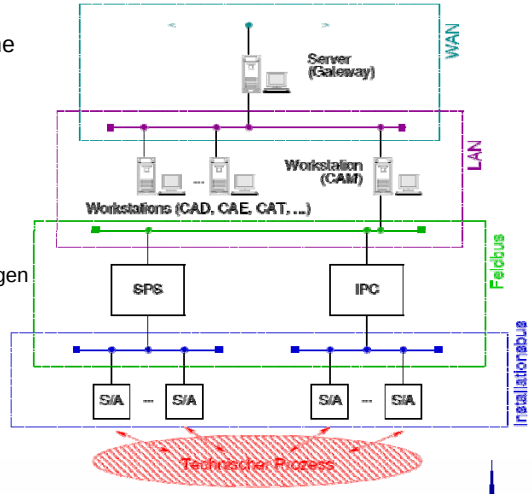
Profibus-DP

CP242-8
CP5613
Programmierung

Wolfgang Kastner, EMail: k@auto.tuwien.ac.at
Institut für Rechnergestützte Automation, TU Wien

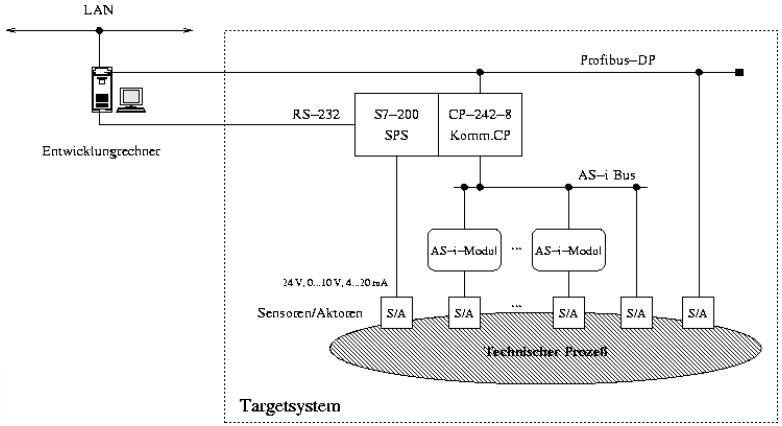
Dezentrale Automatisierungssysteme

- Management und Führungsebene
  Server und Workstations
- Steuer- und Leitebene
  Workstations und PCs
- Automationsebene
  Industrie/Embedded PCs,
  Speicherprogrammierbare Steuerungen
- Feldebene
  Sensoren, Aktuatoren, Controller



Übersicht Verbindungshierarchie

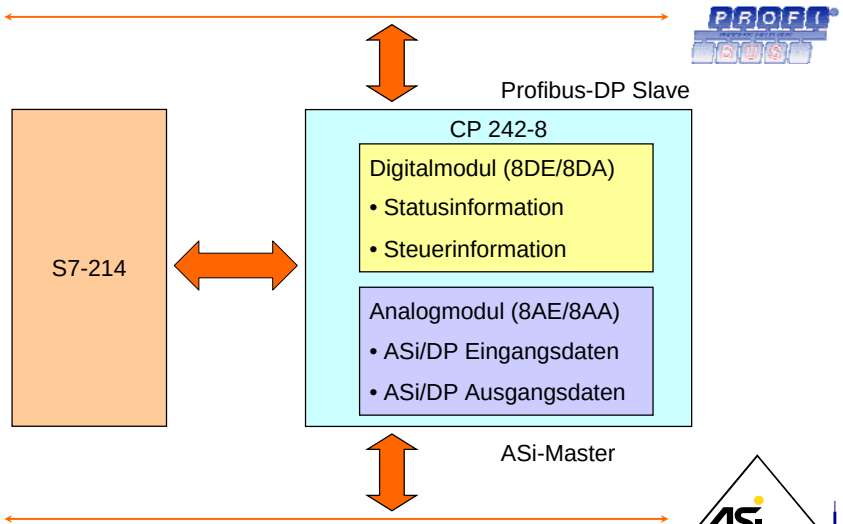
- Profibus-DP Master CP5613
- Siemens S7-214 mit Kommunikationsprozessor CP242-8
- Sensoren und Aktoren



Profibus-DP Geräte

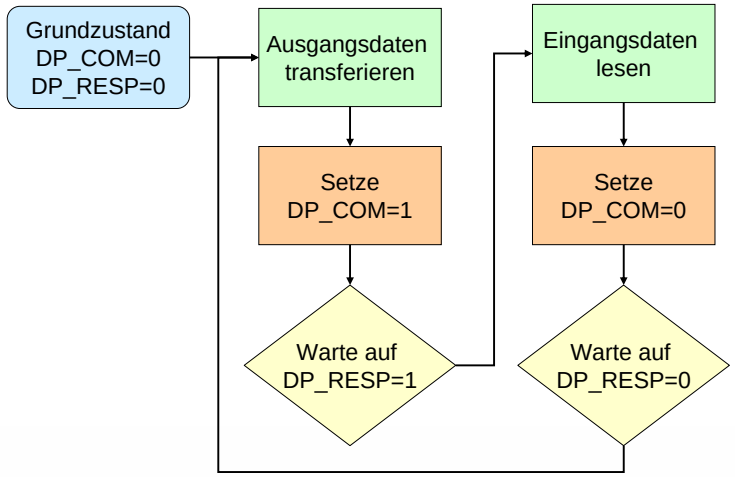
- DP-Master Klasse 1 (DPM1)
  Zentrale Steuerung, die Daten mit dezentralen E/A-Geräten austauscht. Typischerweise eine SPS oder ein Controller.
  In unserem Fall: CP5613 (Adresse 101 - Adresse 110).
- DP-Master Klasse 2 (DPM2)
  Projektierungs-, Überwachungs- oder Engineering-Werkzeug zur Inbetriebnahme von Profibus-DP Geräten
- DP-Slave
  Peripheriegeräte mit direkter Schnittstelle zu den E/A Signalen (z.B. E/A Komponenten, Antriebe, Ventile).
  In unserem Fall: CP242-8 (Adresse 1) und
  Encoder (Adresse 55).

Kommunikationsprozessor CP 242-8



Blockkonsistenter Datentransfer

Softwarehandshake mit DP_COM und DP_RESP



Statusbyte (Digitalmodul, 8DE)

- Aufbau:

0	ASi-R	DP_RESP	DP_CON	DP1	DP0	CP_Ready	ASi-M

ASi-R	0/1	Antwortbit für ASi-Kommando-schnittstelle (für LU nicht relevant)
DP-RESP	0/1	Antwortbit für blockkonsistenten Datentransfer 0…DP-Ausgänge können geschrieben werden 1…DP-Eingänge können gelesen werden
DP-CON	0/1	DP-Master tauscht mit CP 242-8 0…bytekonsistente Daten aus 1…blockkonsistente Daten aus
DP1-DP0	00-11	Status der DP Schnittstelle
CP_READY	0	CP 242-8 ist noch nicht betriebsbereit
	1	CP 242-8 ist betriebsbereit Auswertung der E/A-Daten ist zulässig
ASi-M	0	Geschützter Betrieb
	1	Projektiermodus

- Digitalmodul EB2
- Systemhandbuch
  - CP 242-8: 2-11, 2-12

Steuerbyte (Digitalmodul, 8DA)

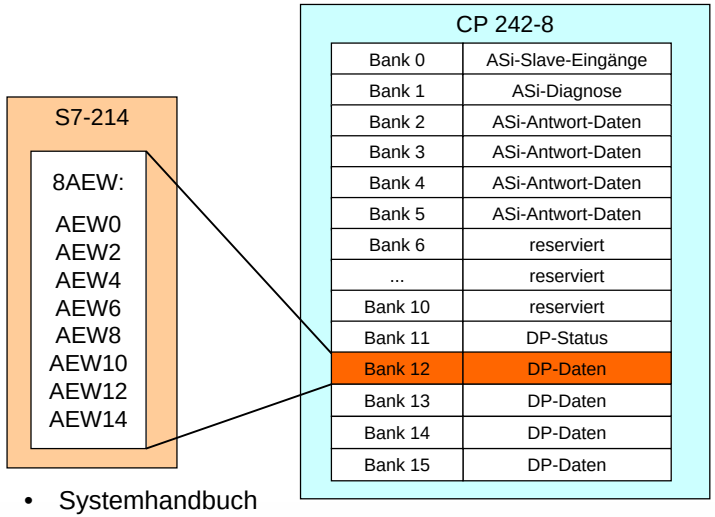
- Aufbau:

PLC_RUN	ASi-C	DP-COM	0	BS3	BS2	BS1	BS0

PLC_RUN	1	Signalisierung an CP 242-8, daß sich die S7-214 im Zustand RUN befindet. WICHTIG: Das Anwenderprogramm muß dieses Bit im Anlauf (SM0.1) auf 1 setzen!
ASI_C	0	Auftragsbit für ASi-Kommandoschnittstelle (für LU nicht relevant)
DP-COM	0/1	Auftragsbit für blockkonsistenten Datentransfer 0…beende Aktualisierung 1…starte Aktualisierung
BS3	0/1	Bankauswahl
BS2	0/1	Bankauswahl
BS1	0	Bankauswahl
BS0	0	Bankauswahl

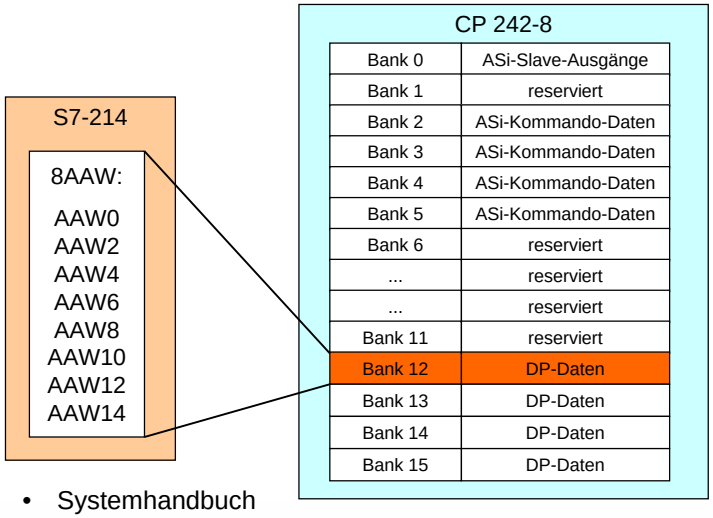
- Digitalmodul AB2
- Systemhandbuch
  - CP 242-8: 2-13, 2-17

Bankauswahl 8AEW



- Systemhandbuch
  - CP 242-8: 2-18

Bankauswahl 8AAW

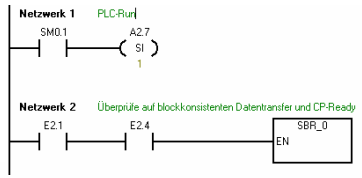


- Systemhandbuch
  - CP 242-8: 2-20

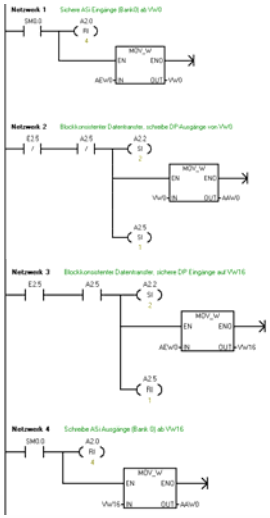
Beispiel

Aufgabe:

DP Eingangsdaten auf ASi Ausgänge
ASi Eingänge auf DP Ausgänge



OB1



SBR0

DP Master

Anlauf und Betriebsphase:

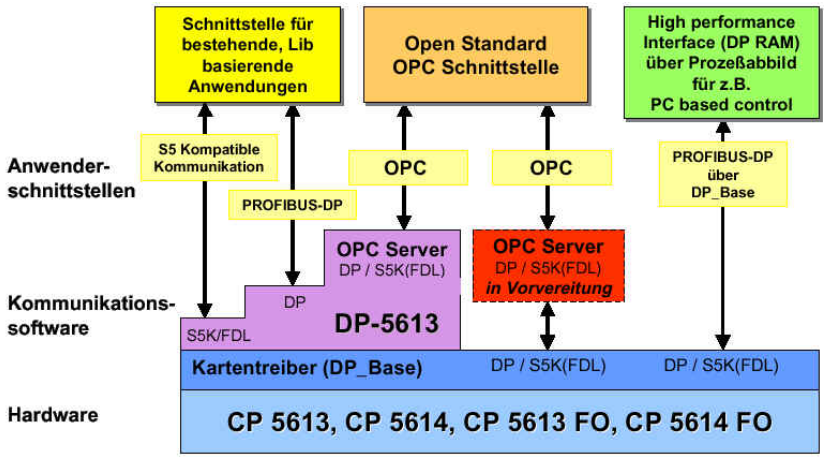
- Initialisierung des DP Systems
- Parametrierung und Konfiguration der DP Slaves
- zyklischer Datentransfer zu den DP Slaves
- Überwachung der DP Slaves
- Bereitstellen von Diagnoseinformation

Zustände:

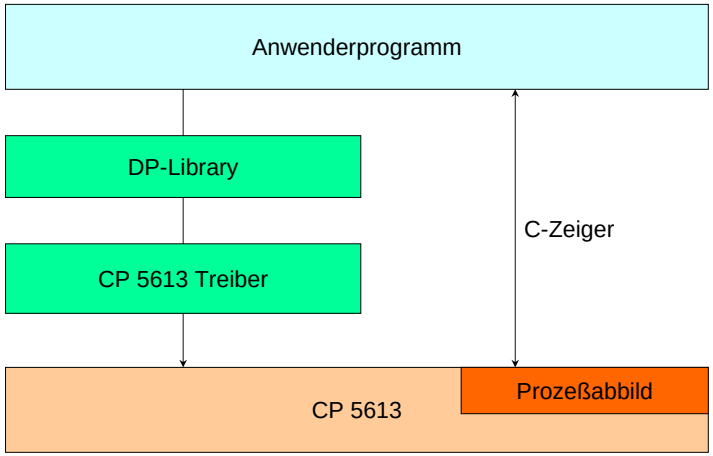
- OFFLINE: Startzustand (kein Datenaustausch zwischen Master und Slaves)
- STOP: DP-Master Klasse 2 kann Diagnose-Information auslesen
- CLEAR: Konfigurierung bzw. Parametrierung der DP Slaves, senden des ersten Ausgabe-Telegramms
- OPERATE: zyklischer Datenaustausch mit den Slaves

OFFLINE -> STOP -> CLEAR -> OPERATE

CP 5613



DP Anwenderprogramm



Anlaufphase

1. Laden der Firmware und der Datenbasis in den CP5613

```
DP_start_cp((DPR_STRING *)CP_NAME, pDatenbasis, &ErrStruct)
```

2. Anmelden eines DP Anwenderprogramms, Vergabe eines User-Handle

```
DP_open((DPR_STRING *)CP_NAME, &DPUserHandle, &ErrStruct)
```

3. Zeiger auf Prozeßabbild anfordern

```
DP_get_pointer(DPUserHandle, TIMEOUT, &pDPRam, &ErrStruct)
```

4. Auslesen DP Zustand und Wechsel in gewünschten Zustand

```
pDPRam->info_watch.master_info.USIF_state
DP_set_mode(DPUserHandle, DP_STOP, &ErrStruct)
DP_set_mode(DPUserHandle, DP_CLEAR, &ErrStruct)
DP_set_mode(DPUserHandle, DP_OPERATE, &ErrStruct)
```


Betriebsphase (Polling)

1. Lesen eines Datenbytes (BYTE ch)

```
/* lock input area */
pDPRam->ctr.D_lock_in_slave_adr = Slave_Nr;

/* get process data*/
memcpy(&ch,
       (void*)(&(pDPRam->pi.slave_in[Slave_Nr].data[0])), 1);

/* unlock input area */
pDPRam->ctr.D_lock_in_slave_adr = DPR_DP_UNLOCK;
```

2. Schreiben eines Datenbytes (BYTE ch)

```
/* transfer to process data*/
memcpy((void*)&(pDPRam->pi.slave_out[Slave_Nr].data[0]),
       &ch, 1);
/* activate writing */
pDPRam->ctr.D_out_slave_adr = Slave_Nr;
```

Betriebsphase (HW-Events)

1. Anlegen einer Semaphore

```
DP_init_sema_object(DPUserHandle, DP_OBJECT_TYPE_INPUT_CHANGE,
                    &DPSemaHandle, &ErrStruct);
```

2. Rücksetzen der Event Maske

```
pDPRam->ef.input[Slave_Nr].req_mask =
   DPR_DATA_INT_CLEAR_AND_UNMASK;
```

3. Warten auf Zustandsänderung

```
WaitForSingleObject((HANDLE) DPSemaHandle, TIMEOUT);
```

4. Lesen eines Datenbytes (char ch)

```
/* lock input area */
pDPRam->ctr.D_lock_in_slave_adr = Slave_Nr;

/* get process data*/
memcpy(&ch,
       (void*)(&(pDPRam->pi.slave_in[Slave_Nr].data[0])), 1);

/* unlock input area */
pDPRam->ctr.D_lock_in_slave_adr = DPR_DP_UNLOCK;
```


Nachlaufphase

1. Auslesen DP Zustand und Wechsel in gewünschten Zustand

```
pDPRam->info_watch.master_info.USIF_state
DP_set_mode(DPUserHandle, DP_CLEAR, &ErrStruct)
DP_set_mode(DPUserHandle, DP_STOP, &ErrStruct)
DP_set_mode(DPUserHandle, DP_OFFLINE, &ErrStruct)
```

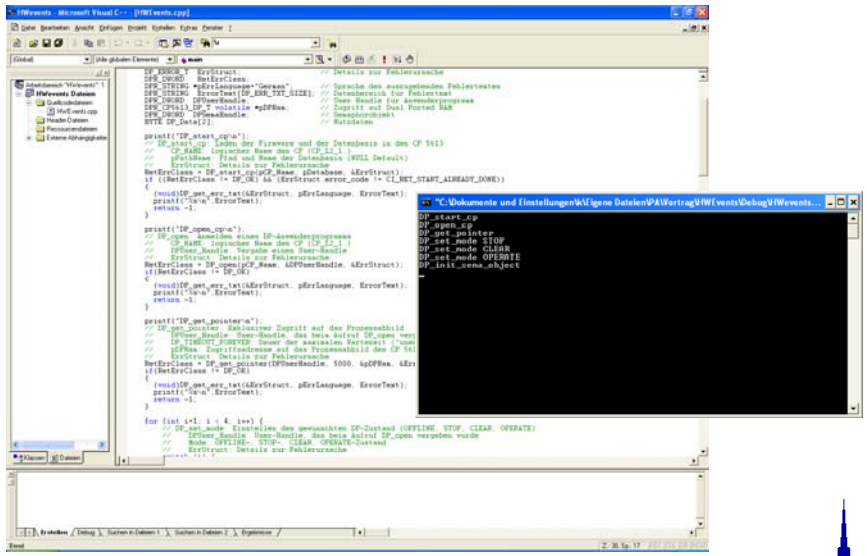
2. Zeiger auf Prozeßabbild rückgeben

```
DP_release_pointer(DPUserHandle, &ErrStruct)
```

3. Abmelden eines DP Anwenderprogramms

```
DP_close(DPUserHandle, &ErrStruct)
```


Demoprogramm



Multithreading

```
#include <windows.h>

HANDLE CreateThread(
      // pointer to security attributes
      LPSECURITY_ATTRIBUTES lpThreadAttributes,
      // initial thread stack size
      WORD dwStackSize,
      // pointer to thread function
      LPTHREAD_START_ROUTINE lpStartAddress,
      // argument for new thread
      LPVOID lpParameter,
      // creation flags
      DWORD dwCreationFlags,
      // pointer to receive thread ID
      LPDWORD lpThreadId
   );

BOOL CloseHandle( HANDLE hObject )
```

Tips und Tricks

- Multithreading:

```
DWORD WINAPI MasterThread(void* foo) { … }
main() {
     …
     hThread = CreateThread(0, 0, MasterThread, (void*)0, 0, &id);
     …
     WaitForSingleObject(hThread, TIMEOUT);
     CloseHandle(hThread);
}
```

- Includes:

```
"dp_5613.h"      // CP Kommandos
"5613_ret.h"     // Return Werte
<windows.h>      // Multithreading, Sleep
```

- Library:

```
dp_base.lib
```

- Encoder liefert Werte im Big-Endian Format!

Weiterführende Information

- S7-200 CPU 214 Handbuch
- CP 242-8 Handbuch
- CP 5613 Handbuch
- Tutoren
- Forum
- **http://www.tilab.tuwien.ac.at/sa**
- Profibus-DP Simulator!

Viel Erfolg!