DBonDP

Standard-Software für Simatic S7

Inhaltsverzeichnis

1. Lieferumfang....2
2. Funktion 3
3. Aufbau und Arbeitsweise....5
4. Installation und Benutzung des Archivs....11
5. Copyrights....11
5. Beschreibung des Beispielprogramms....12
Liste der verfügbaren Tutorials und Software-Produkte von Heisch Automation 22
Dienstleistungen und Anlagen rund um die Simatic S5 / S7 24

Die aktuellen Produkte finden Sie unter http://sites.inka.de/heisch

Stand 04.05.2009 HW

Heisch Automation
Ingenieurbüro für Industrieautomatisierung
Büro: Ostring 15 D 76829 Landau / Pfalz
Postadresse: Im Vorderen Großthal 4 D 76857 Albersweiler /Pfalz
Tel: +49 6341 890117 Fax: +49 6341 890118
e-mail: hwauto@heisch-automation.de www.heisch-automation.de

1. Lieferumfang

Die Bibliothek DBonDP bestehen aus

1. dieser **Beschreibung**

2. dem **FB 120 „DBonDP“**

3. Beispiel- und Demoprogrammen

bestehend aus:
- OB 1 als Hauptprogramm für die Beispielprogramme
- FC 9 Generierung Zeitimpulse und Log0,Log1
- FC 120 Beispielprogramm: Simulation lokales und Partner-AG, deutsche Dokumentation
- FC 120 Beispielprogramm: Simulation lokales und Partner-AG, englische Dokumentation
- FB 121 Simulations-Version des FB120

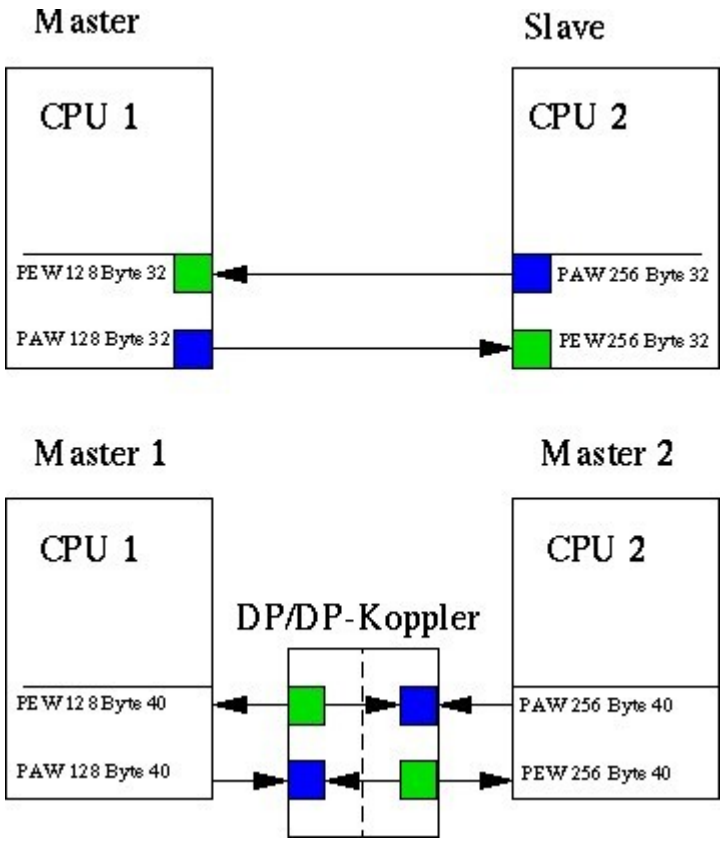
- DB120 Instanz-DB für FB 121 (lokal-AG)
- DB121 Instanz-DB für FB 121 (Partner-AG)
- DB1120 Quell-DB lokales AG
- DB1121 Ziel-DB im Partner AG

Die Demo-Version enthält nicht den FB120. Er kann gekauft werden.

2. Funktion

Bidirektionaler Austausch von Datenbausteinen zwischen 2 Simatic-S7-CPUs über E/A-Kopplung.

Geeignet für die Datenübertragung zwischen Maschinen, die über eine E/A-Kopplung verfügen, aber nicht direkt Telegramme austauschen können.



Dies sind im Wesentlichen:

- CPUs mit Profibus-Schnittstelle, die am gleichen Profibus-Netz hängen und über DP-Kopplung im Master-Slave-Verfahren Daten austauschen können. (eine CPU ist Master, 2. CPU ist Slave)
- CPUs, die über DP/DP-Koppler Daten austauschen.
- Beliebige E/A-Kopplungen, die die nachfolgenden Voraussetzungen erfüllen.

Voraussetzungen:

- Dezentrale Peripherie, Datenkonsistenz blockweise einstellbar. (*)
- minimale Blockgröße 8 Byte (*)
- Koppelbereich: PEW / PAW einer Maschine müssen im Nummernbereich übereinstimmen. (z.B: PEW 256, PAW 256) (*)
- Die Peripherie-Adressen der beiden Maschinen können unterschiedlich sein. (*)

(*) Diese Voraussetzung wird von allen DP-CPUs sowie den DP/DP-Kopplern erfüllt.

Bemerkung:

Der Kern der Kopplung, der FB120, wurde 2001 für einen Kunden erstellt. Es werden DBs zwischen 8 Maschinen ausgetauscht, pro Maschine-Maschine-Verbindung jeweils mehrere DBs gemultiplext.

Probleme sind bisher keine bekannt. Die Kopplung ist also praxiserprobt.

Nachträgliche Änderungen für dieses Paket:
- DP-Blockgröße jetzt flexibel von 8 bis 224 Byte.
- Fehlermeldungen verbessert.

Die Demo-Programme wurden nachträglich zusätzlich erstellt, der Simulations-FB 121 dient zum Programmtest und für die ausgelieferte Demo-Version.

3. Aufbau und Arbeitsweise

Funktion:
Der FB überträgt DBs in ein Partner-Gerät und nutzt hierzu Dezentrale Peripherie, z.B über DP/DP-Koppler oder über E/A verbundene CPUs.

Voraussetzung:
Für den Adressbereich, über den gekoppelt wird, ist die PEW- und PAW-Adresse gleich, ebenso die Länge der Bereiche der beiden Richtungen. Dieser Bereich muss mindestens 4 Worte breit sein.

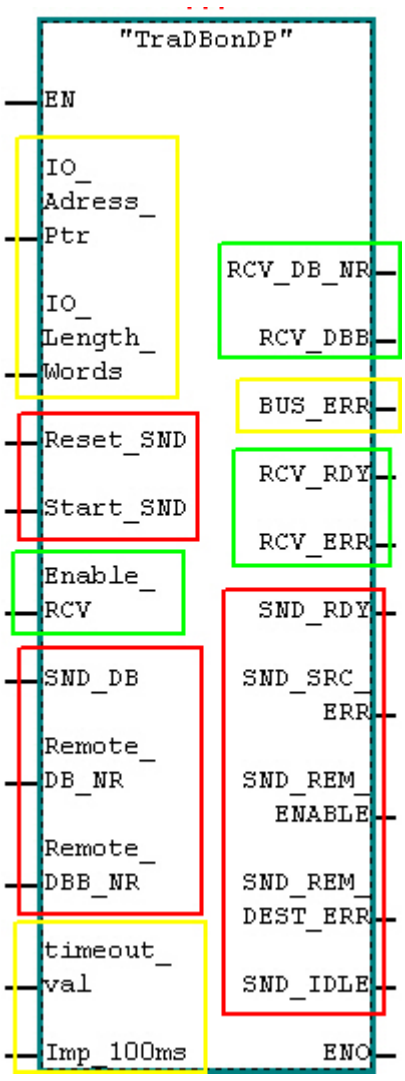
Funktionsweise:
Der FB arbeitet Voll-Duplex, d.h. er bearbeitet gleichzeitig Sende- und Empfangsaufträge.
Realisiert: Write-Aktiv, Receive-Passiv. Fetch-Aufträge sind direkt nicht möglich.

Datenkonsistenz der übertragenen DBs: ("EA_Laenge_in_Wort" -1) Worte.
Der FB wird in beiden AGs eingesetzt, er arbeitet symmetrisch.

Die gelben Eingänge dienen der Herstellung und Überwachung der Kopplung. Sie sind immer zu beschalten
Der gelbe Ausgang „BUS_ERR“ ist sinnvoll.

Die roten Ein-und-Ausgänge gehören zum Sende-Teil, sie sind obsolet, falls der FB nur empfängt.

Die grünen Ein-und-Ausgänge gehören zum Empfangs-Teil, sie sind obsolet, falls der FB nur sendet.



Die Eingangsparameter:

Definition des IO-Bereichs:

IO_Adress_Ptr	z.B. PEW256 Start des IO-Bereichs
IO_Length_Words	Länge des EA-Bereiche in Worten. Zulässig 4 .. 112

Sendekommandos

Reset_SND	Sendeauftrag rücksetzen (Normal nicht notwendig, kann aber z.B. benutzt werden, wenn der zu sendende DB sich während der Sendung verändert.)
Start_SND	Start Senden : Wenn SND_IDLE, ist nur ein Impuls notwendig Soll immer der gleiche DB zyklisch gesendet werden, kann der Eingang mit logisch 1 beschaltet werden.
SND_DB	ANY-Pointer auf den DB-Bereich in der lokalen Maschine, der gesendet werden soll (nur DB zulässig). Aus dem ANY-Pointer wird die Anzahl der zu sendenden Bytes ermittelt.
Remote_DB_NR	Nummer des Ziel-DBs in dem Partner-AG
Remote_DBB_NR	Nummer des Ziel-DBB in dem Ziel-DB des Partner-AGs

Empfangskommando

Enable_RCV	Freigabe Datenempfang Dieses Signal wird beim Partner am Ausgang "SND_REM_ENABLE" gemeldet. Da das erste Telegramm nur den Datenkopf enthält, kann der Empfänger mit diesem Eingang den Transfer noch rechtzeitig selektiv steuern.

Timeout-Erkennung

timeout_val	Zeitwert für Timeout (100ms-Einheiten)
Imp_100ms	Zeitimpuls für Timeout-Erkennung (100ms-Takt) (keinen Blinktakt benutzen, es wird ein Impuls benötigt.)

Die Ausgangsparameter:

Allgemein

BUS_ERR Kommunikationsfehler, Details: siehe Bits im Instanz-DB, DIW 34

Empfang

RCV_DB_NR Nummer des DB, der empfangen wird

RCV_DBB Nummer des Offsets des DBB, ab dem empfangen wird.
"RCV_DB_NR" und "RCV_DBB" beinhalten den Wert, der sende-seitig in den Parametern Remote_DB_NR und Remote_DBB_NR eingetragen wurde. Dies zeigt der empfangenden CPU an, welche Daten aktuell übertragen werden.
Beide Werte werden zu 0, wenn aktuell kein Empfang

RCV_RDY Empfangs-Status : Fertig
Dieses Signal kommt als Impuls, wenn der Empfang regulär beendet wurde.

RCV_ERR Empfangs-Status : Fehler

Senden

SND_RDY Sende-Status : Fertig.
Dieses Bit wird gesetzt, wenn vom Partner der fehlerfreie Empfang quittiert wurde. (Siehe Partner:RCV_RDY)
Das Bit wird rückgesetzt, wenn lokal ein neuer Sende-Auftrag gestartet wurde.

SND_SRC_ERR Sendefehler: Der Quell-DB auf der lokalen Maschine konnte nicht gelesen werden. Existiert der DB? Länge ?

SND_REM_ENABLE Sende-Status: Der Partner erlaubt den Empfang

SND_REM_DEST_ERR Sende-Status: Fehlermeldung vom Partner:
Der Ziel-DB fehlt oder ist zu kurz.(Remote-DB-Bereich nicht vorhanden)

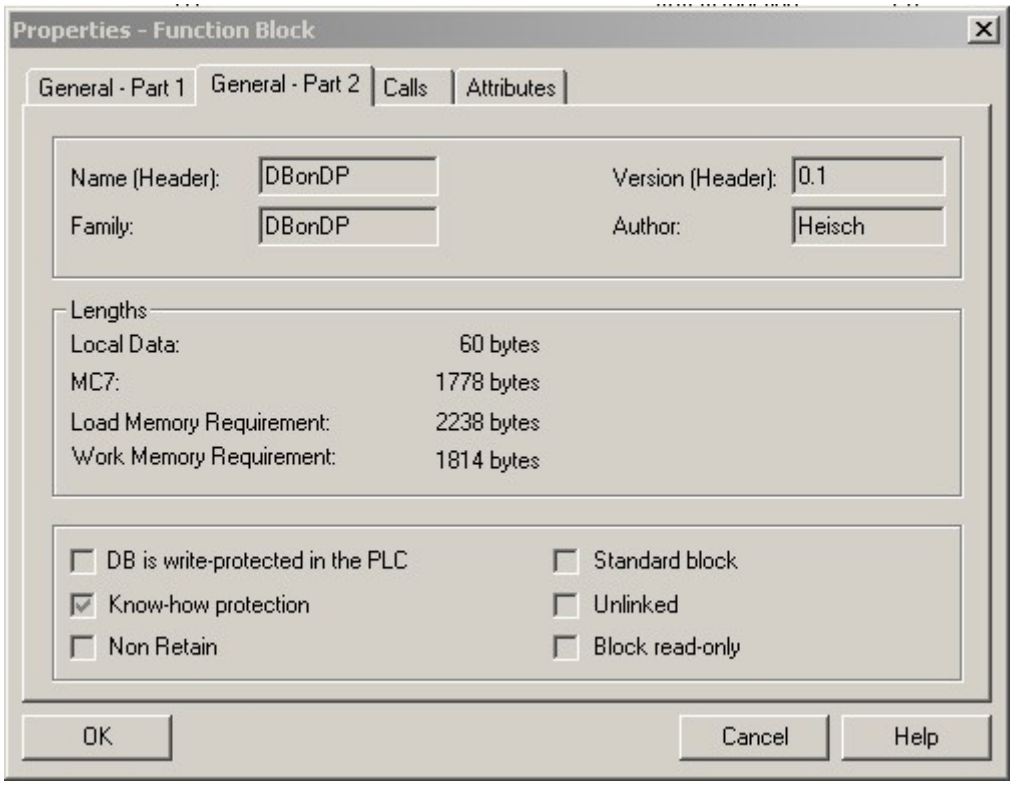
SND_IDLE Sende-Status : keine Bearbeitung, bereit für nächsten Sendeauftrag.

Weitere Informationen:

Fehlerbits im DIW 34 des Instanz-DBs.

34.0	stat	err.b.SFC14_err	BOOL	FALSE	R	SFC14 error detected
34.1	stat	err.b.SFC15_err	BOOL	FALSE	R	SFC15 error detected
34.2	stat	err.b.noHi_err	BOOL	FALSE	R	No Hi-Bit received from remote
34.3	stat	err.b.timeout_err	BOOL	FALSE	R	Timeout error from lifebit toggle
34.4	stat	err.b.block_len_err	BOOL	FALSE	R	defined block is not in [4..112] word limits
34.5	stat	err.b.res_5	BOOL	FALSE	R	
34.6	stat	err.b.res_6	BOOL	FALSE	R	
34.7	stat	err.b.res_7	BOOL	FALSE	R	
35.0	stat	err.b.SFC20_rcv_err	BOOL	FALSE	R	SFC20 receive part error detected
35.1	stat	err.b.SFC20_snd_err	BOOL	FALSE	R	SFC20 send part error detected
35.2	stat	err.b.res_12	BOOL	FALSE	R	
35.3	stat	err.b.res_13	BOOL	FALSE	R	
35.4	stat	err.b.res_14	BOOL	FALSE	R	
35.5	stat	err.b.res_15	BOOL	FALSE	R	
35.6	stat	err.b.res_16	BOOL	FALSE	R	
35.7	stat	err.b.res_17	BOOL	FALSE	R	

Ressourcen:



Transfergeschwindigkeit:
Der FB benötigt zu einem Senden mindestens 3 Zyklen.
Der erste Zyklus überträgt die Zieladresse im Partner-AG, die weiteren Zyklen transportieren die Daten, der letzte Zyklus ist der Bestätigungszyklus: der Sender wartet auf eine positive Quittung vom Empfänger.

Die Anzahl der Datenzyklen hängt von der Datenmenge und der Breite des IO-Koppelbereichs (siehe Eingangsparameter IO_Length_Words) ab.
Das erste Wort im IO-Bereich wird immer für die Kontrollbits benutzt, es bleibt für die Datenübertragung also noch (IO_Length_Words – 1) *2 Bytes.

Beispiel:
Als Blockgröße wurden 32 Worte eingestellt, dies ist der größte in einer CPU315-DP einstellbare konsistente Datenbereich.

Es soll 1kByte übertragen werden.

Die Anzahl der Zyklen errechnet sich:

$$Cdat = \frac{Länge[Byte]}{(IO\_Length\_Word - 1) * 2} = \frac{1024}{(32-1)*2} = \frac{1024}{62} = 16{,}52 \rightarrow 17$$

Anzahl der Gesamtzyklen : 17 +2 = 19

Das dominierende Element der Übertragung ist die CPU mit dem längsten Zyklus.
Abschätzung an bekannten Werten:
eine gängige Zykluszeit ist sicherlich 20ms, eine gängige Buslaufzeit ca. 2-3 ms. Dies wären zusammen ca. 23 ms, wir nehmen mal 25 an.

Laufzeit des Telegramms: 19 Zyklen * 25 ms = 475ms.
Dies entspräche etwa 2,1 kByte/s oder einer Baudrate von ca. 17 kBd.

Da der FB 120 jeweils ein unabhängig arbeitendes ein Sende- und Empfangsteil beinhaltet, kann über eine Verbindung gleichzeitig gesendet und empfangen werden. Dies hat keine Auswirkung auf die Übertragungszeit.

Herstellung von Datenintegrität:

Beim Entwurf des FB120 wurde auf die Herstellung von Datenintegrität verzichtet, denn dies hätte den Einbau von auf maximale Größe ausgelegten Sende- und Empfangs-Buffern bedeutet, was den Einsatz besonders in kleinen CPUs beeinträchtigt hätte.

Datenintegrität kann aber einfach hergestellt werden:
Sende-Seite
Benutzen Sie einen Sende-DB als Sende-Buffer. Schreiben Sie am Anfang dieses Buffers einen Telegrammkopf: die Nummer des Ziel-DBs, des ersten Bytes und Länge der Daten in Byte-Einheiten.
Kopieren Sie dahinter den eigentlichen Telegramminhalt (Rohdaten).
Die Länge des zu sendenden Telegramms kann auf die Länge des Telegrammkopfs + Länge Rohdaten beschränkt werden.
Verschicken Sie dieses Sendebuffer mit der ermittelten Länge an des Partner-seitige Empfangsbuffer.
Empfangs-Seite:
Wenn das Signal „RCV_RDY“ kommt, wurde der Empfang vollständig durchgeführt.
Mit diesem Impuls kann die Weiterverarbeitung getriggert werden.

Werten Sie den Kopf aus und kopieren Sie die Rohdaten mit dem SFC20 an den eigentlichen Ziel-DB. Gegebenenfalls kann sogar eine Koppelmerker aus dem Kopf abgeleitet werden:

```
L       „Empfangsbuffer“.“header.DB-Nr
L       234   // Wenn DB 234
 ==I
U       „RCV_RDY“
=       Koppelmerker  // Dieser Merker steht nur für einen Zyklus an. Da RCV_RDY ein Impuls ist.
```

Für die Codierung des Headers empfiehlt sich, das ANY-Pointer-Format einzusetzen, weil dieser dann mit dem SFC20 problemlos weiterverarbeitet werden kann.

Bemerkung:
Falls Sie Probleme mit dem ANY-Pointer haben, möchten wir auf unser Tutorial zu diesem Thema hinweisen. Darin sich viele Beispiele und einige FCs, die die obige Problemstellung quasi im Alleingang lösen.

4. Installation und Benutzung des Archivs

- Kopieren Sie die beiliegende ZIP-Datei auf Ihr Programmiergerät.
- Entpacken Sie die Bibliothek in Ihr Bibliotheksverzeichnis (z.B: C:\Programme\Siemens\Step7\S7libs\ ).
- Erzeugen Sie ein neues Projekt für Ihre Testmaschine.
- Kopieren Sie den Inhalt der Bibliothek in das neue Projekt.
- Rufen Sie im OB1 den FC 1 auf.
- Übertragen Sie das Projekt in Ihre Testmaschine.

!!! Hinweis:

Der mitgelieferte FC 9 geistert durch alle unsere Projekte.
Er setzt voraus, dass in der Hardwarekonfiguration der CPU als Taktmerkerbyte das MB1 eingetragen wurde.
(HardwareKonfig->CPU->Eigenschaften->Zyklus/Taktmerker dort: Taktmerker eingeschaltet, Merkerbyte 1)

5. Copyrights

5. Beschreibung des Beispielprogramms

In dem Beispielprogramm wird statt der FB120 der Simulationsbaustein FB 121 eingesetzt.
Er kommuniziert nicht mit dezentraler Peripherie sondern liest und schreibt in einen DB.
Dies ermöglicht das Experimentieren mit der Kommunikation auf nur einer Maschine.

OB 1

```
Netzwerk 1: Allgemeine Funktionen und Zeitimpulse
MB 1 muss als Takt-Merkerbyte der CPU definiert sein !!

--------------------
General functions:  Log 0 / log 1 / time pulses

FY 1 has to be defined as takt FY of the CPU !!
```


```
Symbolinformation:
 FC_AllgemeineFunktionen  FC9                  -- Zeit-Impulse, Blinktakte, Log0,Log1,etc
```

```
Netzwerk 2 : Blocktransfer über DP-Kopplung Deutsche DOKU
Kommentar:
```


```
Symbolinformation:
 log1                     M0.1                 -- Logisch 1
 fc_DEMO_DBonDP_deutsch   FC120                -- Demoprogramm DBonDP deutsche Dokumentation
```

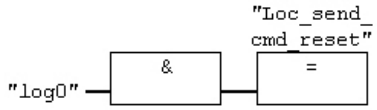
```
Netzwerk 3 : Block transfer over DP link English Dokumentation
Kommentar:
```

Der OB1 beinhaltet den FC 9 für die Log0,Log1 und Taktgenerierung,
Die Taktgenerierung wird für dieTimeout-Erkennung benötigt.

FC 120 Simulation des Lokalen AGs und des Partner-AGs

Um das Beispiel einfach zu halten, wird nur in eine Richtung transferiert. Der FB120 (bzw.FB121) kann natürlich beide Richtungen gleichzeitig.

```
FC120 : DEMO-Baustein deutsch für FB120 ( hier FB121, Simulation)
Kommentar:

Netzwerk 1: LOKAL-Simulation
================================================================================

SIMULATION DES LOKAL AUFGERUFENEN FB 120

================================================================================

In diesem Beispiel ist nur die Sende-Seite beschaltet.


Netzwerk 2: ============== SENDE-SEITE ==============================

Netzwerk 3: Lokal: Sende-Auftrag Reset  / local: reset send command
Normalerweise wird der Eingang Send_Reset nicht beschaltet.
(Vorbesetzung = FALSE)
Falls ein Reset vom Ablauf her notwendig ist, führt der FB den Reset
selbständig aus.

Der Eingang kann aber benutzt werden, um eine Übertragung abzubrechen.
```



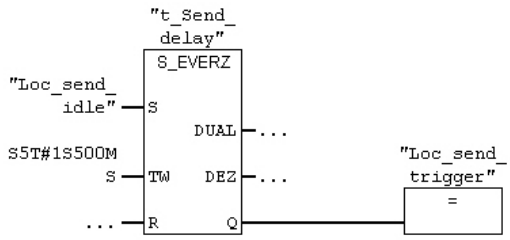

```
Symbolinformation:
 log0                   M0.0              -- Logisch 0
 Loc_send_cmd_reset     M120.1            -- Lokal: Sende-Auftrag Reset  / local: reset send command
```

Netzwerk 4 : Lokal: Sende-Auftrag (Impuls) / local: send command (pulse)

Der Sende-Befehl kann dauerhaft auf "1" stehen, dann wird der parametrierte DB (SND_DB) zyklisch übertragen.

Zum Start reicht aber ein Impuls, am Besten, indem man SEN_IDLE (= bereit für neuen Auftrag) abfragt.

Die Zeit wurde nur zu Demonstrationszwecken eingebaut.

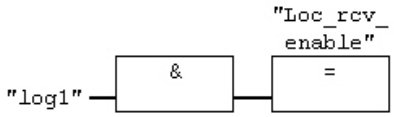


Symbolinformation:

t_Send_delay	T121	-- Verzögerung Senden Neustart (zu Demo-Zwecken)
Loc_send_idle	M121.7	-- Lokal: Senden frei (für neuen Auftrag) / local: indication send idle
Loc_send_trigger	M120.0	-- Lokal: Sende-Auftrag (Impuls) / local: send command (pulse)

Netzwerk 5 : Freigabe des Empfangsteils

Näheres: siehe im Remote-Teil



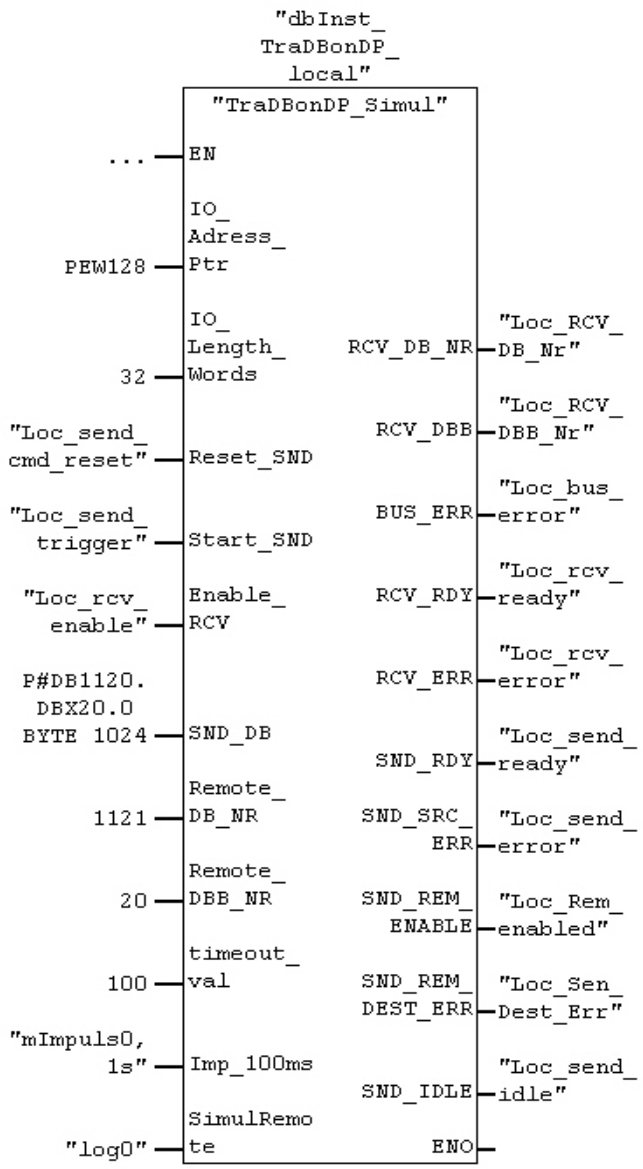
Symbolinformation:

log1	M0.1	-- Logisch 1
Loc_rcv_enable	M120.2	-- Lokal: Freigabe Empfang / local: enable receive

Netzwerk 6 : Lokal: AUFRUF FB 120 (Doku ..)

```
Angabe des IO-Bereichs der Dezentralen Peripherie:
--------------------------------------------------
IO_Address_Ptr   : hier : PEW *128*
IO_Length_Words  : hier : *32*

  also:
   Es müssen im DP-Koppelbereich 2 konsistente Blöcke angelegt werden.
   Beide müssen *32* Worte = 64 Bytes lang sein.
   Eingänge ab PEW *128*, Ausgänge ab PAW *128*

  (weil FB 121, wird hier eigentlich auf den DB *128* verwiesen.)

Senden:
-------
Sende "SND_DB" (hier : P#DB1120.DBX0.0 BYTE 1024 )
zum Partner-AG DB [Remote_DB_NR]. DBB [Remote_DBB] (also DB 1121.DBX0.0)

(Trouble mit ANY-Pointern? für 40 Euro gibts bei uns ein Tutorial incl.
vielen Beispielprogrammen !)
```

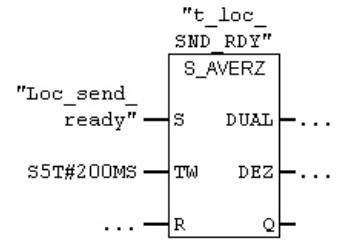
```
Timeout-Erkennung:
------------------
Sollte nach "Timeout_val" * 100ms kein neues Telegramm vom Partner gekommen
sein, wird Timeout -> "BUS_ERR" gemeldet.

"Imp_100ms" muss mit einem 100ms-Impuls (kein Takt!) beschaltet werden.
In diesem Beispiel kommt er aus unserem FC9.

Zusatzbeschaltung für FB121 (gibts in dem "echten" FB120 nicht:)
----------------
"SimulRemote" == 0 zeigt dem FB121 an, dass er hier als FB120-Lokal-Simulation
eingesetzt wird.
```


"dbInst_
TraDBonDP_
local"
"TraDBonDP_Simul"
... EN
IO_
Adress_
PEW128 Ptr
IO_
Length_
32 Words
"Loc_send_
cmd_reset" Reset_SND
"Loc_send_
trigger" Start_SND
"Loc_rcv_
enable" Enable_
RCV
P#DB1120.
DBX20.0
BYTE 1024 SND_DB
Remote_
1121 DB_NR
Remote_
20 DBB_NR
timeout_
100 val
"mImpuls0,
1s" Imp_100ms
SimulRemo
"log0" te
RCV_DB_NR "Loc_RCV_
DB_Nr"
RCV_DBB "Loc_RCV_
DBB_Nr"
BUS_ERR "Loc_bus_
error"
RCV_RDY "Loc_rcv_
ready"
RCV_ERR "Loc_rcv_
error"
SND_RDY "Loc_send_
ready"
SND_SRC_
ERR "Loc_send_
error"
SND_REM_
ENABLE "Loc_Rem_
enabled"
SND_REM_
DEST_ERR "Loc_Sen_
Dest_Err"
SND_IDLE "Loc_send_
idle"
ENO

```
Netzwerk 8: Lokal: AUFRUF FB 120  (Doku Ausgänge )
Parameter OUTPUT
================
RCV_DB_NR          : Nummer des DB, der empfangen wird
RCV_DBB            : Nummer des Offsets des DBB, ab dem empfangen wird.
                     "RCV_DB_NR" und "RCV_DBB" beinhalten den Wert, der
                     Sendeseitig in den Parametern Remote_DB_NR und
                     Remote_DBB_NR eingetragen wurde.
                     Dies zeigt der empfangenden CPU an, welche Daten
                     aktuell übertragen werden.
                     Beide Werte werden zu 0, wenn aktuell kein Empfang.

 Allgemein:
 ----------
BUS_ERR            : Kommunikationsfehler, Details: siehe Bits in DIW 34

 Empfang:
 --------
RCV_RDY            : Empfangs-Status : Fertig
                     Dieses Signal kommt als Impuls, wenn der
                     Empfang regulär beendet wurde.
RCV_ERR            : Empfangs-Status : Fehler
                     Der Ziel-Datenbereich kann unvollständige Daten enthalten.

 Senden:
 -------
SND_RDY            : Sende-Status : Fertig.
                     Dieses Bit wird gesetzt, wenn vom Partner der
                     fehlerfreie Empfang quittiert wurde.
                     ( Siehe Partner:RCV_RDY)
                     Das Bit wird rückgesetzt, wenn lokal ein neuer
                     Sende-Auftrag gestartet wurde. Es wird auch im Fehlerfall
                     zurückgesetzt.
SND_SRC_ERR        : Sendefehler: Der Quell-DB auf der lokalen Maschine
                     konnte nicht gelesen werden. Existiert der DB? Länge ?
SND_REM_ENABLE     : Sende-Status: Der Partner erlaubt den Empfang
SND_REM_DEST_ERR   : Sende-Status: Fehlermeldung vom Partner:
                     Der Ziel-DB fehlt oder ist zu kurz.
                     (Remote-DB-Bereich nicht vorhanden)
SND_IDLE           : Sende-Status : keine Bearbeitung, bereit für nächsten
                     Sendeauftrag.
```

Netzwerk 11 : Hier: Nur zur Anzeige, dass SND_RDY kommt.

Kommentar:



Symbolinformation:

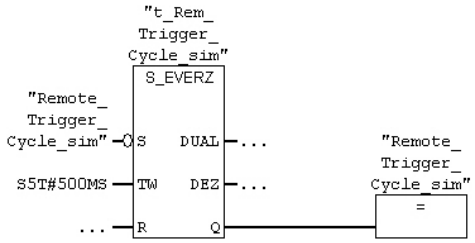
```
t_loc_SND_RDY            T120              -- Lokal: Demo SND_RDY  / local : demo SND_RDY
Loc_send_ready           M121.3            -- Lokal: Senden  fertig / local: indication send ready
```

Netzwerk 12 : REMOTE-Simulation

```
================================================================================

SIMULATION DES IM PARTNER-AG AUFGERUFENEN FB 120

================================================================================

In diesem Beispiel ist nur die Empfangs-Seite beschaltet.
```

Netzwerk 13 : Remote Simulation Trigger des FB

```
Simulation der Zykluszeit des Partner-AGs

Hier: gigantische 500ms !!
```

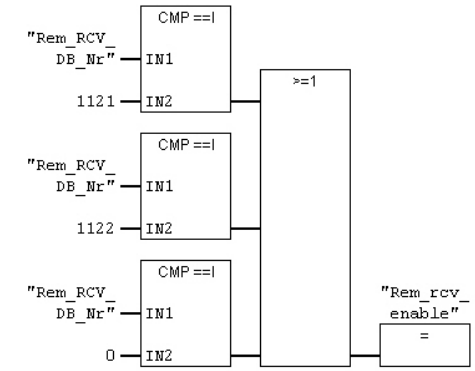


Symbolinformation:

```
t_Rem_Trigger_Cycle_sim  T122              -- Simulation des Zyklus der Gegenstelle / simulation of cycle on remote station
Remote_Trigger_Cycle_sim M137.0            -- Simulation des Zyklus der Gegenstelle / simulation of cycle on remote station
```

Netzwerk 14 : Freigabe Empfang nur für DB 1121 und DB 1122

```
Wenn kein Empfang erlaubt: einfach mit logisch 0 beschalten.
Wenn generell erlaubt: einfach mit Logisch 1 beschalten.

Mit dieser Beschaltung kann der Empfang auf bestimmte Bausteine eingeschränkt
werden.
Hier: DB 1121 und DB1122 sind erlaubt.

Der 3. Zweig ( Vergleich mit 0)ist notwendig, weil es sonst der Partner erst
garnicht probiert.
```

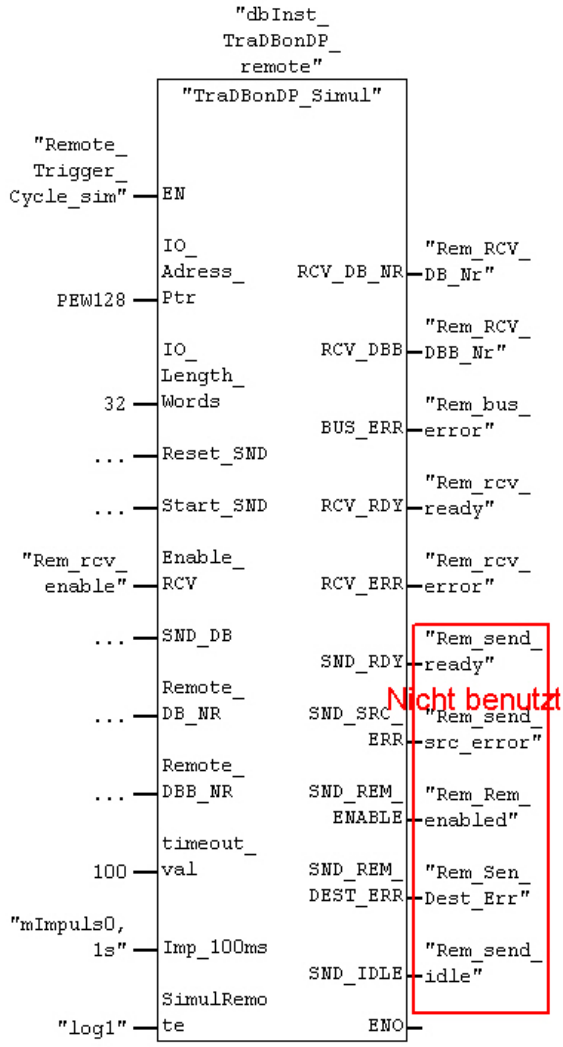


Symbolinformation:

```
Rem_RCV_DB_Nr          MW132       -- Gegenstellen-Simulation: Nr des empfangenen DBs / remote: number of received db
Rem_rcv_enable         M130.2      -- Gegenstellen-Simulation:: Freigabe Empfang / remote: enable receive
```

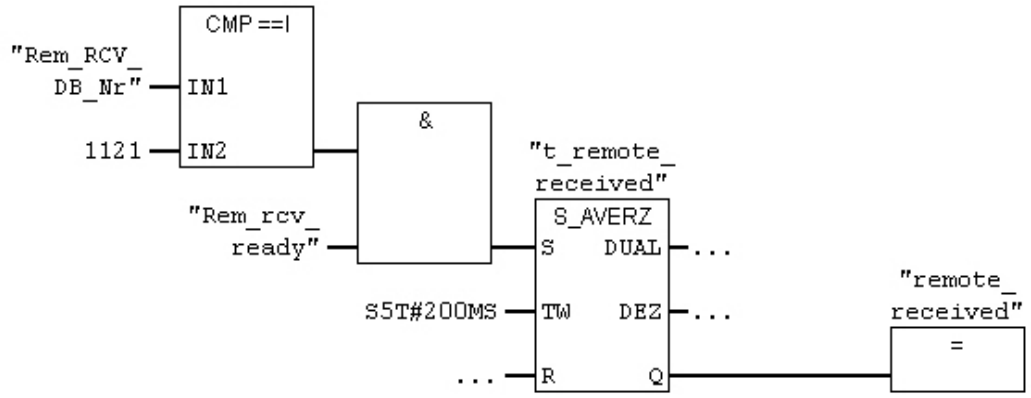
Netzwerk 15 : Remote Simulation Hier: nur Empfang

```
Angabe des IO-Bereichs der Dezentralen Peripherie:
--------------------------------------------------
IO_Address_Ptr    : hier : PEW *128*
IO_Length_Words   : hier : *32*

  also: (weil FB 121, wird hier eigentlich auf den DB *128* verwiesen.
   Es müssen im DP-Koppelbereich 2 konsistente Blöcke angelegt werden.
   Beide müssen *32* Worte = 64 Bytes lang sein.
   Eingänge ab PEW *128*, Ausgänge ab PAW *128*


Senden:  in diesem Beispiel inaktiv
-------

Zusatzbeschaltung für FB121 (Diesen Parameter gibts in dem "echten FB120 nicht:)
-----------------
"SimulRemote" == 1 zeigt dem FB121 an, dass er hier als FB120-Partner-Simulation
eingesetzt wird.
```



Netzwerk 16 : Ausgang Empfang fertig

```
Empfang:
--------
RCV_RDY           : Empfangs-Status : Fertig
                    Dieses Signal kommt als Impuls, wenn der
                    Empfang regulär beendet wurde.
RCV_ERR           : Empfangs-Status : Fehler


Dieses Netzwerk zeigt, wie der Empfang eines bestimmten DBs ermittelt werden
kann.
RCV_RDY wird für einen Zyklus = 1 ( hier: für einen FB121-Aufruf), wenn der
Empfang fehlerfrei beendet wurde.

Die Zeit dient in diesem Beispiel nur dazu, den Impuls länger zu sehen.

Wird gerade nichts übertragen, werden RCV_DB_NR und RCV_DBB gelöscht.
```



Liste der verfügbaren Tutorials und Software-Produkte von Heisch Automation

Der Überblick über aktuelle Angebote für
- Simatic S7 Runtime Software
- Simatic S5 Runtime Software
- S7 Tutorials
- Standard-Tools für die S5- und S7-Programmierung
- Programme für die Linux-Simatic-Connectivity

ist verfügbar über http://sites.inka.de/heisch
Heisch Automation arbeitet mit Emig Software zusammen.

Derzeit sind folgende Produkte verfügbar:

Simatic S7 Runtime Software:

- **Fifo**
  Aufbau von Fifos in der SPS mit beliebiger Anzahl von Einträgen, über beliebig viele DBs,
  Maximale Länge eines Eintrags: 32000 Bytes
  Beschreibungen: deutsch und englisch

- **Sommerzeit**
  kleine Bibliothek zur Sommerzeit-Winterzeit-Umschaltung. Die Echtzeit-Uhr der CPU läuft in Winterzeit oder in UTC.
  Die Echtzeituhr der CPU wird bei der Umstellung nicht verstellt, die errechnete Zeit wird in verschiedenen Formaten bereit gestellt.
  Diese Software läuft auch in alten CPUs.
  Beschreibungen: deutsch und englisch

- **Float5X7**
  Umrechnung des S7-Real-Formats in das S5-KG-Format und Umrechnung des S5-KG-Formats in das S7-Real-Format.
  Diese kleine Bibliothek eignet sich zum direkten Austausch von Gleitpunktwerten zwischen S5- und S7-Maschinen.
  Beschreibungen: deutsch und englisch

Simatic S5 Runtime Software:
direkt verfügbar:
Diverse Treiber zur Aufrüstung von 3964R-Verbindungen auf RK512-Funktionalität.
Beschreibungen unter http://sites.inka.de/heisch

Die S5 ist am Aussterben, wenn auch sehr langsam.
Zwar verfügt unsere Haus-Standard-Software über eine gut ausgestattete Bibliothek an erprobten FBs, speziell für regelungstechnische Belange, allerdings rechtfertigt die Nachfrage den Aufwand nicht, diese gut dokumentiert verfügbar zu machen.

Trotzdem: Wir sind gerne bereit, Ihnen weiterzuhelfen. Vielleicht haben wir das Programm, das Sie

benötigen, noch in der Bibliothek ? (u.a. diverse Regler, Rampen, P-T1-Glied, Min-Max-Auswahl, Polygonzug, Fifo-Verwaltung)

S7 Tutorials:

- **ANY-Tools**
  Kurzlehrgang zum Umgang mit ANY-Pointern. Beschreibung und gut dokumentierte Beispielprogramme.
  Beschreibung: deutsch

Standard-Tools für die S5- und S7-Programmierung

Diese Tools wurden von Emig Software erstellt:

- **ProPro**
  ProPro generiert aus Störmelde-Datenbausteinen Import-Dateien, die direkt in ProTool-Pro importiert werden können.
  Der S7-Programmierer schreibt den einer Störung zugeordneten Text direkt als Kommentar an das Bit im Störmeldebaustein.
  Nach dem Export des DBs als AWL-Quelle erzeugt ProPro aus der exportierten Datei eine Import-Datei, die in ProTool-Pro eingelesen werden kann.
  Vorteil: keine Bit-Rechnerei notwendig, gut dokumentiertes S7 Programm. Keine Tipperei in dem Störmelde-Editor von ProTool-Pro.
  ProPro läuft unter Windows

- **DBQVL**
  Dokumentations-Tool für Simatic S5: Querverweisliste über Datenbausteine
  Das S5 Programmierpaket Step 5 bietet keine Möglichkeit, direkt nach einem Dbx.DWy zu suchen.
  DBQVL erzeugt aus einem S5-Projekt eine Datei, die eine vollständige Querverweisliste über die Variablen in Datenbausteinen erzeugt.
  DBQVL wurde erfolgreich zur Inbetriebnahmeunterstützung und Dokumentation bei bei S5-Großprojekten eingesetzt.
  DBQVL läuft unter DOS und Linux

- **rkslave**
  rkslave ist ein Testprogramm für selbst-entwickelte RK512-Treiber.
  Es simuliert eine SPS mit einem RK512-fähigen Kommunikationsprozessor.
  Es sind viele Testmöglichkeiten einstellbar.
  Das Programm läuft unter DOS und Linux

Programme für die Linux-Simatic-Connectivity

- **rk512_server**
  Kommunikationsserver für Linux zur Kommunikation zu Simatic S5 und S7 über RK512-fähige serielle Verbindungen.

- **rk511_server**
  Kommunikationsserver für Linux zur Kommunikation zu Simatic S5 und S7 über die Programmierschnittstelle.

- **rktcp_server**
  Kommunikationsserver für Linux zur Kommunikation an mehrere Simatic S5 und S7 über TCP/IP.

- **convert_lib**
  Konvertiert S5 und S7-Datentypen zu Linux-Datentypen. Das ANSI-C Programm wird im Quellcode ausgeliefert und ist auch unter Windows einsatzfähig.

Dienstleistungen und Anlagen rund um die Simatic S5 / S7

Sie haben eine Automatisierungsaufgabe oder benötigen einen Simatic-Programmierer ?
Wir sind sicher Ihnen weiterhelfen zu können.

Sie benötigen die komplette elektrische und softwaretechnische Ausrüstung einer Anlage?
Wie sind zwar „nur“ auf Software spezialisiert, sind aber nach über 25 Jahren im Automatisierungsbereich sehr gut vernetzt. Das Spektrum reicht vom spezialisierten Freiberufler bis hin zum mittelständigen Unternehmen mit mehreren hundert Beschäftigten.
Zusammen mit unseren Partnern sind wir in der Lage, bis hin zur schlüsselfertigen Anlage den ganzen Bereich der elektrischen Automatisierung abzudecken.