FIFO

Standard-Software für Simatic S7

Inhalt / Contents

HEISCH AUTOMATISIERUNGSTECHNIK
Ingenieurbüro für Industrieautomatisierung
Ostring 15 Landau/Pfalz Tel. +49 6341 / 890-117 FAX -118
Postadresse: Im Vorderen Großthal 4 76857 Albersweiler / Pfalz
http: //www.heisch-automation.de e-mail: hwauto@heisch.inka.de

A. Beschreibung (deutsch)

A1. Übersicht

Die Funktionsbibliothek FIFO realisiert Fifo-Funktionen, die von bei Step 7 mitgelieferten Bibliotheken nicht mehr abgedeckt werden.

Begrenzungen:

Maximale Länge eines Fifo-Eintrags :	1- 32000 Byte
Maximale Länge eines Fifos:	unbegrenzt
Maximalzahl von Fifo-Einträgen:	unbegrenzt
Anzahl der pro Fifo benutzten DBs :	unbegrenzt

Die effektiven Grenzen werden durch die eingesetzte CPU vorgegeben.

Diese Bibliothek wurde sowohl auf kleinen CPUs (CPU314) als auch auf großen CPUs (CPU414-3, CPU416-2) eingesetzt. Die Bibliothek ist ressourcen-schoned geschrieben.

A2. Funktionsweise

Die Bibliothek besteht aus einem FC (FIFO_INIT) und einem FB (FIFO_ORG) .

Der FC (FIFO_INIT) initialisiert die Fifoverwaltung. Er wird normalerweise nur beim Programmstart durchlaufen, es ist parametrierbar, ob der Fifo dabei gelöscht wird.

Der FB (FIFO_ORG) organisiert das Schreiben und lesen des Fifos.
Funktionen:

- I Schreiben in den Fifo, wenn Fifo voll, nicht schreiben
- U Schreiben in den Fifo, wenn Fifo voll, ältesten Eintrag überschreiben
- O Lesen aus den Fifo (FB liefert positive Rückmeldung, wenn aus dem Fifo gelesen wurde.)
- C Kontrolle Rückgabe des Füllstands in (Anzahl Einträge)
- R Fifo Reset (löschen)

Der FB (FIFO_ORG) muß zum Realisierung eines Fifos mindestens 2 mal aufgerufen werden. Diese "zerrissene" Arbeitsweise wurde gewählt, weil dadurch der Fifo weitergehend konfigurierbar ist, als wäre eine gemeinsame Ein-Ausgangsfunktion geschaffen worden:

- Nur die Funktion benötigt Rechenzeit, die auch verwendet wird.
- Weniger Ein-Ausgabeparameter -> weniger Stack Operationen ->höhere Geschwindigkeit, weniger Programmspeicherbedarf
- Bessere Skalierbarkeit der Eingangsfunktionen: es kann an verschiedenen Programmstellen und aus verschiedenen Eingangsbuffern gelesen werden, ohne indirekte Adressierung benutzen zu müssen.
- Bessere Skalierbarkeit der Ausgabefunktionen (z.B: Rechner leert Fifo nur bei Bedarf, soll dann aber mehrere Einträge gleichzeitig lesen können)

Hinweise:

- FIFO_ORG füllt nicht den gesamten Fifo, ein Eintrag bleibt immer leer.
- Eingabe und Ausgabe-Operationen eines Fifo müssen in der gleichen Zeitbearbeitung

stattfinden.
(z.B : alle Aufrufe innerhalb OB1 Bearbeitung). DER FB FIFO_ORG ist im strengen Sinne nicht reentrant. (gemeinsamer Arbeitsbereich durch Instanz-DB)

Der zu dem FB generierte Instanz-DB ist der Verwaltungs-DB des Fifo (DB_FIFO_ORG). Pro Fifo wird ein Verwaltungs-DB angelegt. Dieser DB dient als Zustandsspeicher der Fifo--Organisation, er speichert keine Fifo-Daten

Die DBs als Speicher des Fifo (Speicher-DBs) müssen vom Anwender zusätzlich generiert werden.

Anforderungen:
- Ein Speicher-DB muß mindestens so groß sein, wie ein Fifo-Eintrag (Record). Records werden nicht Baustein-überlappend abgelegt, es wird deshalb empfohlen, als Größe eines Speicher-DBs das Vielfache einer Record-Länge zu wählen.
- Alle Speicher-DBs eines Fifos müssen gleich groß sein.
- Die Speicher-DBs müssen in einem zusammenhängenden Nummernbereich liegen (z.B: DB 30 bis DB 41)

A3. Komponenten und Ressourcen

FC 20 FIFO_INIT

Der FC initialisiert den Verwaltungs-DB DB_FIFO_ORG

Länge: 192 Byte
Länge Lokaldaten 30 Byte
verschiebbar: ja
benutzte E/A/M: keine

PARAMETER:
==========
EINGANGSPARAMETER

Fifo_org	BLOCK_DB	: DB, Identisch mit Instanz-DB des FB FIFO_ORG
Fifo_DB	BLOCK_DB	: DB, 1. DB des Fifo-Ringpuffers. Es können mehrere DBs angelegt werden. Die DBs müssen identisch aufgebaut sein. Die Aufrufnummern müssen hintereinander liegen.
Cnt_Fifo_dbs	INT	: Anzahl der DBs des FIFO-Ringpuffers (Min. 1)
Bytes_per_db	DINT	: Länge eines DBs des FIFO-Ringpuffers in Bytes
Bytes_per_rec	INT	: Anzahl der Bytes pro Fifo-Eintrag
Reset	BOOL	: 0 = keine Funktion 1 = Rücksetzen des Fifos

ENO ist immer gesetzt.

FB 20 FIFO_ORG

Der FB bearbeitet lesen und schreiben in den Fifo.

Länge:	804 Byte
Länge Instanz	74 Byte
Länge Lokaldaten	38 Byte
verschiebbar:	ja
benutzte E/A/M:	keine

PARAMETER:
==========
EINGANGSPARAMETER:

Start	BOOL	: Freigabe Bearbeitung bei Funktionen I,U,O,R
Mode	CHAR	: Betriebsart, siehe unten
EA_DB_Ptr	POINTER	: Pointer auf 1.Bit in EingabeBuffer / AusgabeBuffer, Abhängig von Funktion

AUSGANGPARAMETER:

State	INT	: Status / Rückmeldung, abhängig von Mode
OK	BOOL	: Binäre Rückmeldung, abhängig von Mode

Mode = 'C' Check Füllstand: State bringt Anzahl Einträge zurück
EA_DB_Ptr muß für diese Funktion nicht besetzt sein.
Rückmeldungen:
State = Anzahl der Einträge im Buffer, falls mehr als 32767 Einträge im Fifo, dann wird 32767 ausgegeben.
OK = 0 : keine Einträge
= 1 : lesen möglich, Daten in Fifo

Mode = 'I' in Fifo schreiben, wenn voll, nicht schreiben
Rückmeldungen:
State = 0 : OK, geschrieben
= 1 : Schreiben nicht möglich, Fifo voll
OK = 0 : Fehler
= 1 : OK, geschrieben

Mode = 'U' in Fifo schreiben, wenn voll, dann ältesten Eintrag überschreiben
sonst wie 'I'

Mode = 'O' aus Fifo lesen
Rückmeldungen:
State = 0 : OK, gelesen
= 1 : lesen nicht möglich, Fifo leer
OK = 0 : nicht gelesen
= 1 : OK, Daten in Zielbuffer geschrieben

Mode = 'R' Fifo Reset (löschen)
Rückmeldungen:
State = 0 : immer 0
OK = 1 : immer 1

ENO ist immer gesetzt.

B. Description (english)

B1. Overview

The library FIFO contains the functionality to build fifos of a kind, which can not been built by the fifo funtions which are included in Step7 standard libraries.

Limits:

Maximum length of a fifo record :	1- 32000 bytes
Maximum length of a fifo:	unlimited
Maximum count of a fifo record :	unlimited
maximum count of DBs used by a fifo :	unlimited

The effective limits are defined by the properties of the used CPU .

This library has been used on small CPUs (CPU314) and also on big CPUs (CPU414-3, CPU416-2) . The library was written to handle very carefull with the use of resources.

B2. Functionality

The library consists of a FC (FIFO_INIT) and a FB (FIFO_ORG) .

FC (FIFO_INIT) initialises the fifo organization. normally it is called while starting the programm (OB100). It is possible to parameter if the fifo has to be reset while running FC FIFO_INIT.

FB (FIFO_ORG) organizes writing to the fifo and reading from the fifo.
Modes:

- I writing into the fifo, if full, do not write
- U writing into the fifo, if full, overwrite the oldest record
- O reading out of the fifo (output flag OK, if reading was successful)
- C Control returns the count of records in the fifo (count of records)
- R Fifo reset

Programming a fifo, the FB (FIFO_ORG) needs to be called 2 times as a minimum, on time for input and one time for output.
This fuzzy method was chosen, because it enables a higher flexibility of the usage and a higher performance.

- Only the used mode need cpu time
- fewer parameters-> fewer stack operations -> higher performance, lower memory usage
- Input functions better scalable (may be spead all over the program and may read at different positions in the program (EA_DB_Ptr can point to different input blocks without the necessity of indirect addressing.
- Output functions better scalable: more than one output buffer possible

Hints:
- FIFO_ORG does not fill the complete fifo , one record always is empty.
- Input and output of one fifo must work in the same time level (i.e. All work called by OB1). FB FIFO_INIT is not reentrant (common workspace is DB_FIFO_ORG).

The DB belonging to FB FIFO_ORG is the organization DB (DB_FIFO_ORG).
one fifo needs one organization DB. It only contains organization data, it does not contain storage data.

The DBs wich are needed for data storage (the fifo buffer itself) need to be generated by the user.

Requirements:
- A storage-DB in minimum needs to be as large tas a record.
  Records can not overlap DBs, it is recommended to chose the length of a storage DB as a multiple of a record length.
- All storage DBs need to have the same length.
- The storage DBs need to be in contigous numbers.
  (z.B: DB 30 bis DB 41)

B3. Components and resources

FC 20 FIFO_INIT

The FC FIFO_INIT initializes the DB DB_FIFO_ORG.

Length:	192 Byte
Length Local data	30 Byte
relocatable:	yes
used I/O/F:	none

PARAMETERS:
==========

INPUT PARAMETERS		
Fifo_Org	BLOCK_DB	: DB DB_FIFO_ORG
Fifo_DB	BLOCK_DB	: storage DB, first DB of the Fifo ring buffer. More than one DB can be used. The DBs have to be identical and have to to be on sequential DB numbers.
Cnt_Fifo_dbs	INT	: Count of the DBs of the FIFO ring buffer (Min. 1)
Bytes_per_db	DINT	: Length of an single DB of the FIFO ring buffers (calculated in bytes)
Bytes_per_rec	INT	: count of bytes of one record
Reset	BOOL	: 0 = idle 1 = reset the fifo

ENO allways is set.

FB 20FIFO_ORG

This FB executes reading and writing operations to the fifo.

Length:	804 bytes
Length Local data	74 bytes
Length static data	38 bytes
relocatable:	yes
used I/O/F:	none
uses	SFC20

PARAMETER:
==========

INPUT PARAMETERS

Start	BOOL	: Enable for the modes I,U,O,R
Mode	CHAR	: mode, see below
EA_DB_Ptr	POINTER	: Pointer to the first bit of input buffer / output buffer, depends from Mode, if input buffer or output buffer

OUTPUT PARAMETERS

State	INT	: State / feed back, depends from Mode
OK	BOOL	: Binary feed back, depends from Mode

Mode = 'C' Check contents: State returns the count of records in the fifo
EA_DB_Ptr does not need to be set.
return codes:
State = count of record in fifo, if more than 32767 returns 32767.
OK = 0 : no records
= 1 : reading possible, data in fifo

Mode = 'I' write to fifo, if full, do not write
return codes:
State = 0 : OK, written
= 1 : not written, fifo full
else return code of SFC20
OK = 0 : error
= 1 : OK, input written

Mode = 'U' write to fifo, iif full overwrite oldest record
rest the same as 'I'

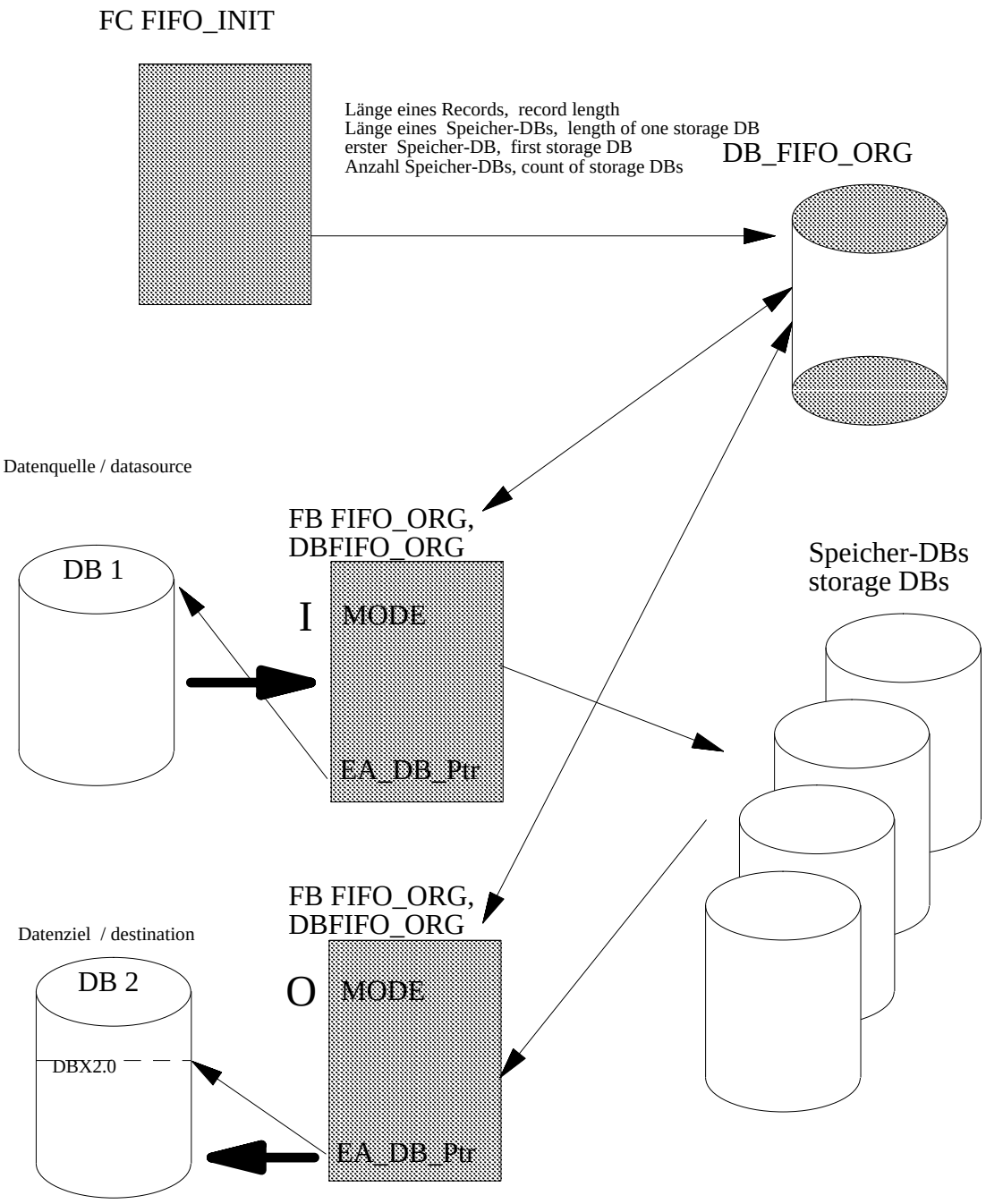
Mode = 'O' read from fifo
return codes:
State = 0 : OK, read
= 1 : impossible to read, fifo empty
else return code of SFC20

OK = 0 : not read
= 1 : OK, data read to destination buffer

Mode = 'R' Fifo Reset (clear fifo)
return codes:
State = 0 : always 0
OK = 1 : always 1

ENO allways is set.

D. Diagramm


FC FIFO_INIT
Länge eines Records, record length
Länge eines Speicher-DBs, length of one storage DB
erster Speicher-DB, first storage DB
Anzahl Speicher-DBs, count of storage DBs
DB_FIFO_ORG
Datenquelle / datasource
FB FIFO_ORG,
DBFIFO_ORG
DB 1
I
MODE
EA_DB_Ptr
Speicher-DBs
storage DBs
FB FIFO_ORG,
DBFIFO_ORG
Datenziel / destination
DB 2
O
MODE
DBX2.0
EA_DB_Ptr

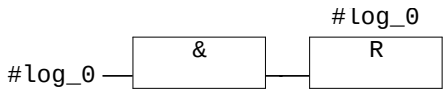
OB100 - <offline>

"OB_Anlauf_Beispiel" Anlauf OB Beispiel für Benutzung von FIFO
Name: **Familie:**
Autor: **Version:** 0.1
Bausteinversion: 2
Zeitstempel Code: 23.08.02 02:07:58
Interface: 14.08.02 03:55:02
Längen (Baustein / Code / Daten): 00200 00086 00032

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	temp	OB100_EV_CLASS	BYTE		16#13, Event class 1, Entering event state, Event logged in diagnostic buffer
1.0	temp	OB100_STRTUP	BYTE		16#81/82/83/84 Method of startup
2.0	temp	OB100_PRIORITY	BYTE		Priority of OB Execution
3.0	temp	OB100_OB_NUMBR	BYTE		100 (Organization block 100, OB100)
4.0	temp	OB100_RESERVED_1	BYTE		Reserved for system
5.0	temp	OB100_RESERVED_2	BYTE		Reserved for system
6.0	temp	OB100_STOP	WORD		Event that caused CPU to stop (16#4xxx)
8.0	temp	OB100_STRT_INFO	DWORD		Information on how system started
12.0	temp	OB100_DATE_TIME	DATE_AND_TIME		Date and time OB100 started
20.0	temp	log_0	BOOL		Logisch 0

Baustein: OB100 "Complete Restart"

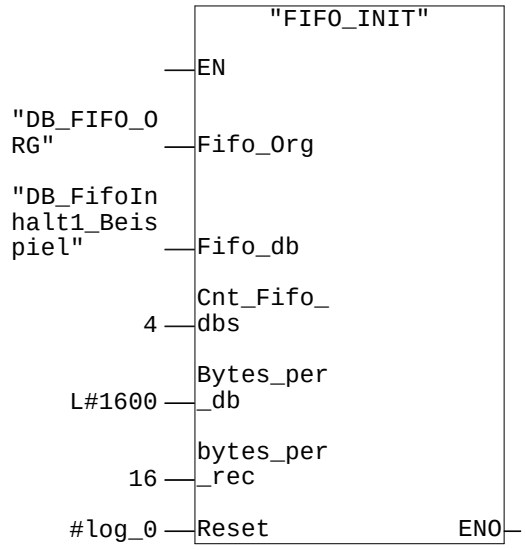
Netzwerk: 1 Logisch 0 bilden / logical zero



Netzwerk: 2 Initialisierung des Fifos beim Anlauf

```
Der FC FIFO_INIT initialisiert den DB FIFO_VERW.


PARAMETER:
==========
Fifo_org        : DB, Identisch mit Instanz-DB des FB FIFO_ORG
Fifo_DB         : DB, 1. DB des Fifo-Ringpuffers.
                  Es können mehrere DBs angelegt werden.
                  Die DBs müssen identisch aufgebaut sein.
                  Die Aufrufnummern müssen hintereinander liegen.
Cnt_Fifo_dbs    : Anzahl der DBs des FIFO-Ringpuffers ( Min. 1 )
Bytes_per_db    : Länge eines DBs des FIFO-Ringpuffers in Bytes
Bytes_per_rec   : Anzahl der Bytes pro Fifo-Eintrag
Reset           : 0 = keine Funktion
                  1 = Rücksetzen des Fifos

Im aktuellen Fall besteht der Fifo-Buffer aus den DBs 30 - 33,
Also: Fifo_DB        = DB 30
      Anz_Fifo_DBs   = 4
      Bytes_pro_DB   = (Länge des DB 30 in Byte ) = 1600

Ein Fifo-Eintrag entspricht UDT 1, also 16 byte
Also: Bytes_pro_eintr = 16

Soll nach einem SPS-Neustart der Fifo gelöscht werden, dann
Reset = Log_1  sonst Reset = Log_0

ENO ist immer gesetzt.

-------------------
```

```
The FC FIFO_INIT initializes the DB FIFO_VERW.


PARAMETER:
==========
Fifo_Org        : DB, identical with the Instanz-DB of FB FIFO_ORG
Fifo_DB         : DB, first DB of the Fifo ring buffer.
                  More than one DB max be used.
                  THE DBs have to be identical and have to to be on
                  sequential DB numers.
Cnt_Fifo_dbs    : Count of the DBs of the FIFO ring buffer ( Min. 1 )
Bytes_per_db    : Length of an single DB of the FIFO   ring buffers
                  (calculated in bytes)
Bytes_per_rec   : count ofd bytes of one record
Reset           : 0 = idle
                  1 = reset the fifo

In the actual example, the fifo consists of 4 DBs,
these are the DBs 30 to 33
->    Fifo_DB        = DB 30
      cnt_Fifo_dbs   = 4
      Bytes_pre_DB   = (Length of DB 30 in Byte ) = 1600

one Fifo record consists of UDT 1, these are 16 byte
->  Bytes_per_rec = 16

If the fifo shall be reset on plc restart,
Reset = Log_1  else Reset = Log_0

ENO allways is set.
```



Symbolinformation

FC20	FIFO_INIT	FIFO Initialisierung
DB20	DB_FIFO_ORG	VerwaltungsDB eines Fifos
DB30	DB_FifoInhalt1_Beispiel	DB Fifo-Ringpuffer 1. DB Beispiel

OB1 - <offline>

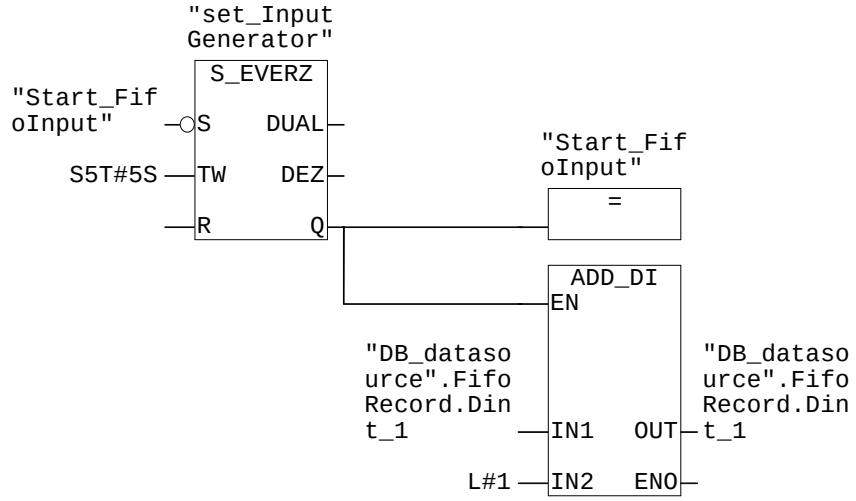
"CYCL_EXC" Cycle Execution
Name: **Familie:**
Autor: **Version:** 0.1
Bausteinversion: 2
Zeitstempel Code: 23.08.02 02:42:44
Interface: 15.02.96 16:51:12
Längen (Baustein / Code / Daten): 00256 00136 00022

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	temp	OB1_EV_CLASS	BYTE		Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
1.0	temp	OB1_SCAN_1	BYTE		1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
2.0	temp	OB1_PRIORITY	BYTE		Priority of OB Execution
3.0	temp	OB1_OB_NUMBR	BYTE		1 (Organization block 1, OB1)
4.0	temp	OB1_RESERVED_1	BYTE		Reserved for system
5.0	temp	OB1_RESERVED_2	BYTE		Reserved for system
6.0	temp	OB1_PREV_CYCLE	INT		Cycle time of previous OB1 scan (milliseconds)
8.0	temp	OB1_MIN_CYCLE	INT		Minimum cycle time of OB1 (milliseconds)
10.0	temp	OB1_MAX_CYCLE	INT		Maximum cycle time of OB1 (milliseconds)
12.0	temp	OB1_DATE_TIME	DATE_AND_TIME		Date and time OB1 started

Baustein: OB1 "Main Program Sweep (Cycle)"

Netzwerk: 1 Beispiel: Generieren der Fifo-Einträge (Nur als Demo)

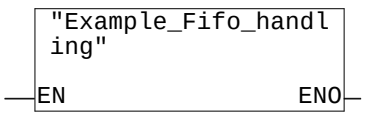
Example: Generation of Fifo-Input-Data



Symbolinformation

T1	set_InputGenerator	Zeit zur Erzeugen neuer Fifo Einträge
M11.0	Start_FifoInput	Start Eintrag in Fifo / Start Input to Fifo
DB1.DBD4	"DB_datasource".FifoRecord.Dint_1	

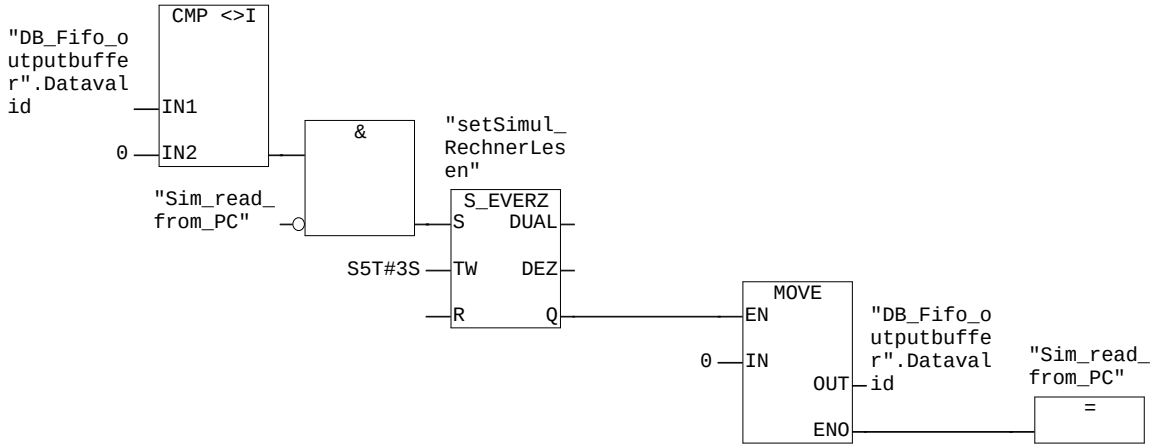
Netzwerk: 2 Fifo-Bearbeitung // Fifo handling



Symbolinformation
FC1 Example_Fifo_handling Beispiel fuer Fifo-Anwendung

Netzwerk: 3 Beispiel: Simulation Rechner zum Auslesen des Fifos

```
EXAMPLE: this part simulates the computer which reads a fifo

--------

Beispiel hier: altes Rechnerhandshake
"DB_Fifo_outputbuffer".Datavalid wird von SPS auf 1 gesetzt, wenn
Ausgangspuffer mit Daten befüllt.

Rechner prüft "DB_Fifo_outputbuffer".Datavalid, falls <> 0
   liest Daten und setzt anschliessend "DB_Fifo_outputbuffer".Datavalid
   wieder auf 0

Wichtig ist hier aber lediglich der Anstoß "Fifo_lesen"

---------

Example here: classical computer handshake
"DB_Fifo_outputbuffer".Datavalid is filled by 1 after filling the
output buffer.
The computer checks sequentially if "DB_Fifo_outputbuffer".Datavalid
is not 0 : This signals thar the buffer has been filled.
Coputer reads the buffer and clears  "DB_Fifo_outputbuffer".Datavalid

Note: Only the start of "Fifo_lesen" ( read fifo ) is important.
```



Symbolinformation

DB2.DBW0	"DB_Fifo_outputbuffer".Datavalid	Kennung an Rechner, Daten vorhanden
M5.0	Sim_read_from_PC	Simulation lesen vom PC
T2	setSimul_RechnerLesen	Simulation lesen vom Rechner

FC1 - <offline>

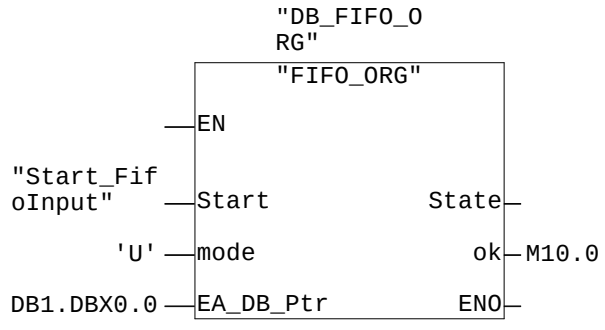
"Example_Fifo_handling" Beispiel fuer Fifo-Anwendung
Name: **Familie:**
Autor: **Version:** 0.1
Bausteinversion: 2
Zeitstempel Code: 23.08.02 02:35:36
Interface: 14.08.02 01:50:16
Längen (Baustein / Code / Daten): 00414 00304 00006

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
	in				
	out				
	in_out				
	temp				

Baustein: FC1 Beispiel für Arbeitweise mit Fifo // Example Fifo handling

Netzwerk: 1 Daten werden an Fifo übergeben // Write Data to Fifo

```
Der Instanz-DB verwaltet den Fifo. Siehe auch Aufruf FC FIFO_INIT in OB 100

Daten werden übergeben, wenn Fifo voll, werden die ältesten Daten überschrieben

Modus = 'I' in Fifo schreiben, wenn voll, nicht schreiben
   Rueckmeldungen:
        State  = 0 : OK, geschrieben
               = 1 : Schreiben nicht moeglich, Fifo voll
          OK   = 0 : Fehler / keine Bearbeitung
               = 1 : OK, geschrieben

EA_DB_Ptr DB1.DBX0.0 ist ein Pointer auf den Bereich, aus dem die Daten geholt
werden.

Modus = 'U' in Fifo schreiben, wenn voll, dann ältesten Eintrag ueberschreiben
  sonst wie 'I'


--------------------

The DB "DB_FIFO_VERW" is the organisation DB of the Fifo. See also FC FIFO_INIT
in OB 100.

Mode  = 'I' : Write into the Fifo, if fifo is full, do not overwrite
    Return codes
       State   = 0 : OK, has written
               = 1 : Fifo is full, impossible to write
          OK   = 0 : error / no action
               = 1 : OK, has written

EA_DB_Ptr DB1.DBX0.0 is a pointer to the first bit of the (actual) input buffer

Mode  = 'U' : Write to the Fifo, if it is full, overwrite oldest.
```

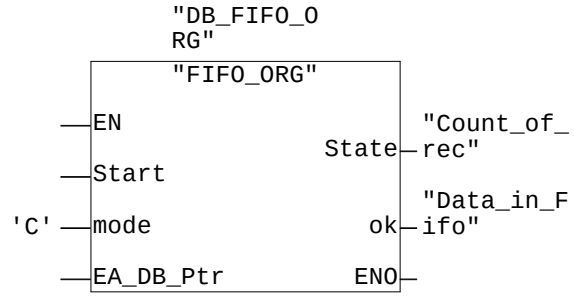


Symbolinformation

FB20	FIFO_ORG	FIFO Verwaltung
DB20	DB_FIFO_ORG	VerwaltungsDB eines Fifos
M11.0	Start_FifoInput	Start Eintrag in Fifo / Start Input to Fifo

Netzwerk: 2 Fifo-Füllstand prüfen / Check count of contents

```
Mode = 'C' Check Fuellstand: State bringt Anzahl Eintraege zurueck
   Rueckmeldungen:
        State  = Anzahl der Einträge im Buffer
        OK      = 0 : keine Einträge
                = 1 : lesen moeglich, Daten in Fifo

EA_DB_Ptr muß für diesen Modus nicht besetzt sein

-------------------

Mode = 'C' Check the count of records in the fifo
    returncodes
       state = count of records in fifo
       OK    =  0 : no record in fifo
             =  1 : reading enabled, >= 1 records in the fifo

EA_DB_Ptr needs not to be set for this mode
```

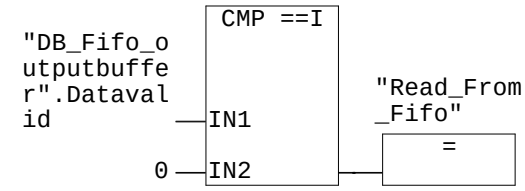


Symbolinformation

FB20	FIFO_ORG	FIFO Verwaltung
DB20	DB_FIFO_ORG	VerwaltungsDB eines Fifos
MW12	Count_of_rec	Anzahl der Einträge im Fifo
M100.1	Data_in_Fifo	Daten sind im Fifo

Netzwerk: 3 Beispiel: Prüfen, ob Daten aus dem Fifo gelesen werden sollen

Example: Check if data should be read



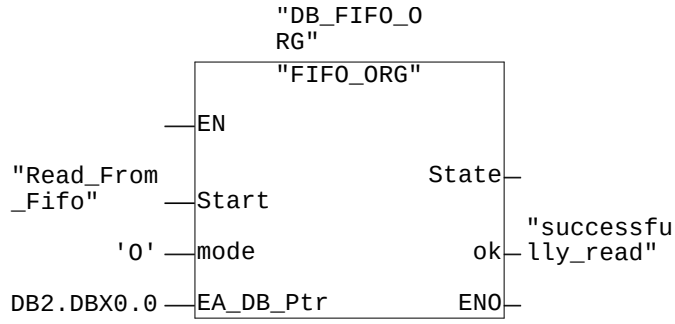
Symbolinformation

DB2.DBW0	"DB_Fifo_outputbuffer".Datavalid	Kennung an Rechner, Daten vorhanden
M11.1	Read_From_Fifo	Lese aus Fifo

Netzwerk: 4 Daten aus Fifo lesen / read data from fifo

```
Mode = 'O' aus Fifo lesen
   Rueckmeldungen:
        State = 0 : OK, gelesen
              = 1 : lesen nicht moeglich, Fifo leer
          OK  = 0 : nicht gelesen
              = 1 : OK, Daten in Ausgangsbuffer geschrieben
```

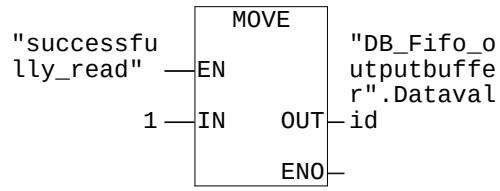
```
EA_DB_Ptr DB1.DBX0.0 ist ein Pointer auf den Bereich, auf den die Daten
geschrieben werden.

--------------------

Mode = 'O' read from fifo
  return codes:

        State = 0 : OK, data read
              = 1 : impossible to read, fifo empty
          OK  = 0 : not read / idle
              = 1 : OK, data written to output buffer

EA_DB_Ptr DB1.DBX0.0 is a pointer to the destination buffer, to where the
data were written.
```



Symbolinformation

FB20	FIFO_ORG	FIFO Verwaltung
DB20	DB_FIFO_ORG	VerwaltungsDB eines Fifos
M11.1	Read_From_Fifo	Lese aus Fifo
M10.1	successfully_read	erfolgreich gelesen

Netzwerk: 5 Beispiel: Kennung Daten vorhanden / Flag data valid

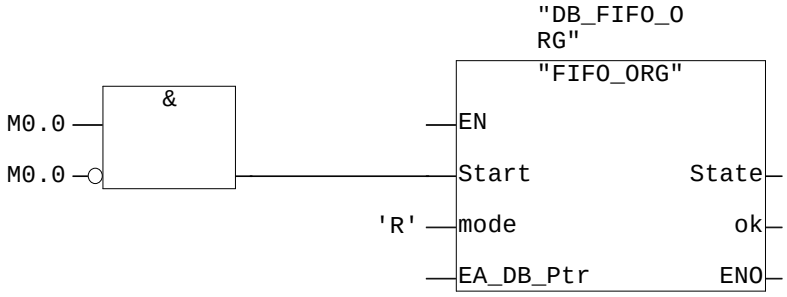


Symbolinformation

M10.1	successfully_read	erfolgreich gelesen
DB2.DBW0	"DB_Fifo_outputbuffer".Datavalid	Kennung an Rechner, Daten vorhanden

Netzwerk: 6 Im Fehlerfall: Fifo rücksetzen / Case of error: reset fifo

```
Mode = 'R' Fifo ruecksetzen ( löschen)

EA_DB_Ptr muß für diese Funktion nicht besetzt sein.

   return codes = 0;

--------------
Mode = 'R' reset fifo (delete)

EA_DB_Ptr need not to be set.

   return codes = 0;
```



Symbolinformation

FB20	FIFO_ORG	FIFO Verwaltung
DB20	DB_FIFO_ORG	VerwaltungsDB eines Fifos

DB1 - <offline>

"DB_datasource" DB Beispiel für Datenquelle

Name: **Familie:**
Autor: **Version:** 0.1
Bausteinversion: 2
Zeitstempel Code: 23.08.02 01:46:14
Interface: 14.08.02 01:41:39
Längen (Baustein / Code / Daten): 00130 00022 00000

Baustein: DB1

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	FifoRecord	"UDT_FifoEintrag_Beispiel"		Beispiel fuer Fifo-Eintrag
+16.0	anything_else	DINT	L#0	Unwichtig .... / anything else
=20.0		END_STRUCT		

DB2 - <offline>

"DB_Fifo_outputbuffer" DB Beispiel für Fifo-Ausgabe

Name: **Familie:**
Autor: **Version:** 0.1
Bausteinversion: 2
Zeitstempel Code: 23.08.02 02:09:15
Interface: 14.08.02 01:51:31
Längen (Baustein / Code / Daten): 00134 00024 00000

Baustein: DB2

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	Datavalid	INT	0	Kennung an Rechner, Daten vorhanden
+2.0	FifoRecord	"UDT_FifoEintrag_Beispiel"		Beispiel fuer Fifo-Eintrag (Quelle für Rechner ?)
+18.0	irgendwas	DINT	L#0	Unwichtig ....
=22.0		END_STRUCT		

DB20 - <offline>

"DB_FIFO_ORG" VerwaltungsDB eines Fifos

Name: **Familie:** DB_TRA
Autor: Heisch **Version:** 0.0
Bausteinversion: 2
Zeitstempel Code: 23.08.02 02:35:29
Interface: 23.08.02 00:15:31
Längen (Baustein / Code / Daten): 00160 00030 00000

Baustein: DB20

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	in	Start	BOOL	FALSE	Funktion ausführen (Modus I,U,O,R)
1.0	in	mode	CHAR	' '	Modus:'I'=inFifoSchreiben,'U'=Überschreiben,'O'=ausFifolesen 'R'=FifoReset
2.0	in	EA_DB_Ptr	POINTER		DBx.DBXy.z Nummer des DB.DBX .. zum Lesen oder schreiben
8.0	out	State	INT	0	Statusausgabe, Betriebsartabhaengig
10.0	out	ok	BOOL	FALSE	Funktion ausgeführt (kein Fehler)
12.0	stat	Fifo_db_1_nr	INT	0	*)Nummer des 1. Fifo-DB
14.0	stat	Fifo_db_n_nr	INT	0	*)Nummer des letzten Fifo_db
16.0	stat	eintrag_laenge	INT	0	*)Anzahl der Bytes pro Eintrag
18.0	stat	Eintraege_gesamt	INT	0	*)Gesamtanzahl der Eintraege
20.0	stat	eintraege_db	INT	0	*)Gesamtanzahl der Einträge pro DB
22.0	stat	Schreib_zeiger	INT	0	Nr des Fifoeintrags zum Schreiben
24.0	stat	Lese_Zeiger	INT	0	Nr des Fifoeintrags zum lesen
26.0	stat	Anz_eintraege_ist	INT	0	aktuelle Anzahl der Eintraege

DB30 - <offline>

"DB_FifoInhalt1_Beispiel" DB Fifo-Ringpuffer 1. DB Beispiel

Name: **Familie:**
Autor: **Version:** 0.1
Bausteinversion: 2
Zeitstempel Code: 14.08.02 01:47:38
Interface: 14.08.02 01:47:38
Längen (Baustein / Code / Daten): 01714 01602 00000

Baustein: DB30

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	x	ARRAY[1..100]		Ist ein Array of UDT x, könnte aber auch z.B. A rray of byte sein
*16.0		"UDT_FifoEintrag_Beispiel"		
=1600.0		END_STRUCT		

UDT1 - <offline>

"UDT_FifoEintrag_Beispiel" UDT Beispiel für einen Fifo-Eintrag

Name: **Familie:**
Autor: **Version:** 0.1
Bausteinversion: 2
Zeitstempel Code: 13.08.02 23:47:51
Interface: 13.08.02 23:47:51
Längen (Baustein / Code / Daten): 00002 00002 00000

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	real_1	REAL	0.000000e+000	
+4.0	Dint_1	DINT	L#0	
+8.0	int_1	INT	0	
+10.0	word_1	WORD	W#16#0	
+12.0	byte_1	BYTE	B#16#0	
+13.0	bool_1	BOOL	FALSE	
+14.0	Res_1	WORD	W#16#0	Reserve
=16.0		END_STRUCT		