SYS TEC ELECTRONIC

Dynamisches Objektverzeichnis

Manual

Auflage Dezember 2014

Systemhaus für verteilte Automatisierung

Im Buch verwendete Bezeichnungen für Erzeugnisse, die zugleich ein eingetragenes Warenzeichen darstellen, wurden nicht besonders gekennzeichnet. Das Fehlen der © Markierung ist demzufolge nicht gleichbedeutend mit der Tatsache, daß die Bezeichnung als freier Warenname gilt. Ebensowenig kann anhand der verwendeten Bezeichnung auf eventuell vorliegende Patente oder einen Gebrauchsmusterschutz geschlossen werden.

Die Informationen in diesem Handbuch wurden sorgfältig überprüft und können als zutreffend angenommen werden. Dennoch sei ausdrücklich darauf verwiesen, daß die Firma SYS TEC electronic GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf den Gebrauch oder den Inhalt dieses Handbuches zurückzuführen sind. Die in diesem Handbuch enthaltenen Angaben können ohne vorherige Ankündigung geändert werden. Die Firma SYS TEC electronic GmbH geht damit keinerlei Verpflichtungen ein.

Ferner sei ausdrücklich darauf verwiesen, dass SYS TEC electronic GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf falschen Gebrauch oder falschen Einsatz der Hard- bzw. Software zurückzuführen sind. Ebenso können ohne vorherige Ankündigung Layout oder Design der Hardware geändert werden. SYS TEC electronic GmbH geht damit keinerlei Verpflichtungen ein.

Kontakt	**Direkt**	**Ihr Lokaler Distributor**
Adresse:	SYS TEC electronic GmbH Am Windrad 2 D-08468 Heinsdorfergrund GERMANY	Sie finden eine Liste unserer Distributoren unter http://www.systec-electronic.com/distributors
Angebots-Hotline:	+49 (0) 3765 / 38600-2110 info@systec-electronic.com	
Technische Hotline:	+49 (0) 3765 / 38600-2140 support@systec-electronic.com	
Fax:	+49 (0) 3765 / 38600-4100	
Webseite:	http://www.systec-electronic.com	

3. Auflage Dezember 2014

Inhalt

Abbildungs- und Tabellenverzeichnis

Abkürzungen

PDO	Process Data Object
OD	Object Dictionary, Objektverzeichnis
DCF	Device Configuration File
PI	Process Image
ro	Read only, read permission
rw	Read write permission
SDO	Service Data Object
OpenPCS	Programmiertool der Fa. Infoteam zur Programmierung von Steuerungen in einer Programmiersprache nach IEC61131-3-

1 Einleitung

Mit dem Modul CcmDyn.c kann ein vorhandenes statisches Objektverzeichnis um PDOs und Prozessvariablen ergänzt werden. Diese Erweiterung des Objektverzeichnisses wird als dynamisches Objektverzeichnis bezeichnet.
Eine andere Anwendung für das Modul ist das Konfigurieren eines Objektverzeichnisses. Die zu ergänzenden Objekteinträge als auch die Defaultwerte von Einträgen werden in komprimierter Form nach dem in CiA DS-302 beschriebenen „Concise configuration storage“-Format hinterlegt.

2 Grundlagen

Das Objektverzeichnis eines CANopen-Gerätes besteht aus einem statischen Teil und kann um einen dynamischen Teil ergänzt werden. Der statische Teil ist fest in der Firmware definiert und beinhaltet alle grundlegenden Kommunikationsobjekte (SDO, Heartbeat, Emergency, PDOs). Der dynamische Teil des Objektverzeichnisses ist optional und kann dazu verwendet werden, weitere Prozessvariablen und PDOs hinzuzufügen. Bei einem dynamischen Objektverzeichnis werden Objekteinträge erst zur Laufzeit einmalig angelegt, anschließend kann auf sie in derselben Art und Weise zugegriffen werden wie auf statisch angelegte Objekteinträge.

Die Grundlage für das Erstellen eines dynamischen Objektverzeichnisses bildet die bei einer Netzwerkkonfiguration erstellte DCF-Datei (**D**evice **C**onfiguration **F**ile) für einen CANopen-Knoten. Die DCF-Datei beinhaltet die konfigurierten Gerätedaten und wird mit Hilfe eines CANopen-Konfigurators ausgehend von einer EDS-Datei (**E**lectronic **D**ata **S**heet) erzeugt, alternativ kann sie auch manuell für eine fest definierte Konfiguration erstellt werden.

Bild 1 zeigt das Schema zur Einbindung von Prozessvariablen für CANopen am Beispiel einer mit OpenPCS programmierbaren SPS. Das Modul CcmDyn.c wird innerhalb der CANopen-Anbindung für die SPSen der Fa. SYS TEC electronic angewendet.

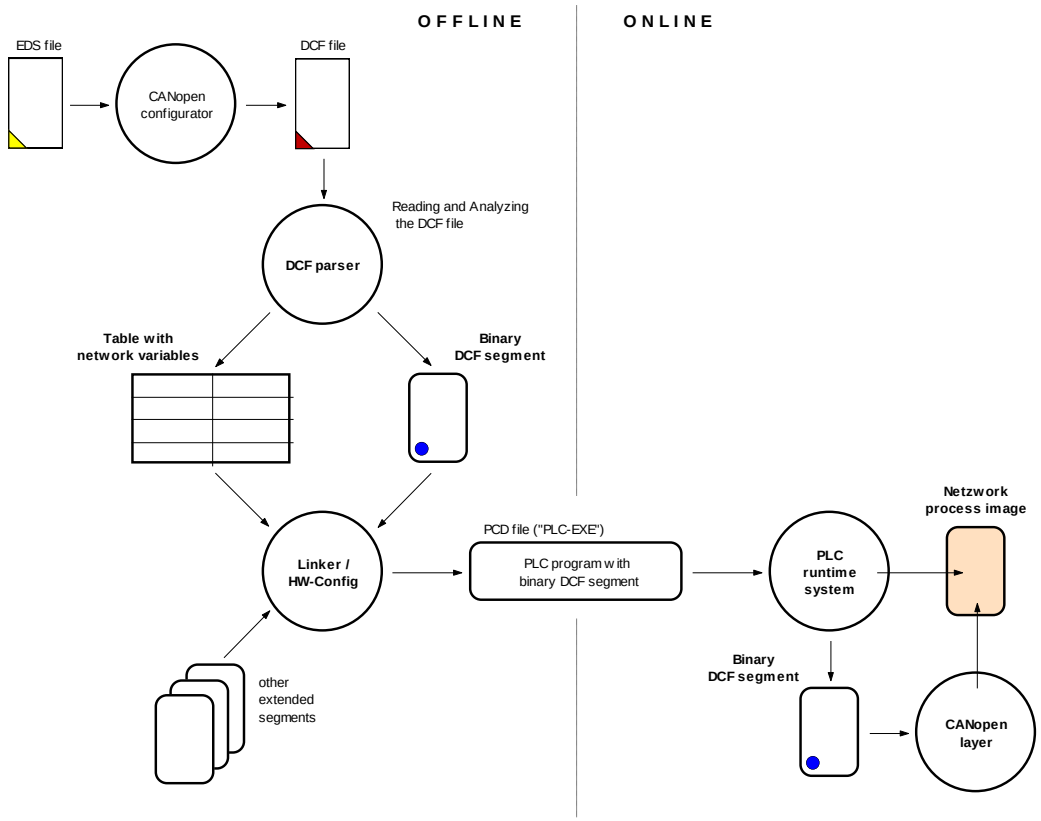

Bild 1: Einbindung von dynamischen Prozessvariablen für CANopen

Für das Einlesen und Auswerten der DCF-Informationen ist auf der PC-Seite der "DCF-Parser" (Dcf2Bin.dll) zuständig. Er liest zunächst alle relevanten Abschnitte der DCF-Datei ein und baut damit eine abstrakte, binäre Datenstruktur auf. Diese dient als Grundlage zum Auflösen der Referenzen auf Netzwerkvariablen und zur Generierung von CANopen-Steuerinformationen in Form des binären DCF-Segments. Beim Download des SPS-Programms gelangt dann das binäre DCF-Segment auf die Steuerung und wird dort an die CANopen-Schicht übergeben.

Weitere Informations über die Programmierschnittstelle des "DCF-Parser" finden sie in der beiliegenden Datei "readme.txt".

Mit Hilfe des binären DCF-Segments können sowohl neue Objekteinträge angelegt als auch bereits vorhandene (statische) Einträge neu parametriert werden. Dies schließt auch die Rekonfiguration existierender PDO-Objekte ein. *Bild 2* verdeutlicht dieses Verfahren.

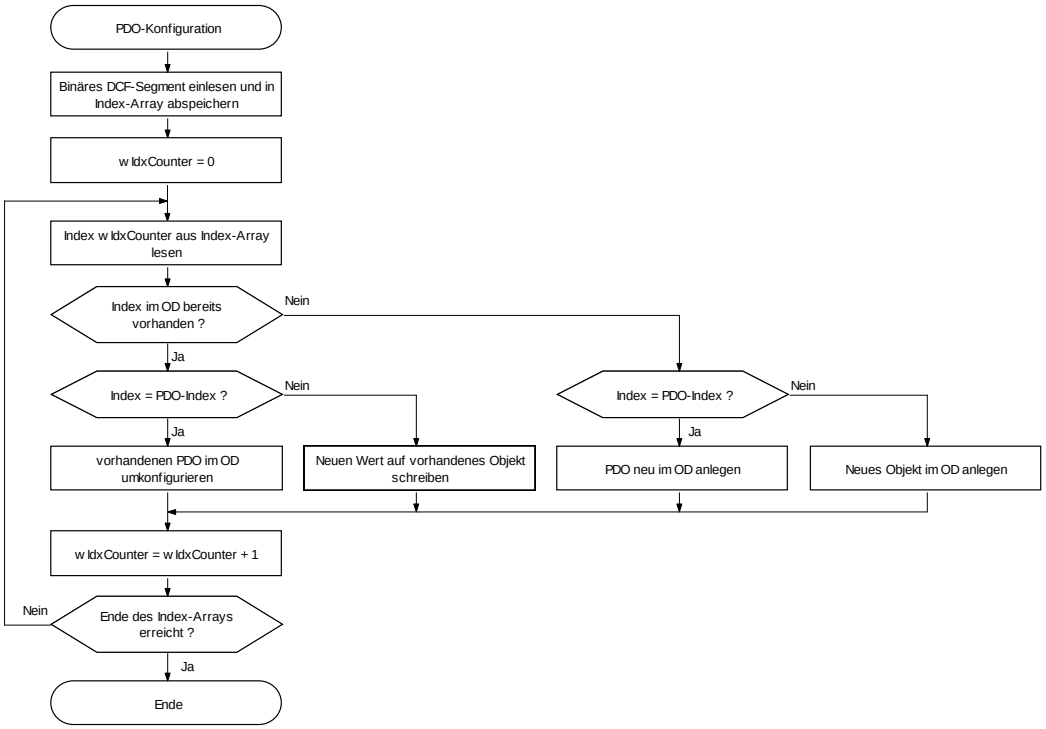

Bild 2: PDO-Rekonfiguration beim Anlegen des dynamischen ODs

Hinweis: PDOs, die durch das dynamische Objektverzeichnis rekonfiguriert wurden, erhalten bei dessen Abbau (dynamisches OD wird zerstört) wieder ihre ursprünglichen Default-Werte des statischen Objektverzeichnisses.

2.1 Aufbau und Format des binären DCF-Segmentes

Das binäre DCF-Segment selbst bildet eine Container-Struktur zur Aufnahme weiterer eingebetteter Segmente. Es enthält als Untermenge (Index-Segment) die im CiA Standard DS-302 definierte Struktur einer verkürzten DCF-Datei ("Concise configuration storage"). *Bild 3* zeigt den Aufbau des Containers für das binäre DCF-Segmet.

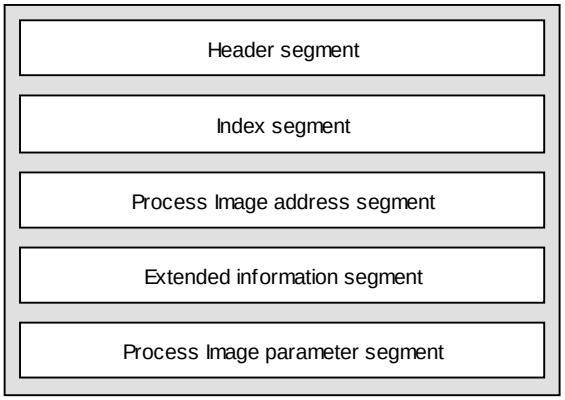

Bild 3: Aufbau des Containers für das binäre DCF-Segment

Das binäre DCF-Segment wird ebenfalls für das Beschreiben von Netzwerkvariablen für SPSen verwendet. Für diese Aufgabe werden die Segmente „Process Image address segment“ und „ProcessImage parameter segment“ benötigt. Die folgenden Abschnitte beschreiben ebenfalls diese Segmente, wenn auch diese Segmente für das Erstellen eines dynamischen Objektverzeichnisses entfallen können. Die Einträge im Headersegment können dann auf Null gesetzt werden.

2.1.1 Header-Segment

Das Header-Segment beinhaltet die Längen der einzelnen Segmente sowie die Gesamtlänge des Container-Segments. Den Segmentaufbau beschreibt *Bild 4*.

DWORD	Size of binary DCF segment
WORD	Version
WORD	Number of segments
DWORD	Offset Index-Segment
DWORD	Size Index-Segment
DWORD	Offset Process Image Adress-Segment
DWORD	Size Process Image Adress-Segment
DWORD	Offset Extended Info-Segment
DWORD	Size Extended Info-Segment
DWORD	Offset Process Image Parameter-Segment
DWORD	Size Process Image Parameter-Segment

Bild 4: Aufbau des Header-Segments

Die Gesamtlänge ist die Summe aller Längen der Einzelsegmente (inkl. Länge des Header-Segments).

Sämtliche Werte werden im Little-Endian-Format (LSB first) hinterlegt. Für nicht verwendete, optionale Segmente ist der Offsetwert und die Länge des Segments auf Null zu setzen.

2.1.2 Index-Segment

Der Aufbau des Index-Segments entspricht der Definition für eine verkürzte DCF-Datei ("Concise configuration storage") gemäß CiA Standard DS-302. Den Segmentaufbau beschreibt *Bild 5*.

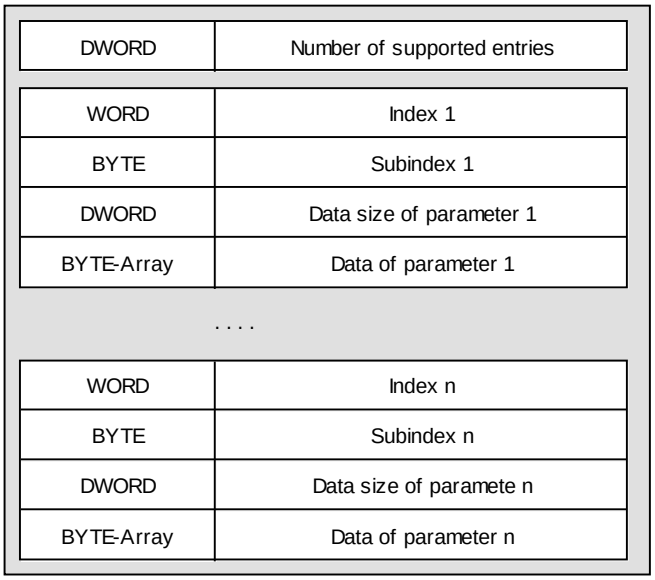

Bild 5: Aufbau des Index-Segments lt. CiA DS-302

Hinweis
Zu jedem PDO-Kommunikationsparameter-Objekt muss ein PDO-Mappingparameter-Objekt existieren, z.B. 0x1400 setzt Objekt 0x1600 voraus und 0x1801 setzt Objekt 0x1A01 voraus.
Objekte im Adressbereich 0xA000-0xAFFF werden stets als Process-Variablen mit folgenden Attributen angelegt:

NUMERIC_VALUE, READ_PERMISSION, WRITE_PERMISSION, PDO_MAPP_PERMISSION

Existiert das ProcessImageAddress-Segment und ein Prozessimage (Netzwerkprozessabbild), so werden die Variablen innerhalb dieses Speicherbereiches angeordnet.

2.1.3 Process Image Adress-Segment

Das Adress-Segment ist ein optionaler Bestandteil des binären DCF-Segments und muss nicht zwingend in der Container-Struktur vorhanden sein. Das Segment wird jedoch dann benötigt, wenn Objekteinträge im Indexbereich 0xA000-0xAFFF innerhalb eines Netzwerkprozessabbildes abzulegen sind.

Hintergrund:
Die Offsets der einzelnen Variablen im Netzwerkprozessabbild ergeben sich aus ihrem jeweiligen Index und Subindex. Abs. 6 beschreibt den im CiA Standard DS-302 spezifizierten Algorithmus zur Berechnung der Variablenoffsets. Die nochmalige explizite Angabe der Variablenoffsets im Adress-Segment bildet daher eine redundante Datenstruktur.

Beim Einlesen und Auswerten der DCF-Informationen auf der PC-Seite durch den "DCF-Parser" (Dcf2Bin.dll) erfolgt bereits die notwendige Adressberechnung anhand von Index und Subindex gemäß Standard. Diese Adressen werden auf der PC-Seite vom Linker benutzt, um Referenzen auf Netzwerkvariablen auflösen zu können. Damit ist aber auch die Lage und Anordnung der Variablen im Netzwerkprozessabbild fixiert. Die CANopen-Schicht der SPS ist damit gezwungen, ebenfalls die für das SPS-Programm benutzten Adressen zu verwenden (sogar unabhängig davon, ob diese normkonform berechnen wurden oder nicht). Andernfalls können Inkonsistenzen bei der Nutzung des Netzwerkprozessabbildes zwischen SPS-Programm/LZS und CANopen-Schicht entstehen.

Um eine einheitliche Nutzung des Netzwerkprozessabbildes zwischen SPS-Programm/LZS und CANopen-Schicht zu gewährleisten, wird auf der SPS-Seite beim Aufbau des dynamischen Objektverzeichnis auf das vom "DCF-Parser" generierte Adress-Segment zurückgegriffen, sofern dieses vorhanden ist. Das Adress-Segment ist ein WORD-Feld, das für jede im Netzwerkprozessabbild angelegte Variable einen Adressoffset enthält.

Bei der Nutzung des Programmiersystems OpenPCS ist das Adress-Segment immer vorhanden, da ja die notwendige Adressberechnung in jedem Fall auf der PC-Seite auch zum Auflösen der Referenzen von Netzwerkvariablen notwendig ist. Generell kann die zum Aufbau und zur Verwaltung eines dynamischen Objektverzeichnises notwendige Funktionalität auch unabhängig vom SPS-System genutzt werden, wobei dann auch der Aufbau des binären DCF-Segments durch eine fremde Komponente erfolgt. In diesem Fall kann es dann aber auch möglich sein, dass das Adress-Segment entfällt und die notwendige Adressberechnung auf dem Zielsystem lokal auszuführen ist.

2.1.4 Extended Info-Segment

Das Extended Info-Segment ist ein BYTE-Feld, das für jeden Subindex im Index-Segment die entsprechenden Zugriffsattribute enthält.

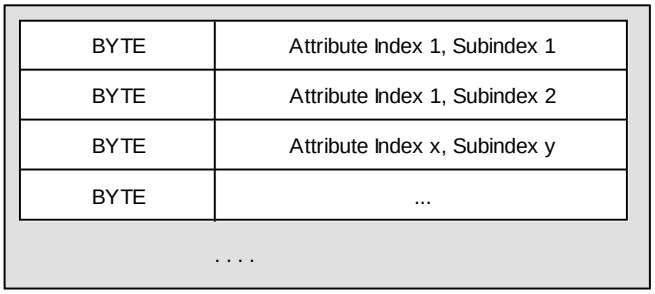

Bild 6: Aufbau des Extended Info Segments

Die Attribute werden als Bitmaske definiert und können so miteinander kombiniert werden. Im einzelnen können das sein:

Attribute	Wert	Beschreibung
BOOLEAN	0x01	Der Objekteintrag wird als BOOLEAN definiert.
VISIBLE_STRING	0x08	Der Objekteintrag wird als VISIBLE_STRING definiert. Die Daten werden byteweise in den Objekteintrag kopiert.
NUMERIC_VALUE	0x10	Der Objekteintrag wird als NUMERIC_VALUE definiert. Die Daten werden je nach Target LSB-first oder MSB-first abgelegt.
READ_PERMISSION	0x20	Das Lesen des Objekteintrages ist erlaubt.
WRITE_PERMISSION	0x40	Das Schreiben des Objekteintrages ist erlaubt.
PDO_MAPP_PERMISSION	0x80	Das Mappen des Objekteintrages in ein PDO ist erlaubt.

Tabelle 1: Zugriffsattribute für dynamische Objekteinträge

Wird das Attribute BOOLEAN, VISIBLE_STRING oder NUMERIC_VALUE nicht definiert, so wird der Objekteintrag als DOMAIN interpretiert. Das hat zur Folge, dass die Daten aus dem Index-Segment des jeweiligen Objekteintrages byteweise kopiert werden.

Beispiele:

Wert	Beschreibung
0x70	Numerischer Wert mit Lese- und Schreiberlaubnis
0x60	Domain mit Lese- und Schreiberlaubnis
0xF0	Numerischer Wert mit Lese- und Schreiberlaubnis, mappbar in ein PDO
0x30	Numerischer Wert mit nur Leseerlaubnis

2.1.5 Process Image Parameter-Segment

Dieses Segment beinhaltet Informationen über ein vorhandenes ProcessImage (Netzwerkprozessabbild). Den Segmentaufbau beschreibt *Bild 7*.

DWORD	Complete size of Process Image
DWORD	Offset of input area
DWORD	Size of input area
DWORD	Offset of output area
DWORD	Size of output area

Bild 7: Aufbau des Process Image Parameter-Segmentes

Für das Anlegen von dynamischen Objekteinträgen wird dieses Segment nicht benötigt.

3 Schnittstelle zur Applikation

Für das Erstellen eines dynamischen Objektverzeichnisses muß ein binäres DCF-Segment, wie in *Abs. 2.1* beschrieben, in das Target geladen werden.[1] Diese Datenstruktur wird der Funktion **DynBuildOd** als Parameter übergeben. Die Funktion fügt dann dem vorhandenen statischen Objektverzeichnis die dynamischen Objekteinträge hinzu und hinterlegt Daten in den Objekteinträgen.

Voraussetzung für das Hinzufügen von Objekteinträgen ist eine dynamische Speicherverwaltung. Mit Hilfe von Makros (COP_MALLOC, COP_FREE) können plattform-spezifische Funktionen wie *malloc* oder *free* eingebunden werden. Die Anpassung der Makros erfolgt im File target.h.

Durch das dynamische OD werden unter Umständen weitere PDOs definiert. Bereits existierende PDOs im statischen Objektverzeichnis werden vor dem Anlegen der neuen PDO ungültig gesetzt. Dadurch gehen die Parameter für das PDO-Linking verloren. Nach dem Erstellen des Objektverzeichnisses sind dieses PDOs erneut zu verknüpfen.

Das Ablegen von Objektdaten der dynamisch angelegten Objekteinträge in einem nichtflüchtigen Speicher (z.B. EEPROM) wird nicht durch das Modul CcmStore unterstützt.

Für dynamisch erstellte Indexeinträge können keine Callback-Funktionen im Zusammenhang mit Zugriffen per SDO oder mit Hilfe der API-Funktionen *ObdWriteEntry()* und *ObdReadEntry()* bzw. *CcmWriteObject()* und *CcmReadObject()* verwendet werden.

Das Erstellen des dynamischen Objektverzeichnisses sollte nach dem Aufruf der Funktion *CcmConnectToNet()* erfolgen. Dadurch werden Objekteinträge angelegt und PDO’s hinzugefügt.

Ein wiederholtes Laden des binären DCF-Segmentes in das Target und das Erstellen des dynamischen Objektverzeichnisses setzt voraus, dass das bisherige dynamische Objektverzeichnis abgemeldet und der dynamisch allokierte Speicher freigegeben wurde. Diese Aufgabe übernimmt die Funktion **DynDestroyOd**.

Der folgende Quelltext-Auszug verdeutlicht die Vorgehensweise. Die vollständigen Sourcen inkl. des binären DCF-Segmentes sind im File ex_dynod.c zu finden.

[1] Das Erstellen und das Laden des binären DCF-Segmentes obliegt dem Anwender.

```
// ============================================================
// segment container (contains the build up rules of all
// objects which should be created dynamicaly and/or which should
// be updated with a new value)
// ============================================================
static CONST BYTE ROM abSegmentContainerRom_l[] =
{
    ...
};

// memory for the Process Image (see PI-Param-Segement)
static WORD MEM awProcessImage_l[0x0020];

void main (void)
{
tCopKernel  Ret = kCopSuccessful;
tProcessImageDscrpt PIDscrpt;

    // init the CANopen Stack
    Ret = CcmInitCANopen (&CcmInitParam_g, kCcmFirstInstance);
    ...

    // set CANopen from state INITIALIZATION to PRE-OPERATIONAL
    Ret = CcmConnectToNet ();
    ...

    // fill out the Process Image Descriptor
    PIDscrpt.m_pbProcessImage = (BYTE FAR*) &awProcessImage_l[0];
    PIDscrpt.m_dwProcessImageSize = sizeof (awProcessImage_l);

    // build the dynamic part of the OD
    Ret = DynBuildOd ((BYTE FAR*) &abSegmentContainerRom_l[0],
        &PIDscrpt);
    ...

    while (APP_RUN_FLAG())
    {
        // main prozess function for the CANopen stack
        CcmProcess ();
        ...
    }

    // destroy a previously build dynamic OD
    Ret = DynDestroyOd ();
    ...
}
```

4 Konfiguration des CANopen-Stacks

Je nach Anwendung des dynamischen Objektverzeichnisses sind in der Konfigurationsdatei *CopCfg.h* Konstanten zu setzen:

OBD_USE_DYNAMIC_OD
Mit der Datei CcmDyn.c können Objekte dynamisch (d.h. zur Laufzeit) zum aktuellen Objektverzeichnis hinzugefügt werden. Um diese Eigenschaft nutzen zu können muss dieses Define auf TRUE gesetzt werden.

Gewählte Einstellung: TRUE
Wertebereich: FALSE, TRUE
Anwendungsbereich: CANopen –Sourcecode mit dynamischen OD

OBD_USER_OD, OBD_USE_VARIABLE_SUBINDEX_TAB
Das dynamische Objektverzeichnis wird als USER_OD verarbeitet. Es werden für das Hinzufügen von Objekteinträgen variable Subindex-Tabellen benötigt. Die beiden Werte sind daher auf TRUE zu setzen.

Gewählte Einstellung: TRUE
Wertebereich: FALSE, TRUE
Anwendungsbereich: CANopen –Sourcecode mit dynamischen OD

PDO_USE_BIT_MAPPING
Für das Hinzufügen und Mappen von Objekteinträgen vom Typ BOOLEAN ist der Wert auf TRUE zu setzen. Anderenfalls ist das Mappen von Bitwerten nicht möglich.

Gewählte Einstellung: TRUE
Wertebereich: FALSE, TRUE
Anwendungsbereich: CANopen –Sourcecode

PDO_GRANULARITY
Um Objekteinträge vom Typ BOOLEAN an beliebiger Stelle im PDO mappen zu können, ist dieser Wert auf 64 zu stellen. Anderenfalls sind max. 8 Bit mappbar.

Gewählte Einstellung: 64
Wertebereich: 8, 64
Anwendungsbereich: CANopen –Sourcecode

5 Beschreibung der Applikationsfunktionen

5.1 Funktion DynBuildOd

Syntax:

```
#include <CcmDyn.h>
tCopKernel PUBLIC DynBuildOd ( CCM_DECL_INSTANCE_HDL_
           BYTE FAR* pbDcfSegCont_p,
           tProcessImageDscrpt* pProcessImageDscrpt_p);
```

Parameter:

CCM_DEC_INSTANCE_HDL_: Instanz-Handle

pbDcfSegCont_p:	Pointer auf das binäre DCF-Segment
pProcessImageDscrpt_p:	Pointer auf eine Datenstruktur zur Beschreibung des Process Images

Return:

kCopSuccessful	0x00	Die Funktion wurde ohne Fehler ausgeführt.
kCopIllegalInstance	0x01	Die parametrierte Instanz existiert nicht.
kCopDynNoMemory	0xA0	Der benötigte Speicherplatz ist nicht vorhanden.
kCopDynInvalidConfig	0xA1	Die verwendeten Datenstrukturen sind ungültig.
kCopCobIllegalCanId	0x23	Der verwendete CAN-Identifier ist ungültig.

kCopCobAllreadyExist	0x21	Es wurde versucht, ein Kommunikationsobjekt anzulege, das bereits existiert.
kCopCobNoFreeEntry	0x20	Das Anlegen eines COB ist gescheitert, da keine freier Eintrag in der TX-COB bzw. RX-COB- Tabelle vorhanden ist. Die Tabellen sind gegebenfalls zu vergrößern.
kCopObdIndexNotExist	0x31	Es wurde versucht, einen Objekteintrag zu beschreiben, der nicht existiert.
kCopObdSubindexNotExist	0x32	Es wurde versucht, einen Objekteintrag zu beschreiben, der nicht existiert.
kCopPdoErrorMapp	0x78	Beim Mappen eines Objekteintrages wurde ein Fehler erkannt.

Beschreibung:

Die Funktion ergänzt aus einem binären DCF-Segment dynamische Objekteinträge. Bereits existierende Objekteinträge werden mit Daten aus dem binären DCF-Segment geladen, falls diese als Einträge im DCF-Segment vorhanden sind.
Wird die Funktion benutzt, um dynamisch ergänzte Objekteinträge innerhalb eines ProcessImages (Netzwerkprozessabbild) anzulegen, so ist die Startadresse (m_pbProcessImage) und max. zur Verfügung stehende Größe des Prozessabbildes (m_dwProcessImageSize) mit Hilfe der Struktur *tProcessImageDscrpt* zu übermitteln.

```
typedef struct
{
    // Address and size of process image
    BYTE FAR* m_pbProcessImage;     // IN-Parameter
    DWORD     m_dwProcessImageSize; // IN/OUT-Parameter

    // Offset and size of inputs
    DWORD     m_dwDynPIOffsetIn;    // OUT-Parameter
    DWORD     m_dwDynPISizeIn;      // OUT-Parameter

    // Offset and size of outputs
    DWORD     m_dwDynPIOffsetOut;   // OUT-Parameter
    DWORD     m_dwDynPISizeOut;     // OUT-Parameter

} tProcessImageDscrpt;
```

Ist das Segment ProcessImage-Param im binären DCF-Segment vorhanden, so hinterlegt die Funktion die dort enthaltenen Parameter in der Struktur *tProcessImageDscrpt*.

5.2 Funktion DynDestroyOd

Syntax:

```
#include <CcmDyn.h>
tCopKernel PUBLIC DynDestroyOd (  CCM_DECL_INSTANCE_HDL);
```

Parameter:

CCM_DEC_INSTANCE_HDL: Instanz-Handle

Return:

kCopSuccessful	0x00	Die Funktion wurde ohne Fehler ausgeführt.
kCopIllegalInstance	0x01	Die parametrierte Instanz existiert nicht.

Beschreibung:

Die Funktion löscht dynamisch ergänzte Objekteinträge und setzt das PDO-Modul zurück.
Bevor das dynamische Objektverzeichnis gelöscht werden kann, sind die Kommunikationsobjekte für die PDOs zurückzusetzen. Dazu wird die NMT-Event-Funktion des PDO-Moduls mit dem Ereignis kNmtEvPreResetCommunication aufgerufen.
Dann werden das dynamische Objektverzeichnis und die angelegten PDO-Tabellen gelöscht und die Defaultwerte für die Kommunikationsparameter der PDOs im statischen Objektverzeichnis gesetzt.
Im Anschluss wird das PDO-Modul in den Zustand PreOperational gesteuert.

6 Indexeinträge und Vergabe von Variablenoffsets

Der CiA Standard DS-405 definiert die in *Tabelle 2* aufgeführten Indexbereiche für Netzwerkvariablen zur Verwendung durch die IEC 61131-3. Durch das Programmiersystem OpenPCS wird jedoch nur ein Teil der möglichen Variablentypen unterstützt.

Entsprechend dem CiA Standard DS-302 erfolgt die Vergabe der Offsets für Netzwerkvariablen durch den CANopen-Konfigurator. Für jede Variablenart (Datentyp und Zugriffsrichtung) wird ein eigenes Subsegment angelegt. Diese Subsegmente können optional durch die EDS-Datei in Lage und Größe innerhalb des Netzwerkprozessabbildes fest definiert werden (PIOffset=, MaxCnt=, Range=). Enthält die EDS-Datei keine entsprechenden Einträge, wird der Startoffset eines Subsegments implizit als Null angenommen, die Größe des Subsegments passt sich dynamisch der Anzahl deklarierter Variablen an. Für Input- und Output-Variablen werden getrennte Prozessabbilder angelegt.

Jedes Subsegment ist als Array der jeweiligen Variablenart (BYTE, WORD, DWORD, ...) zu interpretieren. Der Subindex, unter dem eine Variable in der DCF-Datei angelegt wurde, widerspiegelt deren Array-Index. Hierbei ist jedoch zu beachten, dass der erste Subindex für eine Variablendefinition den Wert 1 hat, während in Hochsprachen (wie z.B. C) das erste Array-Element den Index 0 besitzt. Um den realen Array-Index einer Variablen im Prozessabbild zu erhalten, ist der jeweilige Subindex der Variable um den Wert 1 zu dekrementieren. Der Subindex gibt dabei das der Variablen zugeordnete Array-Element an und nicht deren absoluten Offset innerhalb des Prozessabbildes. So wird beispielsweise durch Subindex=6 für eine Byte-Variable der Offset 5 im Prozessabbild adressiert (=(6-1)*sizeof(BYTE)), mit Subindex=3 einer DWORD-Variable dagegen aber der Offset 8 (=(3-1)*sizeof(DWORD)).

Datenrichtung	Start-Index	Data Type	Zugriffsart	Verwendung in OpenPCS
Input	A000H	Integer8	ro	x
	A040H	Unsigned8	ro	x
	A080H	Boolean	ro	-
	A0C0H	Integer16	ro	x
	A100H	Unsigned16	ro	x
	A140H	Integer24	ro	-
	A180H	Unsigned24	ro	-
	A1C0H	Integer32	ro	x
	A200H	Unsigned32	ro	x
	A240H	Float (32)	ro	(x)
	A280H	Unsigned40	ro	-
	A2C0H	Integer40	ro	-
	A300H	Unsigned48	ro	-
	A340H	Integer48	ro	-
	A380H	Unsigned56	ro	-
	A3C0H	Integer56	ro	-
	A400H	Integer64	ro	-
	A440H	Unsigned64	ro	-
Output	A480H	Integer8	rw	x
	A4C0H	Unsigned8	rw	x
	A500H	Boolean	rw	-
	A540H	Integer16	rw	x
	A580H	Unsigned16	rw	x
	A5C0H	Integer24	rw	-
	A600H	Unsigned24	rw	-
	A640H	Integer32	rw	x
	A680H	Unsigned32	rw	x
	A6C0H	Float (32)	rw	(x)
	A700H	Unsigned40	rw	-
	A740H	Integer40	rw	-
	A780H	Unsigned48	rw	-
	A7C0H	Integer48	rw	-
	A800H	Unsigned56	rw	-
	A840H	Integer56	rw	-
	A880H	Integer64	rw	-
	A8C0H	Unsigned64	rw	-

Tabelle 2: Indexbereiche für Netzwerkvariablen

Jeder Indexeintrag kann für den zugehörigen Datentyp (BYTE, WORD, DWORD, ...) ein Array mit bis zu 254 Elementen verwalten. Um weitere Variablen anzulegen, ist der nachfolgende Indexeintrag zu verwenden. So kann beispielsweise der Index 0A4C0H die ersten 254 Byte-Variablen verwalten, ab der 255. Byte-Variable ist der Index 0A4C1H notwendig.

Bei der Vergabe der Subindizes berücksichtigt der CANopen-Konfigurator eventuelle Überlagerungen von Subsegmenten, wobei die Variablenoffsets dann so gewählt werden, dass jeder Variable ein separater Speicherbereich entsprechend ihres Typs zugewiesen wird. Überlagerte Subsegmente liegen z.B. dann vor, wenn die EDS-Datei keine Vorgaben in Form von Startoffsets für die einzelnen Variablenbereiche enthält. In diesem Fall beginnen alle Subsegmente standardmäßig ab Offset Null.

Der CANopen-Konfigurator plaziert alle Variablen des ersten Subsegments am Anfang des Prozessabbildes (sofern nicht alle Subsegmente per EDS-Datei auf Startoffsets größer Null verschoben wurden) und schließt die Variablen der folgenden Subsegmente daran an. Da der Subindex eine indirekte Entsprechung für den Offset der Variablen im Prozessabbild ist, beginnen die Subindizes der Variablen im ersten Subsegments mit dem Wert 1. Der Subindex der Variable aller folgenden Subsegmente (Variablenarten) hängt vom bereits belegten Speicher und damit vom ersten freien Array-Index für den entsprechenden Variablentyp innerhalb des Prozessabbildes ab *(siehe Bild 8)*.

Der nachstehende Auszug aus einer DCF-Datei verdeutlicht zusammen mit *Bild 8* den Zusammenhang zwischen Index/Subindex und Offset im Prozessabbild.

```
[A4C0]
SubNumber=4

[A4C0sub0]
ParameterName=NrOfElements

[A4C0sub1]
ParameterName=IN0_IN7

[A4C0sub2]
ParameterName=IN8_IN15

[A4C0sub3]
ParameterName=IN16_IN23

[A580]
SubNumber=2

[A580sub0]
ParameterName=NrOfElements

[A580sub3]
ParameterName=AIN0
```

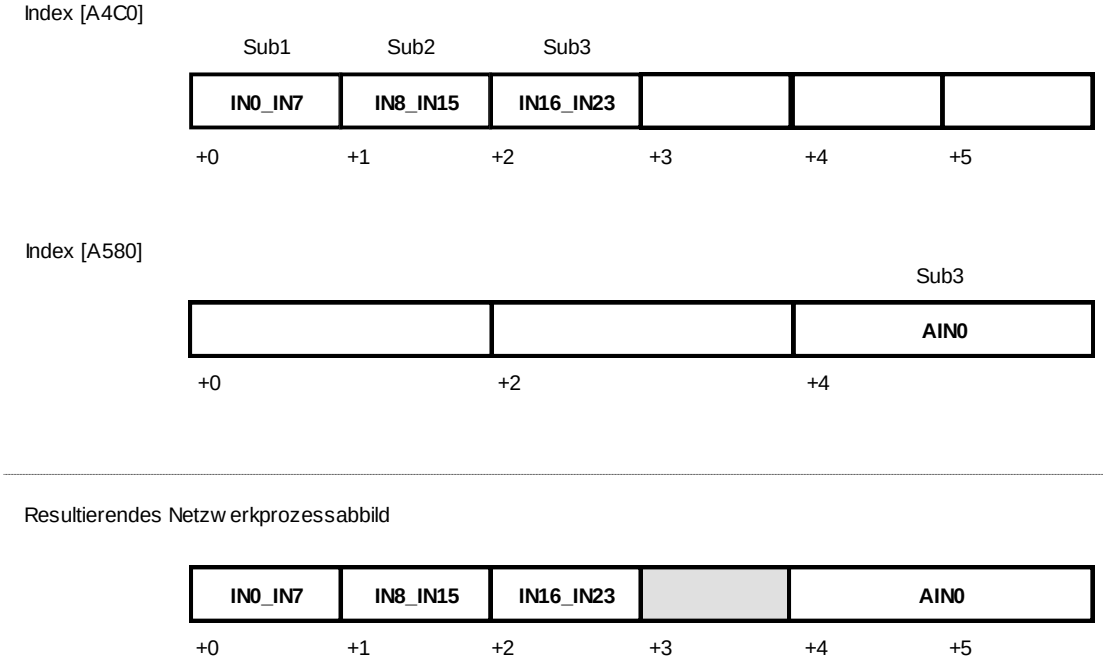

Bild 8: Beispiel für Zusammenhang zwischen Index/Subindex und Variablenoffset

Index

Dokument:	Dynamisches Objektverzeichnis
Dokumentnummer:	L-1087d_3, Auflage Dezember 2014

Wie würden Sie dieses Handbuch verbessern?

Haben Sie in diesem Handbuch Fehler entdeckt? Seite

Eingesandt von:

Kundennummer: _____

Name: _____

Firma: _____

Adresse: _____

Einsenden an:

SYS TEC electronic GmbH
August-Bebel-Str. 29
D-07973 Greiz
GERMANY
Fax : +49 (0) 36 61 / 62 79 99

Veröffentlicht von

SYS TEC ELECTRONIC

Best.-Nr. L-1087d_3
Printed in Germany