PDO mapping is one of the essential features of CANopen: it describes which individual process variables in the data field of a PDO are transmitted. CiA 301 requires a dedicated re-mapping procedure.

The Process Data Object (PDO) service allows exchanging one or several process variables in one single CAN message. The PDO mapping parameter describes which objects in the CANopen object dictionary are transmitted by the sender. The PDO receiver uses also a PDO mapping parameter, which specifies where to store the received process data in the CANopen object dictionary. The PDO mapping parameter of the transmitter and the sender may use different pointers (16-bit index and 8-bit sub-index) depending on the CANopen profile.

In some simple devices, the user does not have the possibility to configure the PDO mapping parameters. This is called static PDO mapping (take it or leave it). More sophisticated devices provide variable PDO mapping. This means the system designer can re-configure the default PDO mapping or generate new PDOs. Normally, this is done in the NMT pre-operational state, when the PDOs are disabled. Of course, the user can also reconfigure the PDO mapping in the NMT operational state, but then it is necessary to avoid inconsistencies in the PDO mapping on the producer and the consumer side. To avoid this, the PDO must not be produced until the entire reconfiguration is finished.

The CiA 301 application layer specification requires a dedicated re-mapping procedure:

- “Destroy” the TPDO by setting the valid bit to 1\_b of sub-index 00\_h of the TPDO communication parameter.
- Disable PDO mapping by setting the sub-index 00\_h of the PDO mapping parameter to 00\_h.
- Modify PDO mapping by changing the values of the corresponding sub-indices of the PDO mapping parameters.
- Enable PDO mapping by setting the sub-index 00\_h to the number mapped process data.
- “Create” a TPDO by setting the valid bit to 0\_b of sub-index 01\_h of the TPDO communication parameter.

If the CANopen device detects that the index and sub-index of the mapped object does not exist or the object cannot be mapped during step 3, the CANopen device responds with the SDO abort transfer service (abort code: 0602 0000\_h or 0604 0041\_h). If the CANopen device detects that the RPDO mapping is not valid or not possible during step 4, the CANopen device responds with the SDO abort transfer service (abort code: 0602 0000\_h or 0604 0042\_h). This is tested in the CANopen conformance test.

**Figure 1:** Example of PDO mapping for CiA 401 devices (eight digital values and one 16-bit-analog value) – application-specific TPDO mapping (left object dictionary) and corresponding application-specific RPDO mapping (right object dictionary)
Normally, the corresponding RPDO mappings need to be re-configured too. This should be done before the TPDO is enabled again. At least between step 4 and 5, all related RPDO mappings should be reconfigured, if necessary. Of course, the user can also first re-map all RPDO mappings (either between step 1 and step 2 or between step 2 and step 3).

By the way, the re-mapping in NMT operational state is called dynamic PDO mapping (e.g. in the CANopen feature list of CiA’s Product Guide). If the CANopen device is correctly implemented, the RPDO re-mapping follows the same procedure as the TPDO re-mapping: “Destroy” the RPDO, disable the RPDO mapping, modify the RPDO mapping, enable the RPDO mapping, and finally “create” the RPDO. Of course, this should be done for all TPDO-corresponding RPDOs, if the TPDO is sent in multi- or broadcast.

If the CANopen device receives a PDO that has more data bytes than the number of mapped data bytes (length), then the CANopen device uses the first data bytes up to the length and sends an EMCY message. If a CANopen device receives a PDO with less data bytes than the number of mapped data bytes (length), then the CANopen device sends an EMCY message with the error code 8210h.

If the TPDO or RPDO re-mapping procedure is not followed, the CANopen device should not change the mapping entries and abort the SDO write service. The device should do this, for example, if the PDO has not been “destroyed” or the mapping has not been disabled. Note: The system designer is responsible for the consistency of the TPDO and the RPDO mapping parameters.