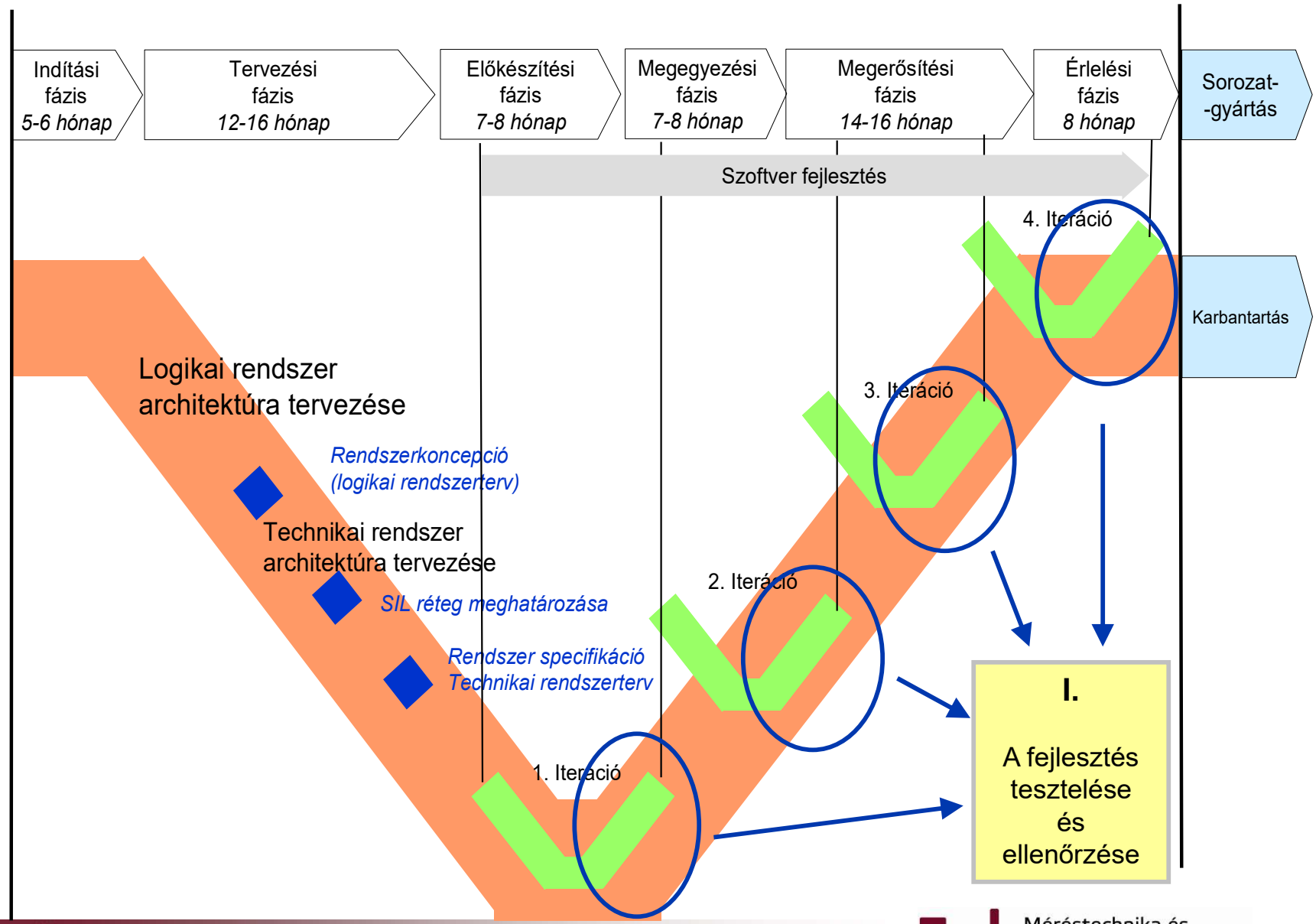
Méréstechnika és
Információs Rendszerek
Tanszék

A diagnosztikai kommunikáció szerepe

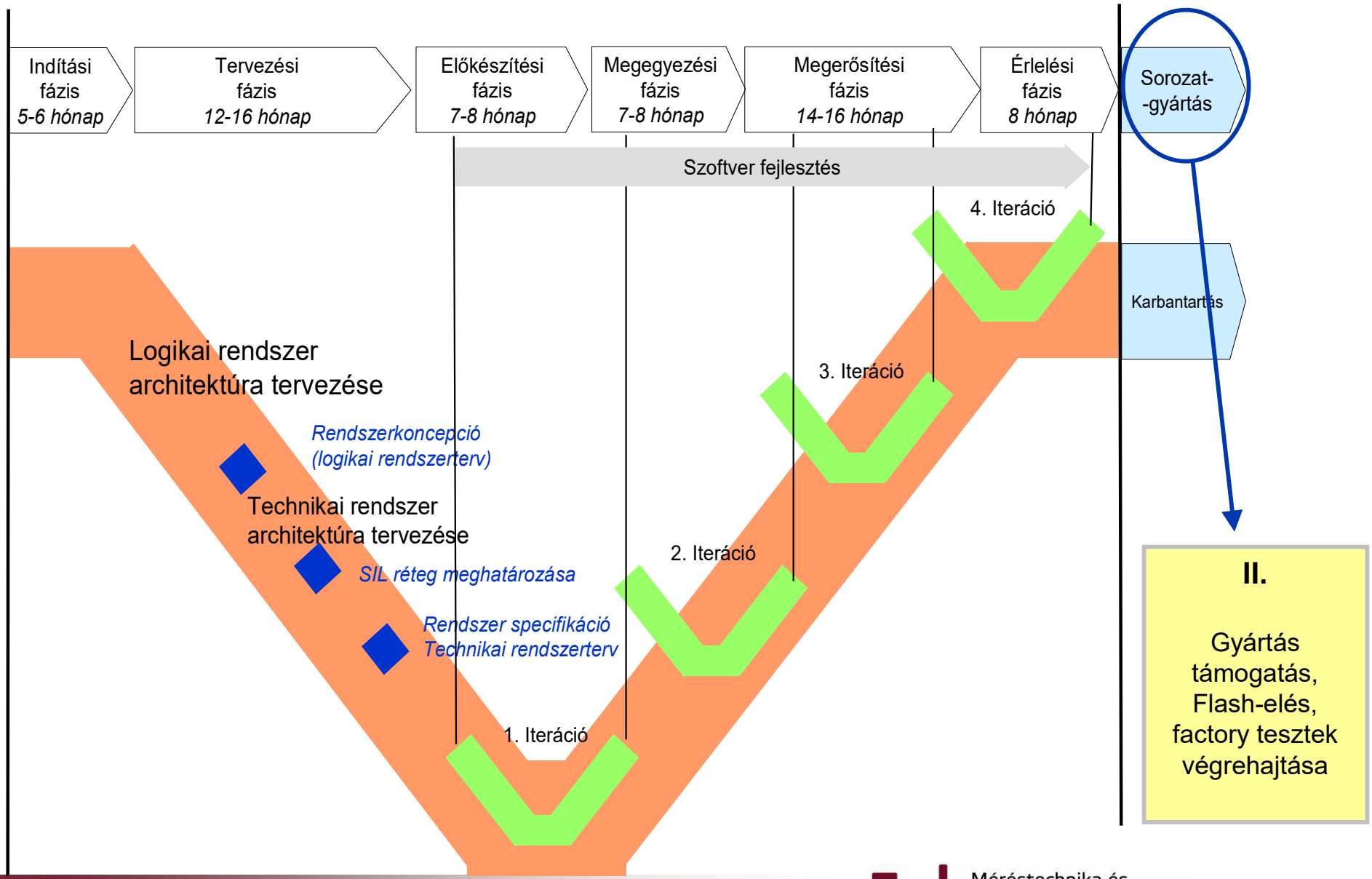
Egy általános autóipari fejlesztés életciklusa



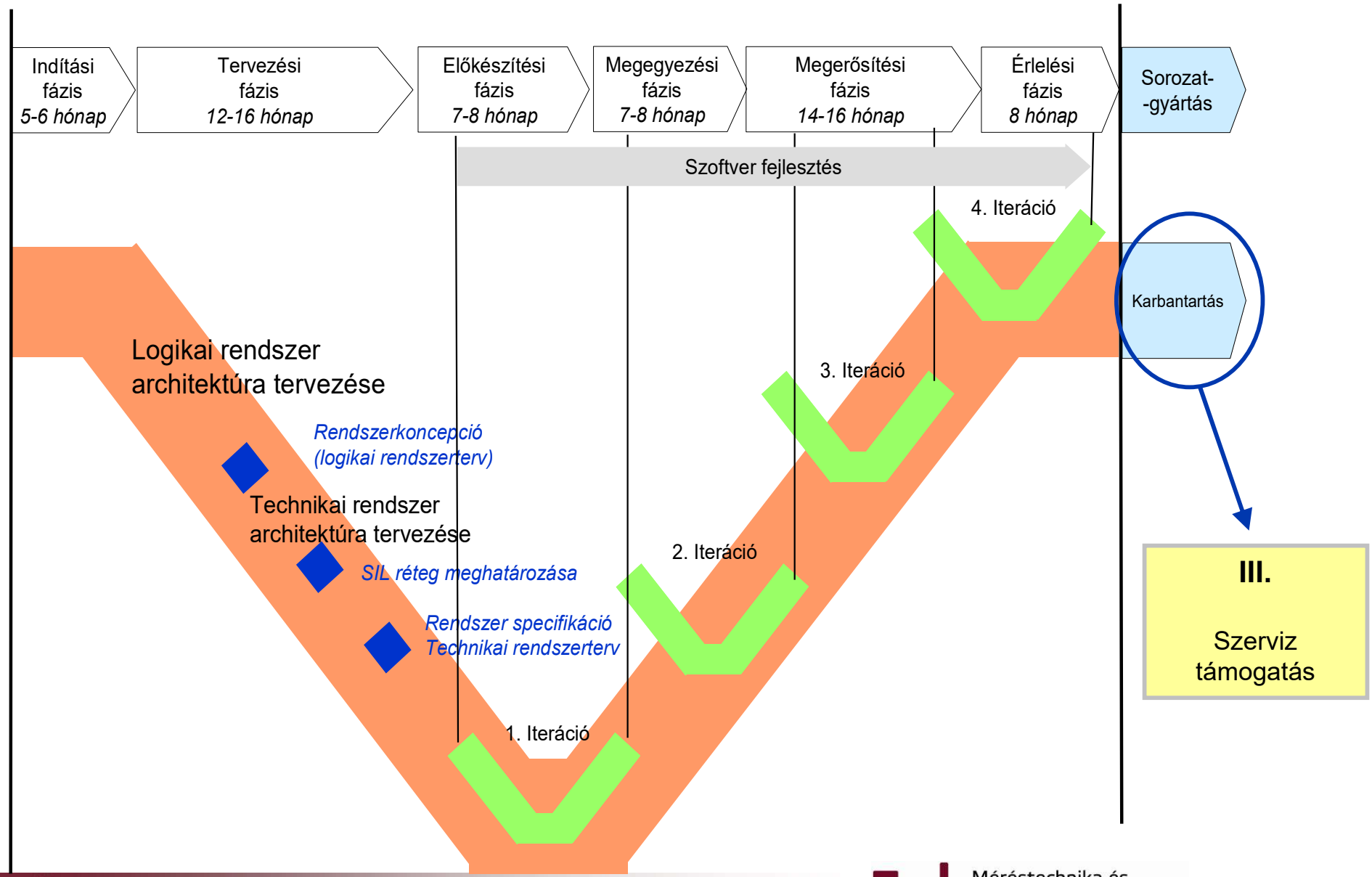
A diagnosztikai protokollok helye és szerepe az életciklusban



A diagnosztikai protokollok helye és szerepe az életciklusban



A diagnosztikai protokollok helye és szerepe az életciklusban

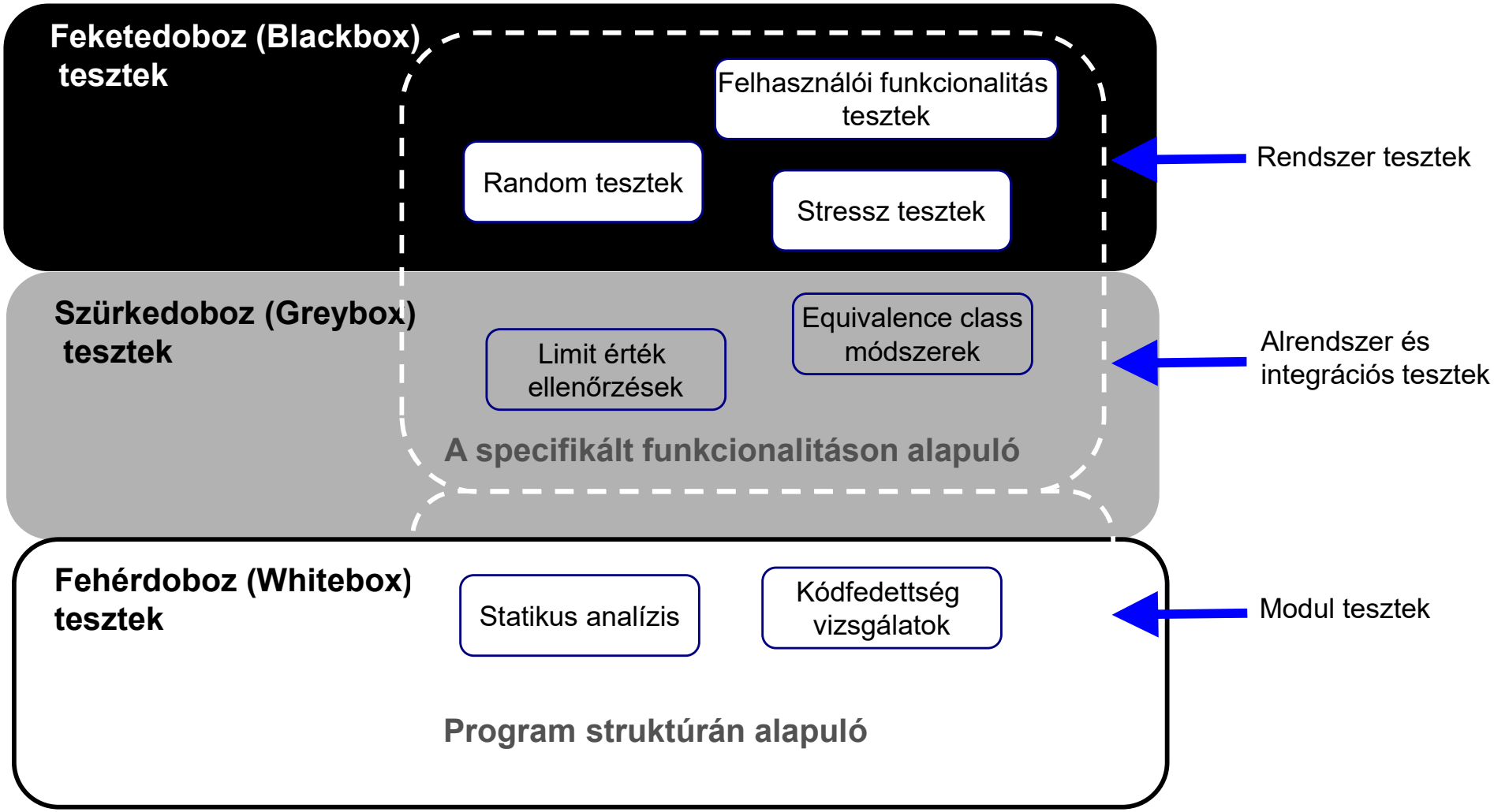


A diagnosztikai kommunikációk célja

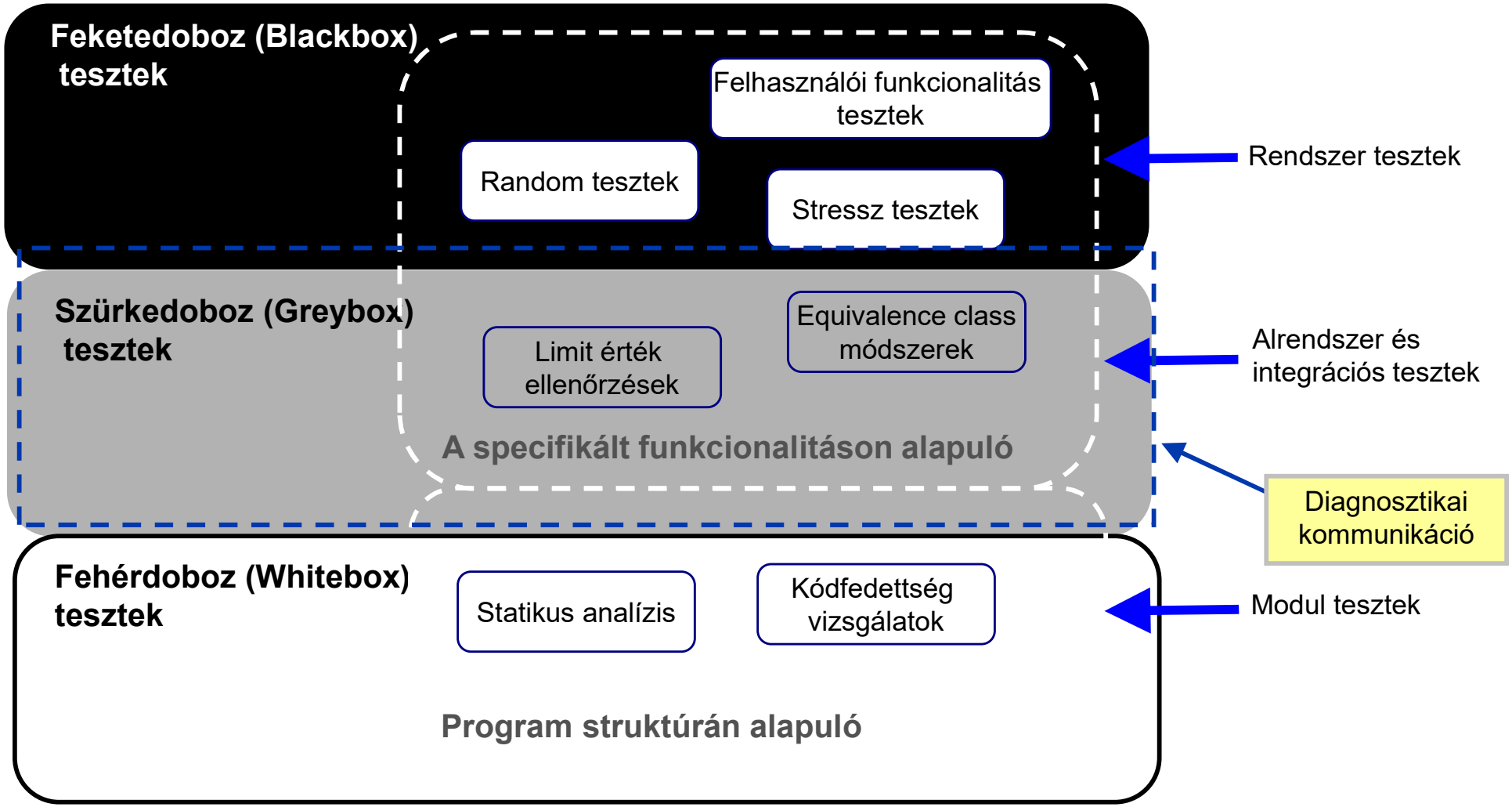
- ECU megfigyelése:
  - Folyamatos adatgyűjtés
  - Polling alapú adatgyűjtés
- ECU vezérlése
  - Adatok, konstansok, kalibrációs adatok megváltoztatása
- ECU programmemória frissítése
- Az ECU hibatárolójának kezelése
  - Lekérdezés
  - Törlés
- Felhasználó azonosítása

Diagnosztikai kommunikáció szerepe a fejlesztési tesztelésnél

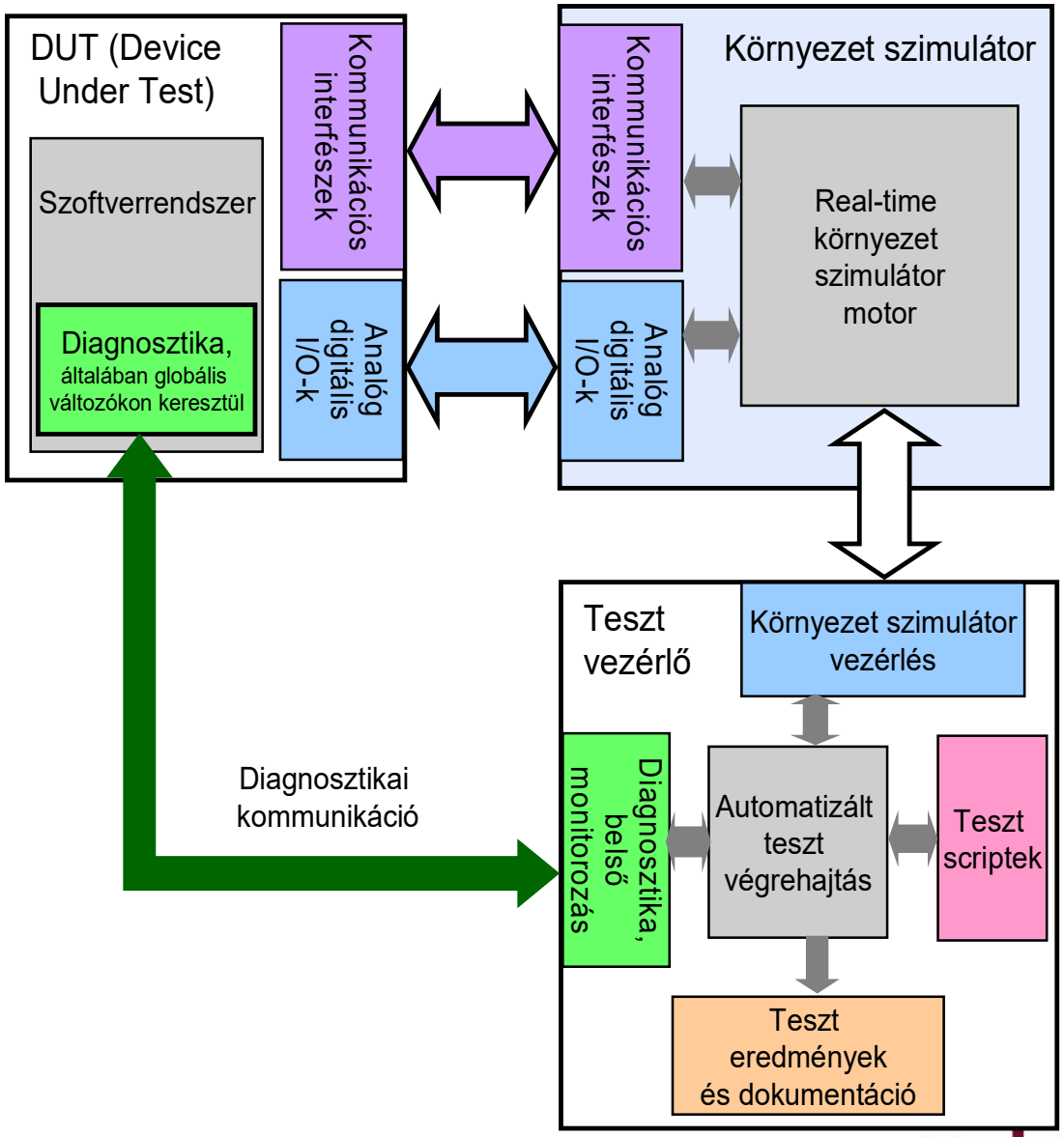
Tesztelés



Tesztelés



Tipikus fejlesztési tesztkörnyezet szürke és feketedoboz tesztekhez



CCP (CAN Calibration Protocol)

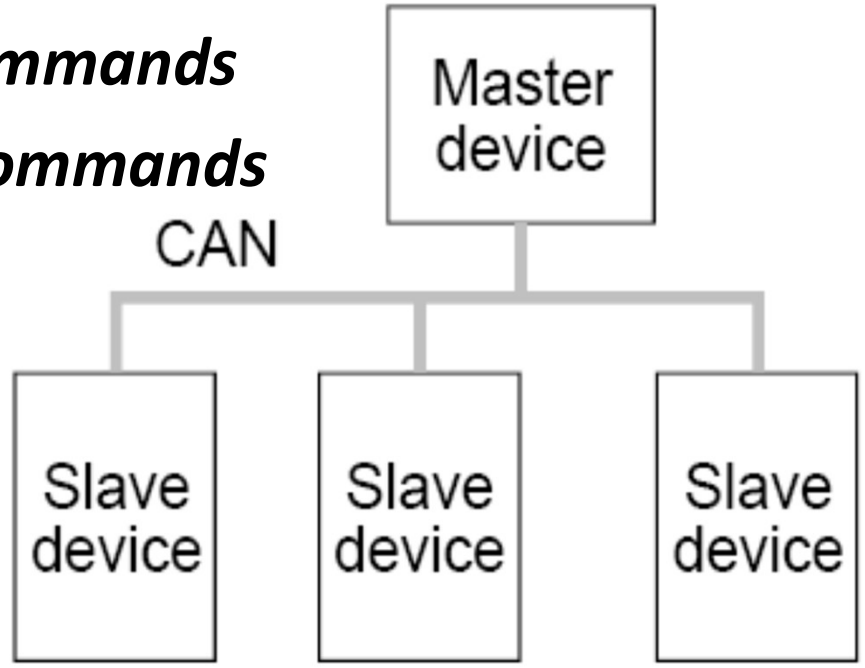
CCP (CAN Calibration Protocol) bevezető

- ***Fejlesztéstámogatás a fő cél, nem a szerviz diagnosztika***

- Eredetileg Bosch, Intel (1992) később átvette a specifikáció fejlesztését az ASAP
- CAN Calibration Protocol Version 2.1
  - 1999

CCP architektúrája

- Master-Slave elrendezés
- Kétfajta üzenettípus
  - ***Generic Control Commands***
  - ***Data Acquisition Commands***

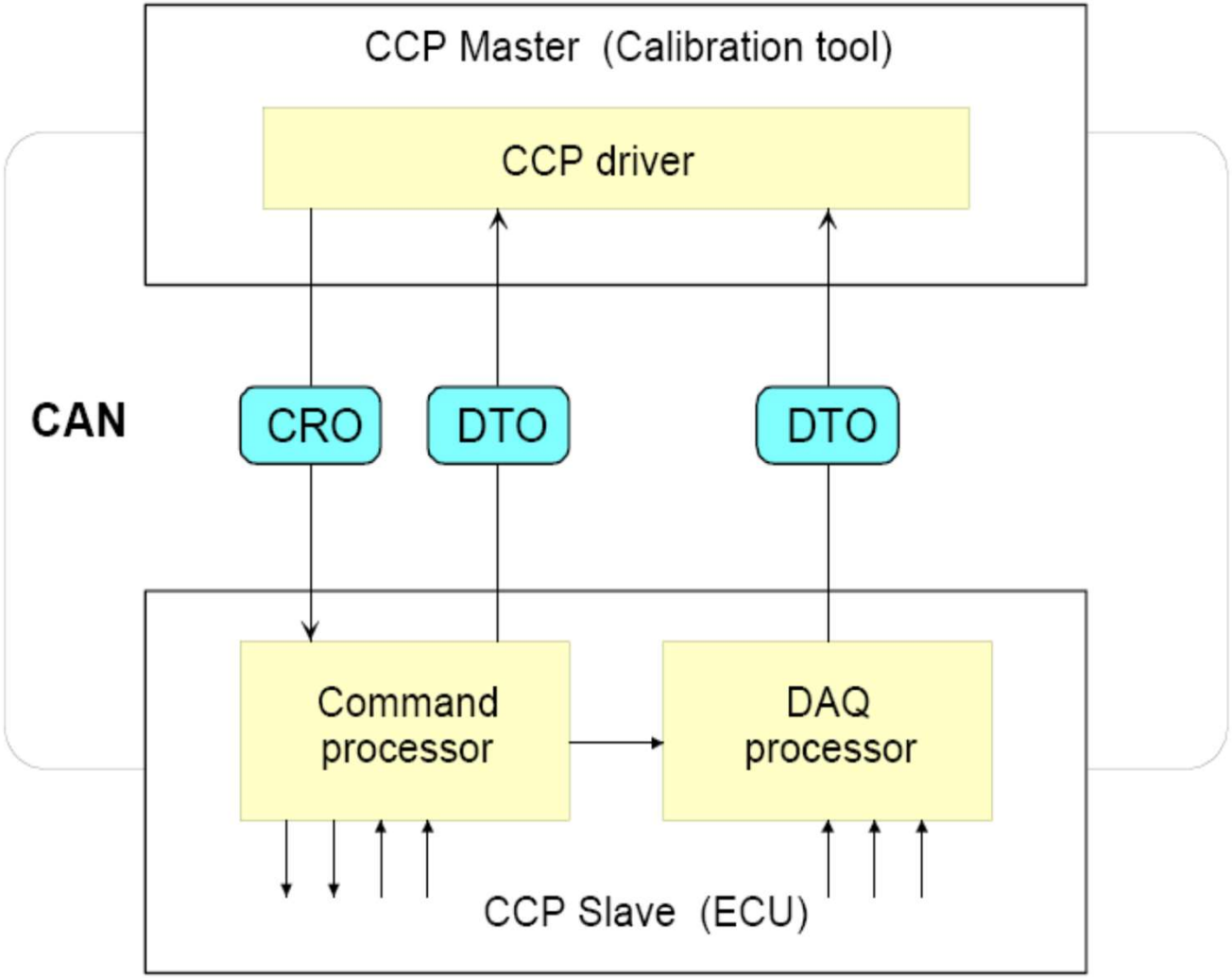


Configuration of Master and Slave devices

CCP üzenet típusai

- Két alap üzenettípus
  - Command Receive Object : CRO (Master/Teszter parancs)
  - Data Transmission Object : DTO
    - ECU válasz: Command Return Message
    - Event Message (Az ECU belső állapotváltozására hívja fel a figyelmet)
    - Data Acquisition Message

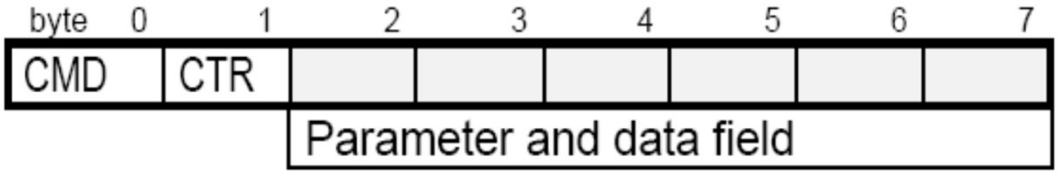
CCP üzenet típusai



Command Receive Object : CRO

- Mindig 8 byte-os CAN üzenet

Position	Type	Description
0	byte	command code = CMD
1	byte	command counter = CTR
2...7	bytes	command related parameter and data area



Data Transmission Object DTO

- Mindig 8 byte-os CAN üzenet

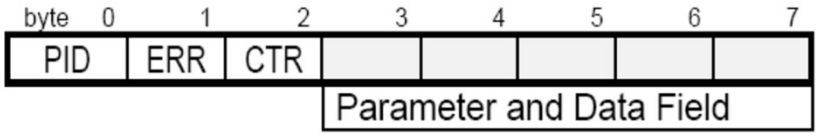
Position	Type	Description
0	byte	data packet ID = PID
1 - 7	byte	data packet

- A Packet ID jelentése

PID	Interpretation	Note
0xFF	DTO contains a Command Return Message	
0xFE	DTO contains an Event Message	CTR is don't care
$0 \leq n \leq$ 0xFD	DTO contains a Data Acquisiton Message corresponding to ODT n	see chapter 'Descriptions of Commands'

Data Transmission Object DTO folytatás

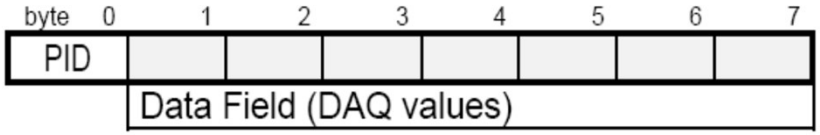
- Command Return és az Event Messages formátuma



ERR: Command Return- / Error Code.

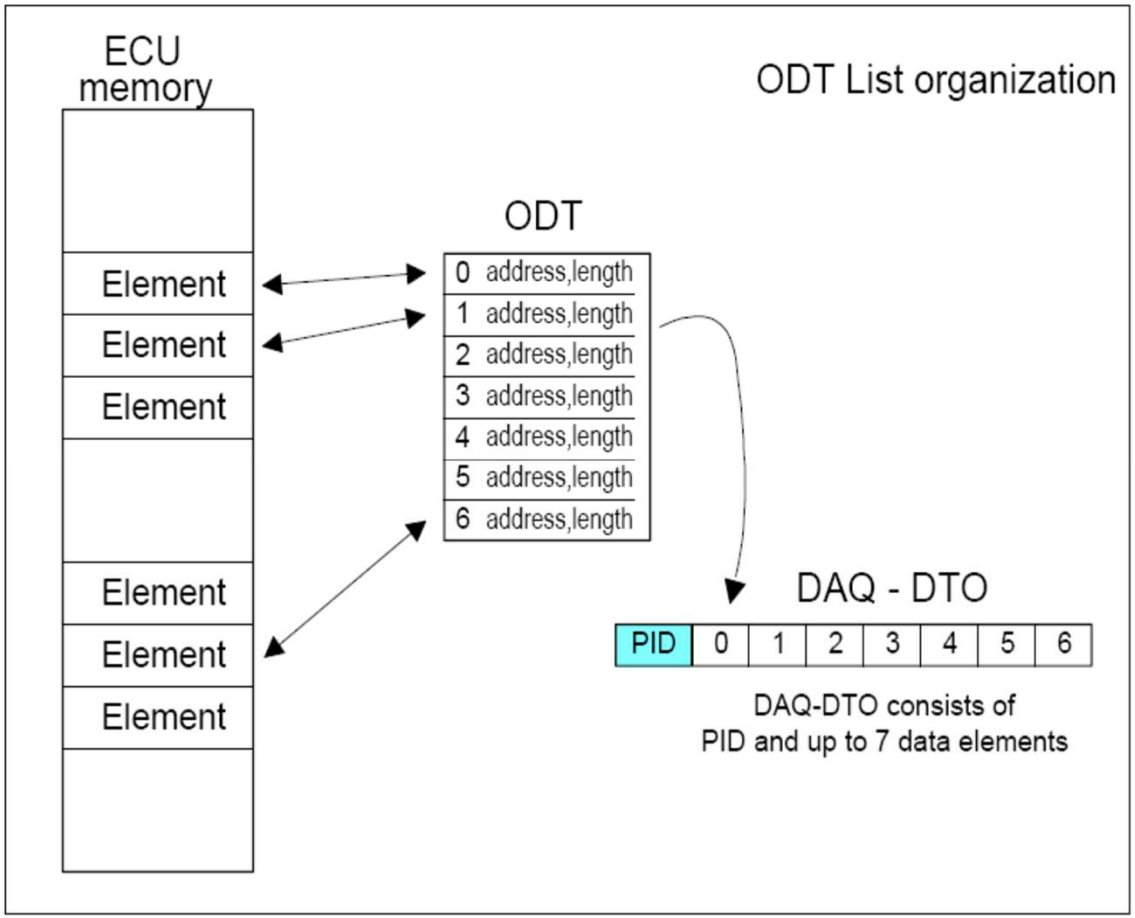
CTR: Command Counter as received in CRO with the last command.

- Data Acquisiton Messages formátuma
  - A PID egyben a megfelelő ODT (Object Desciptor Table) is jelenti

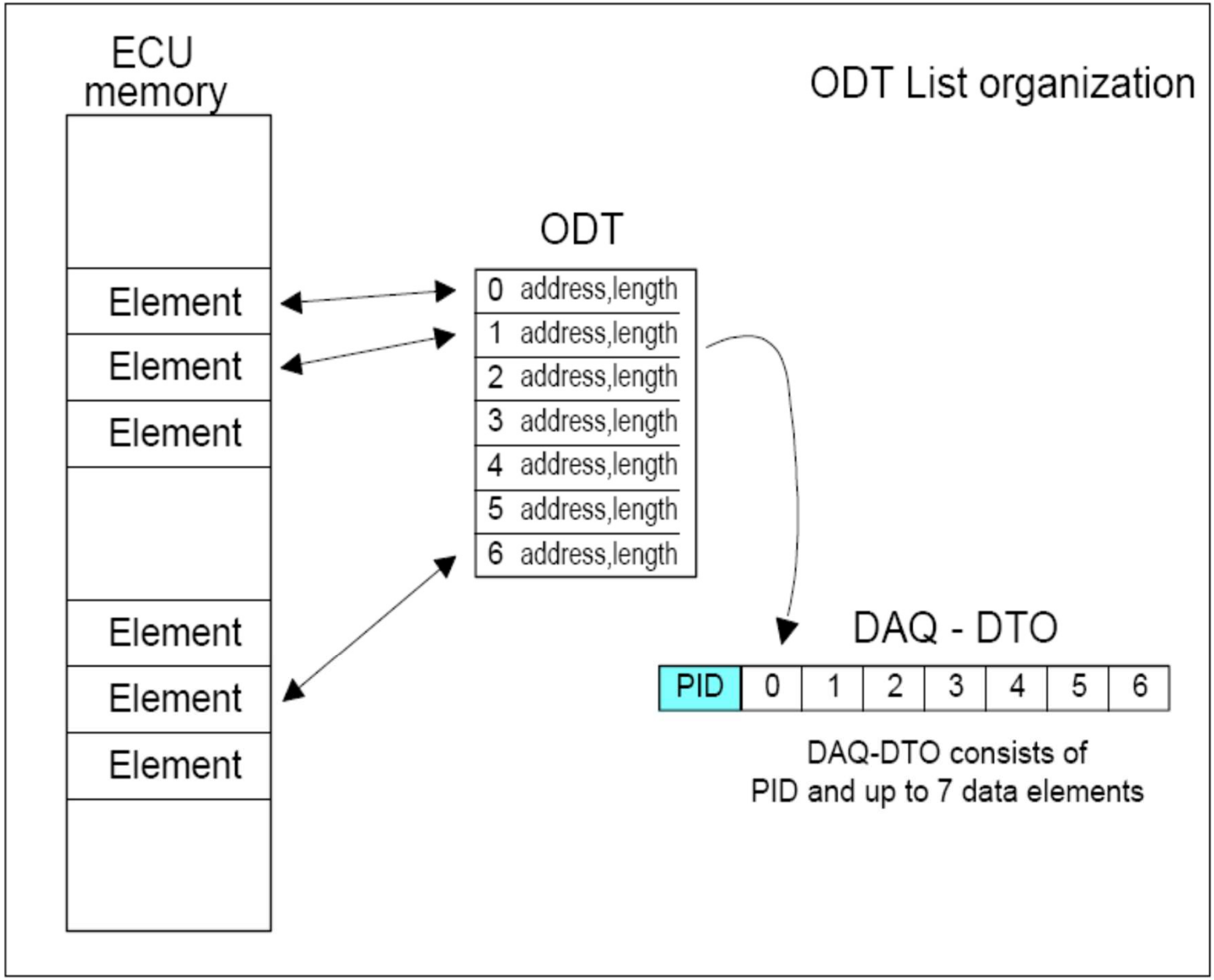


Data Acquisiton táblázatok

- A DAQ üzenetekben az egyes ODT listákban megadott adatokat küldjük el. PID azonosítja az ODT listát
- Az ODT (Object Descriptor Table) lista egy memórialeképezés

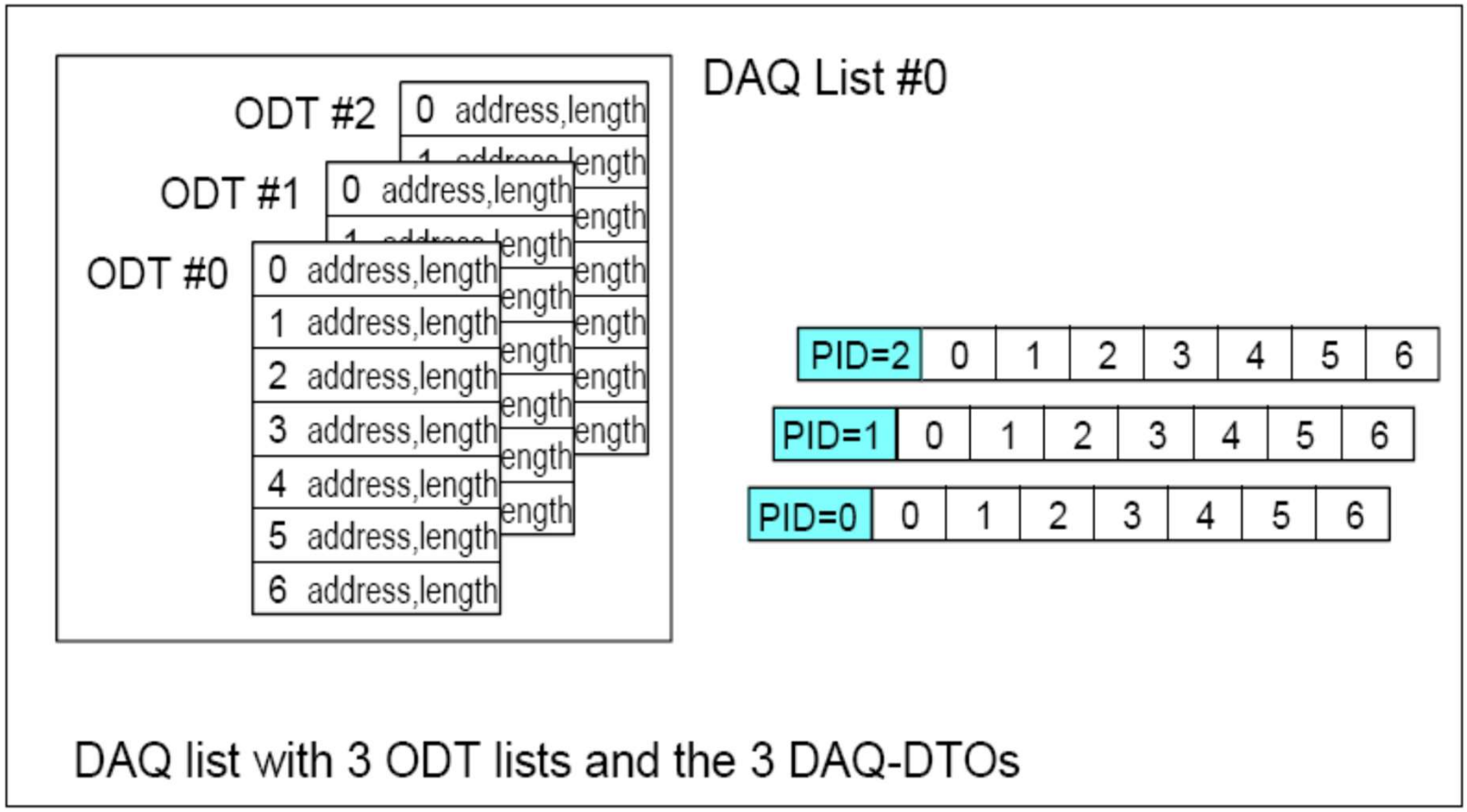


Data Acquisiton táblázatok



Data Acquisiton táblázatok

- ODT sorszáma 0 – 0xFD
- Egy DAQ lista több ODT listát tartalmaz



CCP parancsok

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

CCP hibakódok

Code	Description	Error category	State transition to	
0x00	acknowledge / no error	-		
0x01	DAQ processor overload	C0		
0x10	command processor busy	C1	NONE (wait until ACK or timeout)	
0x11	DAQ processor busy	C1	NONE (wait until ACK or timeout)	
0x12	internal timeout	C1	NONE (wait until ACK or timeout)	
0x18	key request	C1	NONE (embedded seed&key)	
0x19	session status request	C1	NONE (embedded SET_S_STATUS)	
0x20	cold start request	C2	COLD START	
0x21	cal. data init. request	C2	cal. data initialization	
0x22	DAQ list init. request	C2	DAQ list initialization	
0x23	code update request	C2	(COLD START)	
0x30	unknown command	C3	(FAULT)	
0x31	command syntax	C3	FAULT	
0x32	parameter(s) out of range	C3		
0x33	access denied	C3		
0x34	overload	C3		
0x35	access locked	C3		
0x36	resource/function not available	C3		

Category	Description	Action
timeout	no handshake message	retry
C0	warning	-
C1	spurious (comm error, busy, ...)	wait (ACK or timeout)
C2	resolvable (temp. power loss, ...)	reinitialize
C3	unresolvable (setup, overload, ...)	terminate

CCP Implementáció

- Vector CCP implementáció szabadon letölthető
- Kis programmemória, adatmemória igény méret
  - 6Kbyte Flash
  - 200byte RAM
- Könnyen protolható 3 CAN kommunikációs függvény
  - ccpSend
  - ccpSendCallBack
  - ccpCommand
- Hozzárakható plusz funkcionalitás
  - ccpDaq adatgyűjtés
  - EEPROM írás, olvasás
- Nem tartalmaz
  - Flash-elést
  - Több byte-os adatok direkt kezelését

CCP Példa

- ECU 0x112 CAN ID-n várja a parancsot, 0x111 CAN ID-n válaszol
- Első lépés kapcsolatfelvétel CONNECT (1-es parancs)

Position	Type	Description
0	byte	Command Code = CONNECT 0x01
1	byte	Command Counter = CTR
2	word	station address (Intel format)
4...7	bytes	don't care

Example

The master device sends a CONNECT CRO to the slave device with the station address 0x0200. The command counter CTR currently is 0x45:

byte 0	1	2	3	4	5	6	7
0x01	0x45	0x00	0x02	--	--	--	--

The slave device answers with a DTO containing ACKNOWLEDGE (0x00) and the CTR of the CRO:

byte 0	1	2	3	4	5	6	7
0xFF	0x00	0x45	--	--	--	--	--

CCP Példa

- Második lépés a DAQ lista lekérdezése
  - Get Size of DAQ list 0x14

Position	Type	Description
0	byte	Command Code = GET_DAQ_SIZE 0x14
1	byte	Command Counter = CTR
2	byte	DAQ list number (0,1,...)
3	byte	don't care
4..7	unsigned long	CAN Identifier of DTO dedicated to list number

  - Válasz

Position	Type	Description
0	byte	Packet ID: 0xFF
1	byte	Command Return Code
2	byte	Command Counter = CTR
3	byte	DAQ list size (= number of ODTs in this list)
4	byte	First PID of DAQ list
5 ... 7	bytes	don't care

CCP Példa

- Második lépés a DAQ lista lekérdezése

The PID of a specific ODT of a DAQ list is determined by:

PID = First PID of DAQlist + ODT number

Example

The master device sends a GET_DAQ_SIZE CRO to the slave device. The command counter CTR currently is 0x23, the DAQ list number is 0x03 with the ID 0x01020304.

byte 0	1	2	3	4	5	6	7
0x14	0x23	0x03	--	0x01	0x02	0x03	0x04

The slave device answers with a DTO containing ACKNOWLEDGE (0x00), the CTR of the CRO, the first PID=0x08 and the list size 0x10 (10 ODT with up to 7 elements each)

byte 0	1	2	3	4	5	6	7
0xFF	0x00	0x23	0x10	0x08	--	--	--

CCP Példa

- Harmadik lépés a DAQ lista írásához beállítani a mutatót
  - Set DAQ list pointer 0x15

Position	Type	Description
0	byte	Command Code = SET_DAQ_PTR 0x15
1	byte	Command Counter = CTR
2	byte	DAQ list number (0,1,...)
3	byte	Object Descriptor Table ODT number (0,1,...)
4	byte	Element number within ODT (0,1,...)
5...7	bytes	don't care

  - Válasz

Position	Type	Description
0	byte	Packet ID: 0xFF
1	byte	Command Return Code
2	byte	Command Counter = CTR
3 ... 7	bytes	don't care

CCP Példa

- Harmadik lépés a DAQ lista írásához beállítani a mutatót

Example

The master device sends a SET_DAQ_PTR CRO to the slave device. The command counter CTR currently is 0x23, the DAQ list number is 0x03, the ODT number is 0x05 and the element number addressed is 0x02.

byte 0	1	2	3	4	5	6	7
0x15	0x23	0x03	0x05	0x02	--	--	--

The slave device answers with a DTO containing ACKNOWLEDGE (0x00) and the CTR of the CRO:

byte 0	1	2	3	4	5	6	7
0xFF	0x00	0x23	--	--	--	--	--

Next the command WRITE_DAQ is used to set the data elements in the selected ODT.

CCP Példa

- Negyedik lépés felvenni egy elemet a DAQ listába
  - WRITE_DAQ 0x16

Position	Type	Description
0	byte	Command Code = WRITE_DAQ 0x16
1	byte	Command Counter = CTR
2	byte	Size of DAQ element in bytes { 1, 2, 4 }
3	byte	Address extension of DAQ element
4...7	unsigned long	Address of DAQ element

  - Válasz

Position	Type	Description
0	byte	Packet ID: 0xFF
1	byte	Command Return Code
2	byte	Command Counter = CTR
3 ... 7	bytes	don't care

CCP Példa

- Negyedik lépés felvenni egy elemet a DAQ listába

Example

The master device sends an WRITE_DAQ CRO to the slave device. The command counter CTR currently is 0x23, the size of the DAQ element is 0x02 bytes, the address extension is 0x01 and the 32-bit address is 0x02004200.

byte	0	1	2	3	4	5	6	7
	0x16	0x23	0x02	0x01	0x02	0x00	0x42	0x00

The slave device answers with a DTO containing ACKNOWLEDGE (0x00) and the CTR of the CRO:

byte	0	1	2	3	4	5	6	7
	0xFF	0x00	0x23	--	--	--	--	--

CCP Példa

- Ötödik lépés elindítani az adatgyűjtést
  - START_STOP 0x06
    - 0x00 stops specified DAQ list, 0x01 starts specified DAQ list

Position	Type	Description
0	byte	Command Code = START_STOP 0x06
1	byte	Command Counter = CTR
2	byte	Mode : start / stop / prepare data tranmission
3	byte	DAQ list number
4	byte	Last ODT number
5	byte	Event Channel No.
6, 7	word	Transmission rate prescaler

  - Válasz

Position	Type	Description
0	byte	Packet ID: 0xFF
1	byte	Command Return Code
2	byte	Command Counter = CTR
3 ... 7	bytes	don't care

CCP Példa

- Ötödik lépés elindítani az adatgyűjtést

Example

The master device sends a START_STOP CRO to the slave device. The command counter CTR currently is 0x23, the start/stop byte is 0x01 (start), the DAQ list number is 0x03and the packet number to transmit is 0x07. The ECU shell use the event-channel 0x02 with a prescaler of 1 (motorola-format).

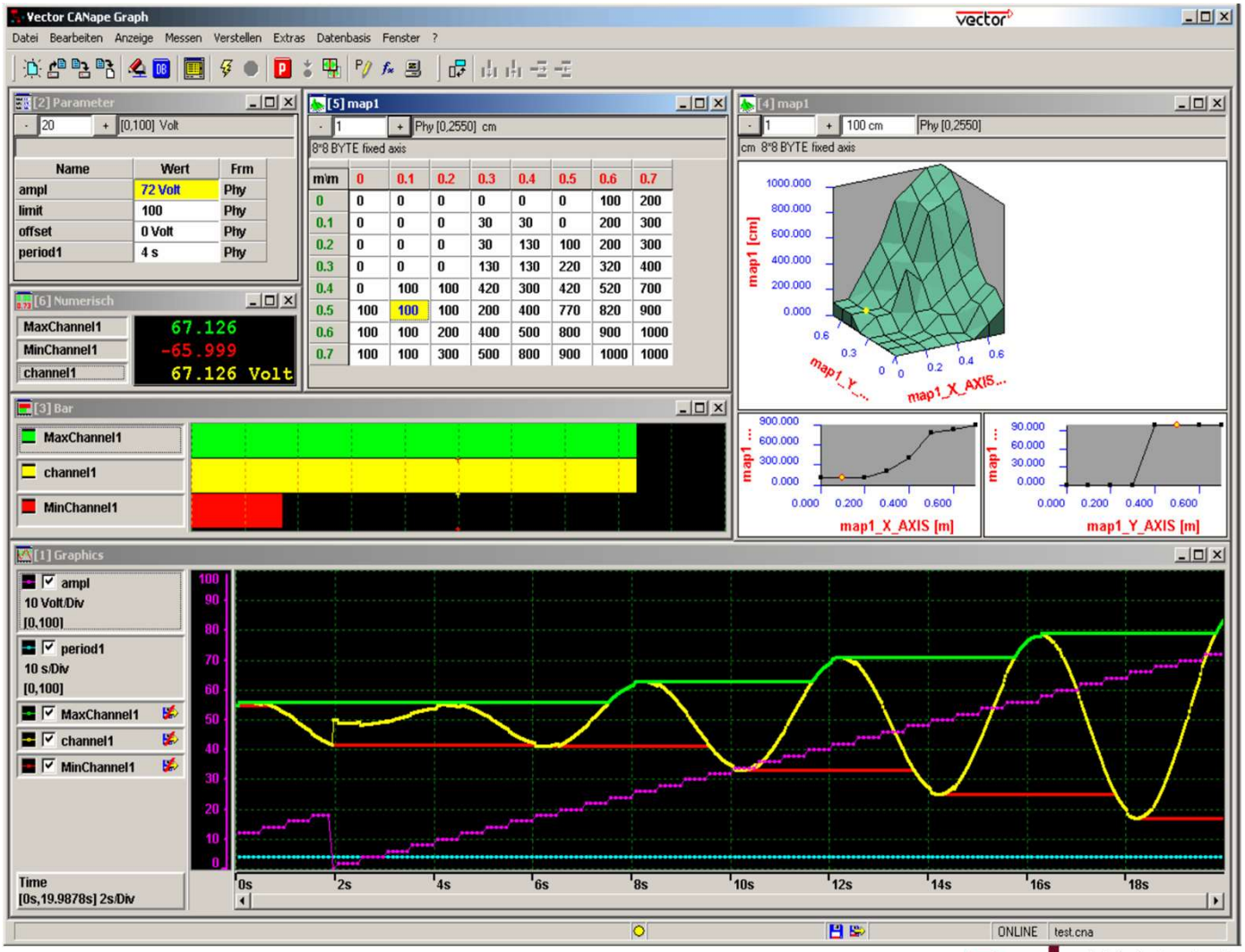
byte 0	1	2	3	4	5	6	7
0x06	0x23	0x01	0x03	0x07	0x02	0x00	0x01

The slave device answers with a DTO containing ACKNOWLEDGE (0x00) and the CTR of the CRO:

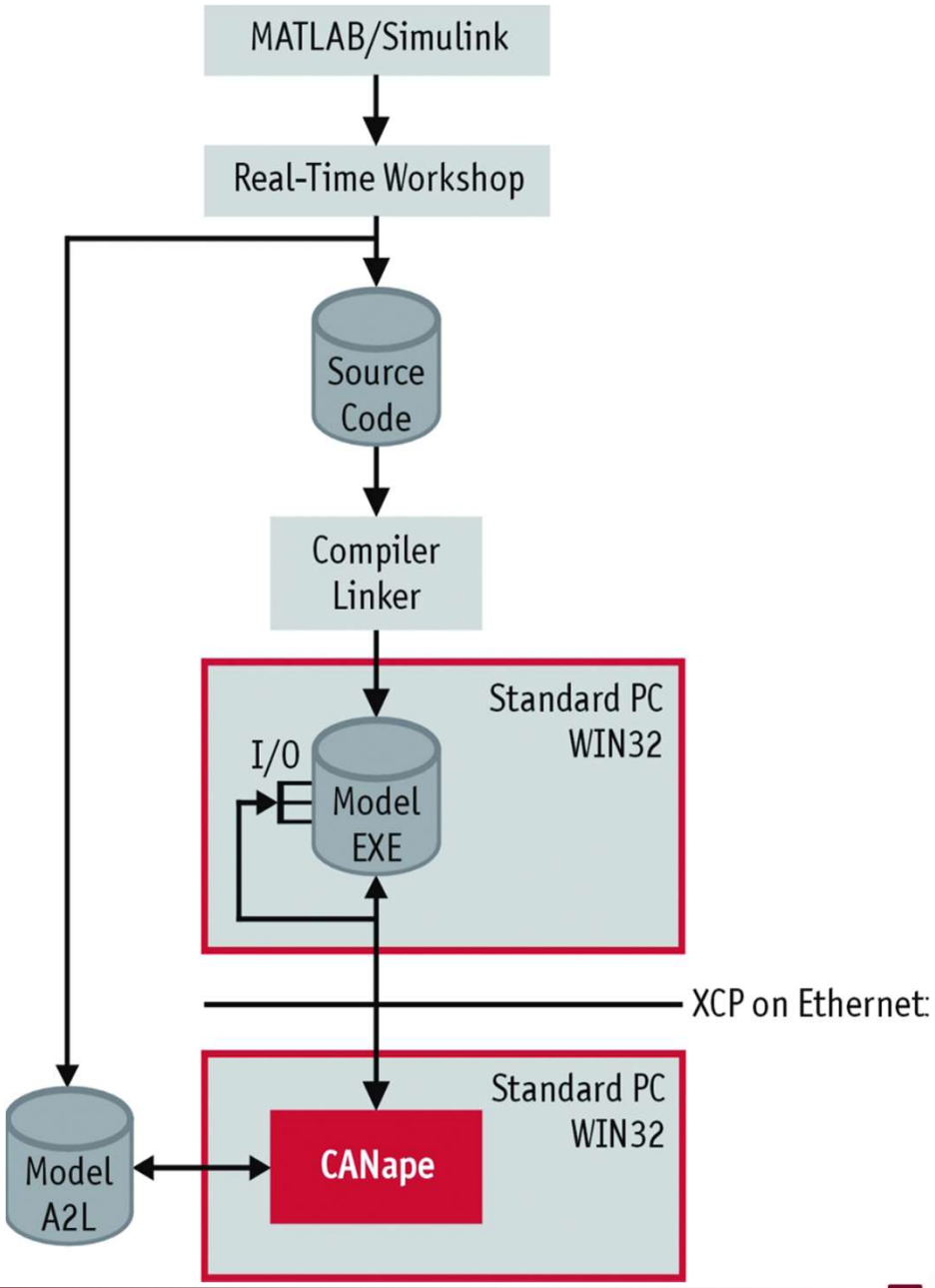
byte 0	1	2	3	4	5	6	7
0xFF	0x00	0x23	--	--	--	--	--

CCP Profi alkalmazások

- Vector CANape

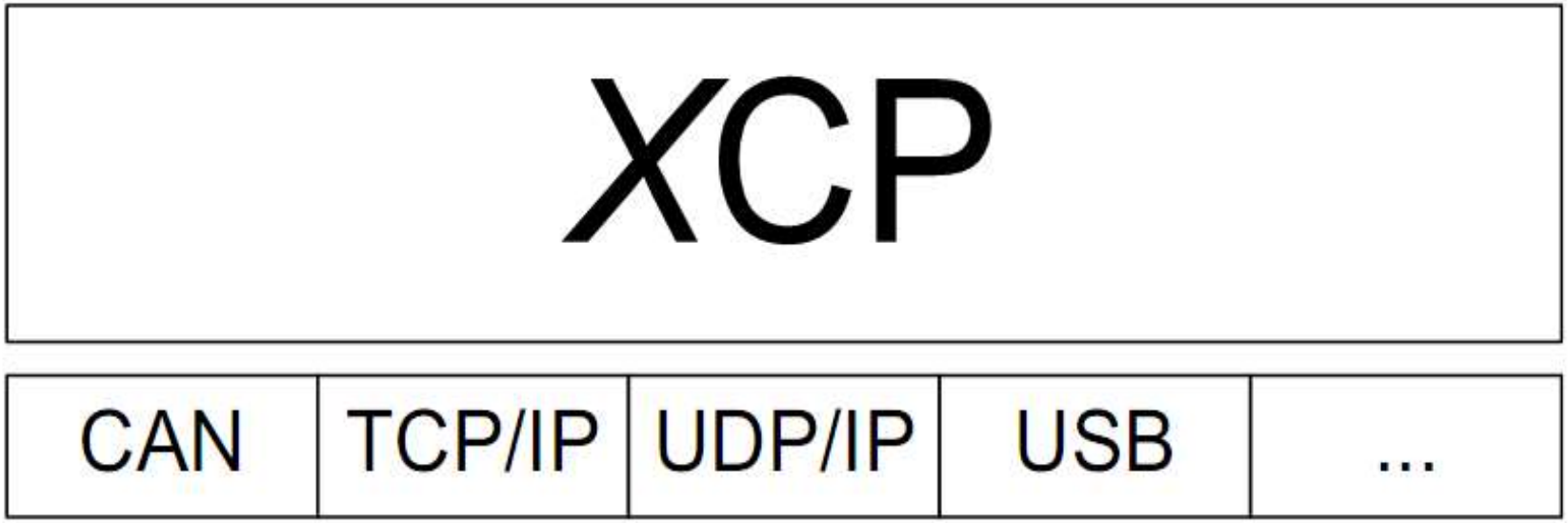


CANape ajánlott fejlesztési folyamat



XCP (Universal Measurement and Calibration Protocol)

- X a különféle adatkapcsolati megoldásokat szimbolizálja



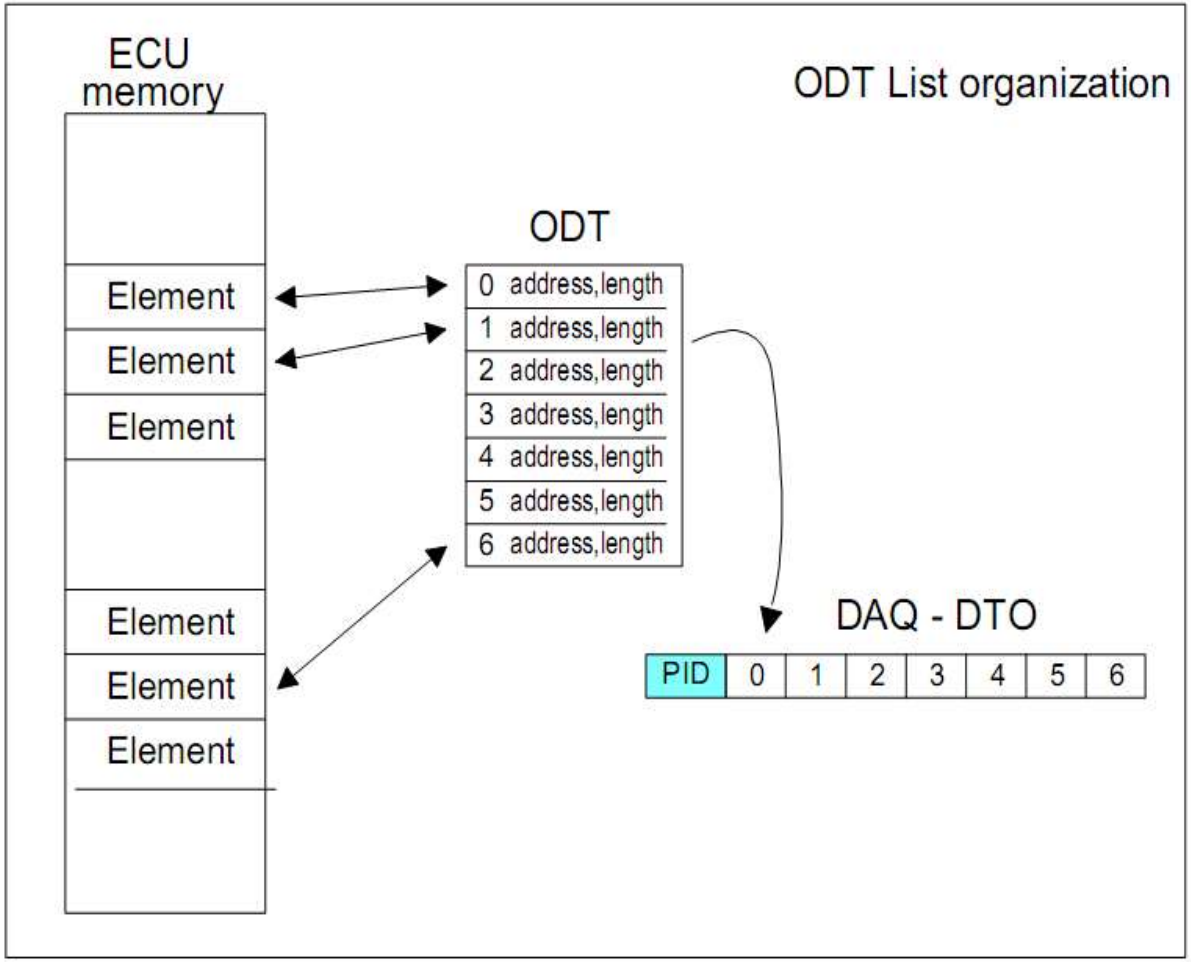
Az XCP tulajdonságai

- XCP alap tulajdonságok
  - Szinkron Adat olvasás/gyűjtés
  - Szinkron Adat írás/megváltoztatás
  - Online memória kalibráció
  - Kalibrációs memória inicializáció és váltás
  - Flash Programozás

- XCP opcionális tulajdonságok
  - Blokkos kommunikáció
  - Interleaved kommunikációs mód
  - Dinamikus adat transzfer konfiguráció
  - Időbélyegezett adat transzfer
  - Adat transzfer szinkronizálása
  - Az adatmenedzsment prioritizálása
  - Atomikus bit modification
  - Bit alapú adat változtatás

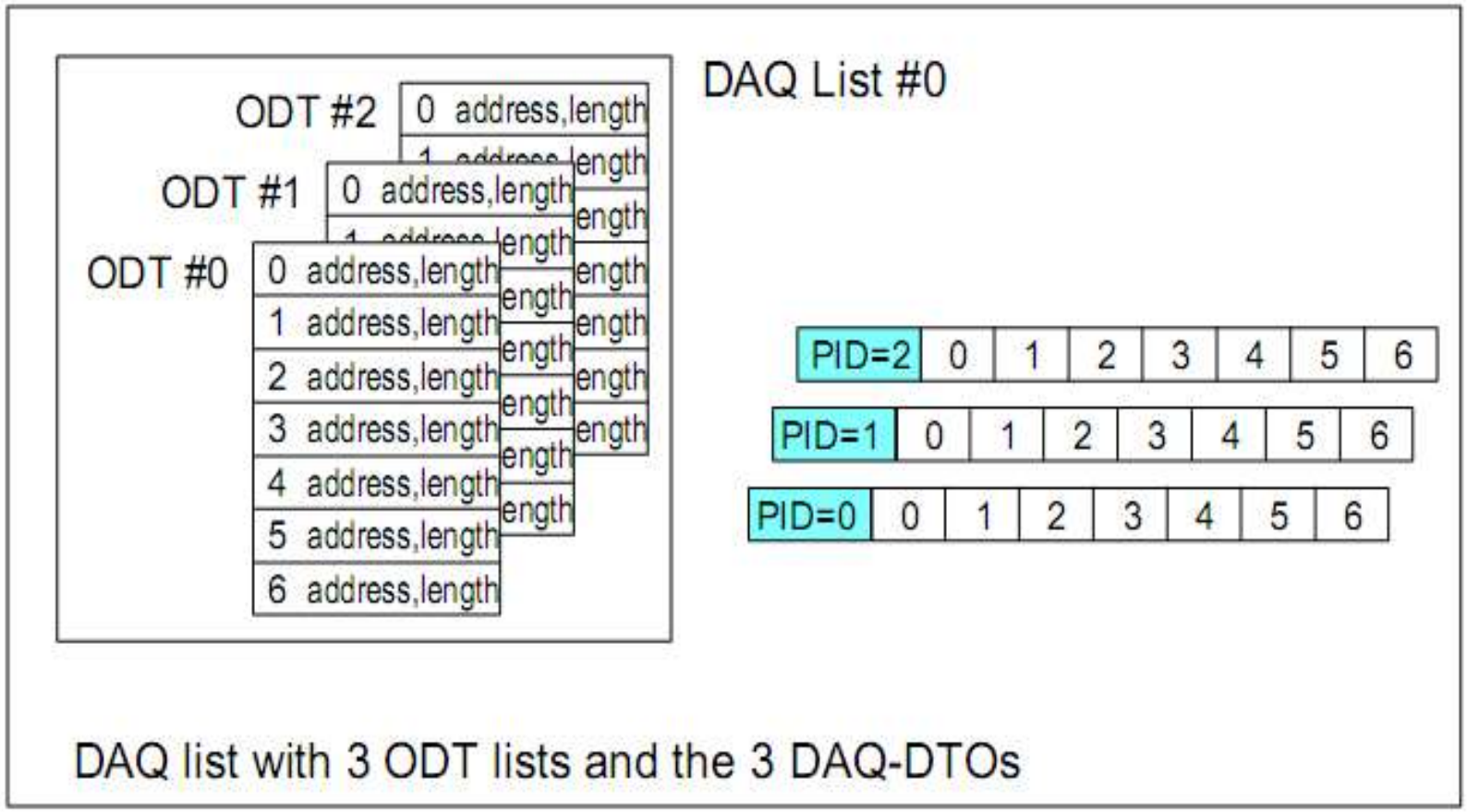
Szinkronizált adatátvitel

- Ugyanaz az alap, mint a CCP-nél



Szinkronizált adatátvitel

- Ugyanúgy DAQ listákba rendezve



Szinkronizált adatírás, küldés

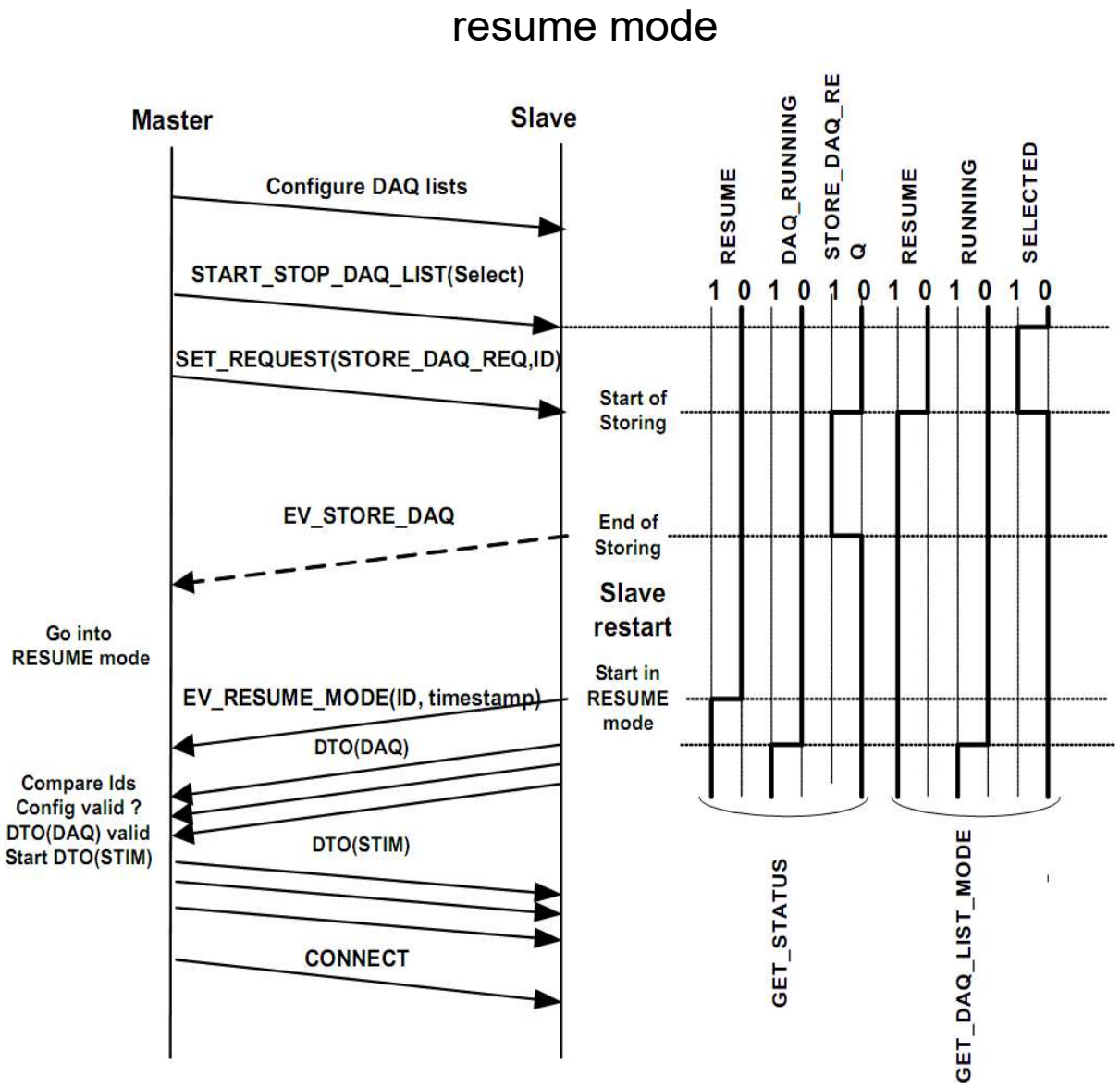
- Synchronous data acquisition (DAQ)
  - Periódikus küldése az DAQ listáknak csakúgy, mint a CCP-nél
- Synchronous data stimulation (STIM)
  - A master (teszter eszkör) elküldi az DAQ listában szereplő ODT-k adatait a slave-nek
  - A következő eventnél a Slave a saját memóriájába frissíti ezeket az adatokat

Event channels, egyéb kis újdonságok

- Konfigurálható számú event channel létezik
  - Mindegyik event-hez hozzá lehet rendelni egy-egy DAQ listát
  - Így az event hatására az automatikusan elküldődik
- Dinamikusan foglalható DAQ listák
    - FREE_DAQ
    - ALLOC_DAQ
    - ALLOC_ODT
    - ALLOC_ODT_ENTRY
  - Előre meghatározott limitek között

Továbbfejlesztett tulajdonságok

- Power-up data transfer (resume mode)
  - A Slave elindulása után azonnal
- Master – Salve szinkronizáció
  - GET_DAQ_CLOCK
- DAQ list prioritás
  - Meg tudják egymást szakítani



Advanced tulajdonságok

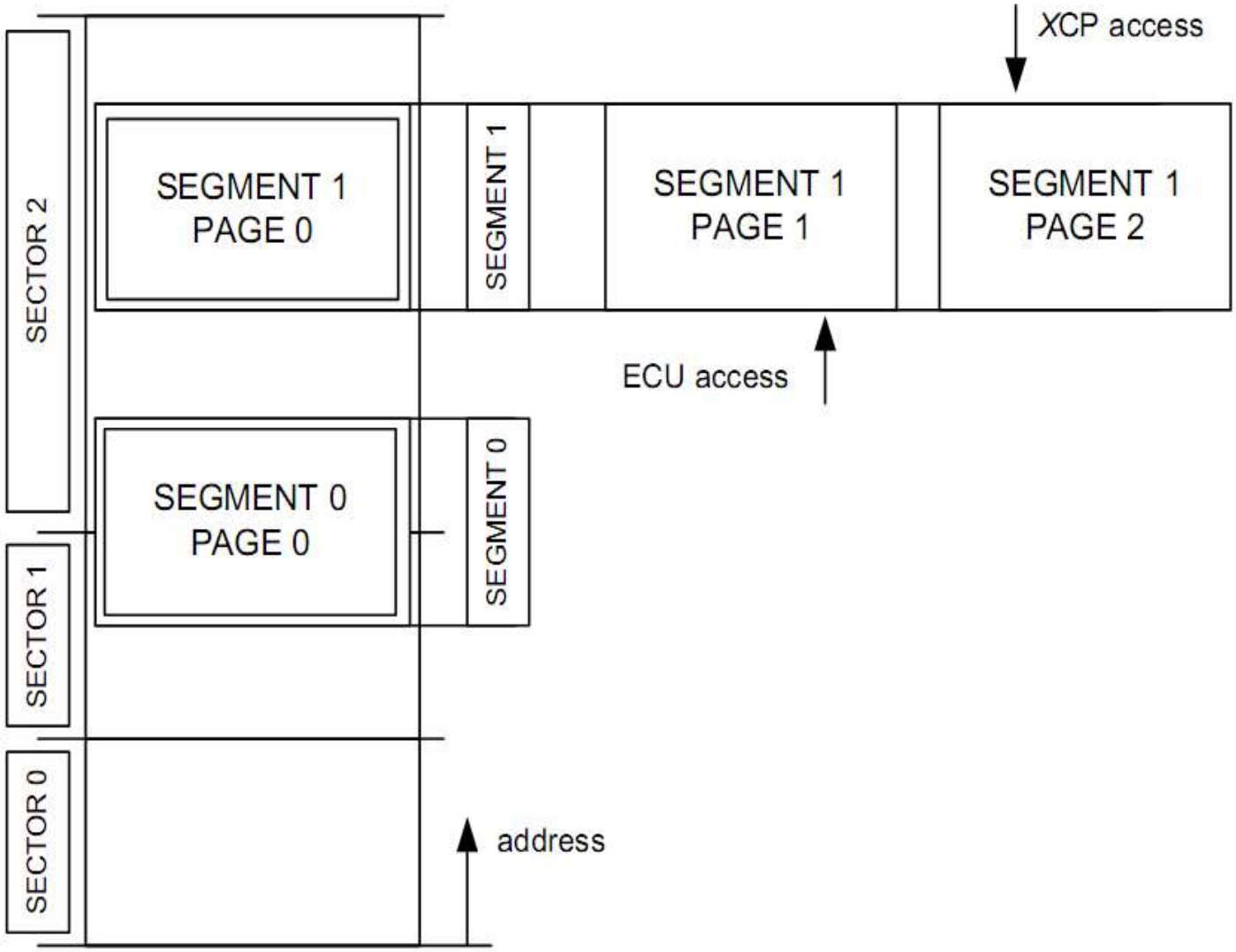
- ODT optimizáció
  - Állit ható, de erősen implementáció függő. Meg lehet adni a támogatott másolási méreteket (csak optimalizáció)
- Bit szintű stimuláció
  - Külön egyes bitek is állíthatóak
  - Nem kell először kiolvasni és utána

Online kalibráció

- Sector
  - Fizikai kialakítás leírása (Flash alapú vezérlők esetében fontos az újraprogramozás és Flash tartalom változtatásnál)
- Segment
  - Logikai információ határ, ami megmondja, hogy a kalibrációs memória hol található meg a memóriában
  - Page: Egy Segmentnek több lapja (Page-e is lehet)
  - Ugyanaz az adat tartalom más tulajdonsággal: érték read/write property stb.
  - Az XCP parancsok mindig egy Page-et használhatnak érhetnek el, és ez az *“active PAGE for XCP access for this SEGMENT”*
  - A Salve control algoritmusai mindig csak egy Page információit az *“active PAGE for ECU access for this SEGMENT”* információit érik el.

Online kalibráció

- Sector, Segment, Page

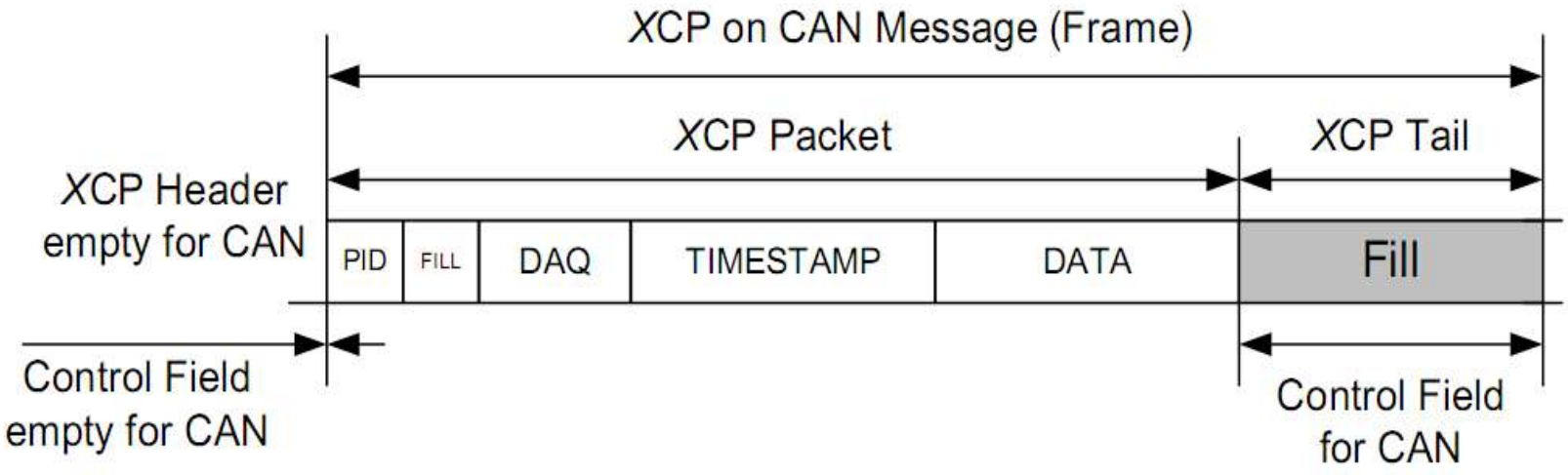


Flash programozás

- Sector
  - Hasonlóan a kalibrációhoz a fizikai kialakítás leírása
- Programozási folyamat
  - Programozás előtti adminisztráció
  - Programozás
  - Programozás utáni adminisztráció

XCP CAN felett

- Címzés
  - Úgy mint a CCP-nél külön CTO, DTO CAN ID
  - Leíró file-ban tárolható
- Üzenet szerkezet
  - Az XCP üzenet hossza (DLC) mindig 8 byte, a Fill erre egészíti ki a keretet
  - A maximum csomag hossz a CTO, DTO csomagokra is 8



XCP UDP és TCP felett

- Címzés
  - Normál TCP / UDP socket kommunikáció
- Üzenet szerkezet
  - LEN: az XCP csomag hossza
  - CTR: számláló a csomag kiesés, vesztés detektálására