User Manual

Intersystem transmission of digital signals with timestamp

between

Siemens S7-PLC Systems

and

S7A Driver / OPC-Server

Date: 31.03.2011

Author: J. Stähler, InCoSol-Industrial Communications Solutions

Table of contents:

Table of figures:

1. Introduction

The S7A Version 7.40 with build no 105 offers a new function which allows you to receive a number of up to 4096 digital signals with timestamp from the PLC and to forward these signals to the clients (iFIX and/or OPC).
By the "regular" polling of the PLC data area the timestamp will be generated by the SCADA System (within the S7A driver). Due to communication delays between the PLC and the driver such a timestamp do **not** represent the real time (the time the signal actually occured) of the signal.

With the new S7A option "Digital Signals" it is now possible to generate the timestamp in the PLC at the time as the signal state change occurs and to transfer this time stamp together with the related signal state to the SCADA system.

Furthermore the signals and their time stamp will be buffered within a FIFO buffer of the PLC and asynchronously transferred to the S7A driver. This buffering ensures that no signal changes will be lost (assumed that the size of the FIFO buffer is sufficient) in case of short signal changes and/or low communication speed between PLC and S7A driver.
The signal transfer to the S7A driver occurs by a dynamically length transfer data block. The S7A driver periodically polls a single handshake byte of this data block. The handshake byte will be set by the PLC when new signal changes occurred and buffered in the FIFO. Afterwards the S7A driver just reads the amount of data bytes which are needed to transfer these signal changes including their timestamps.

Due to this transfer method the communication traffic between PLC and S7A driver is reduced to a minimum and even a large number of signals can be transferred effectively by a quiet short transfer data block.

The following picture shows an overview about this new functionality of the driver.

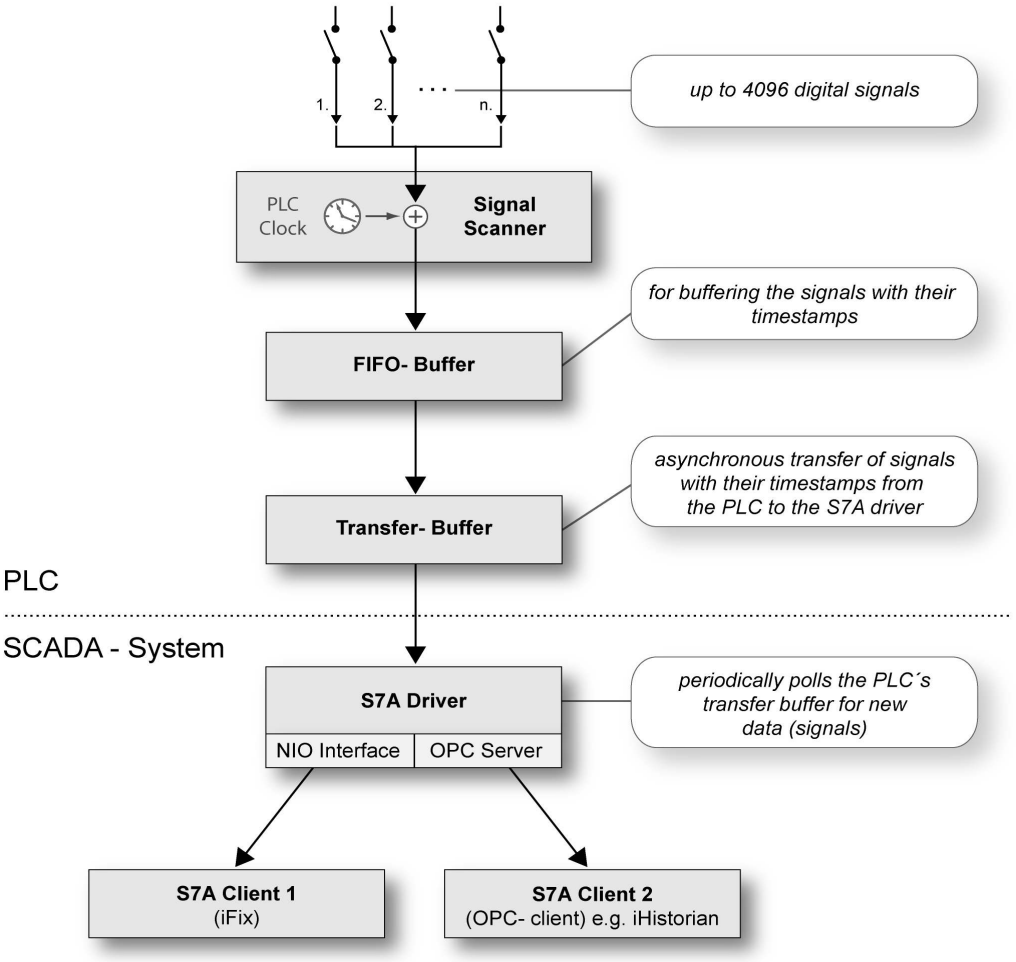


figure 1. overview components

1.1 Calling and datablock structure

The diagram below shows the calling structure of functions (FC) and the function blocks (FB) and their associated data blocks (global and instance DBs).

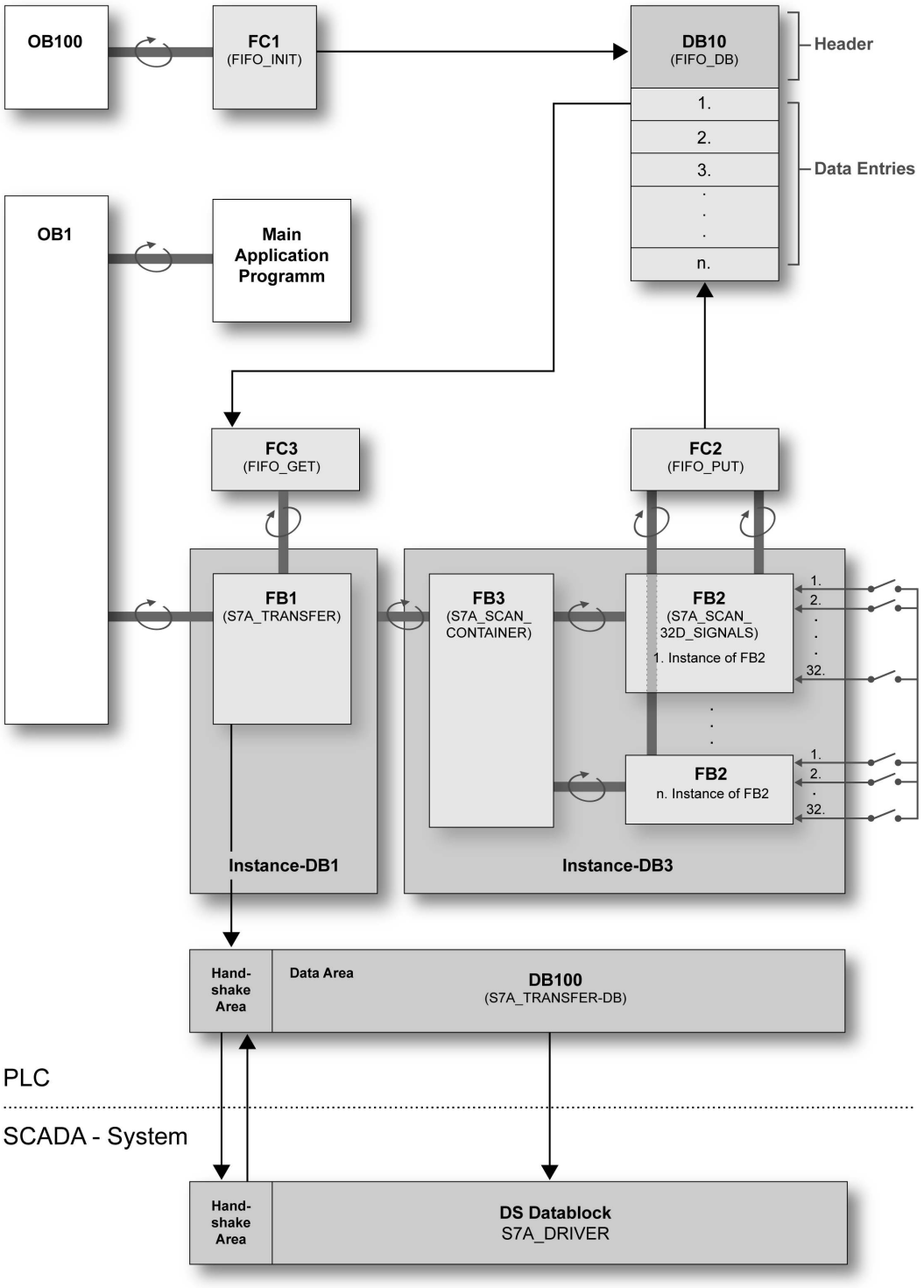


figure 2. calling and data block structure

2. Module structure

The necessary PLC program modules for capturing, buffering and the transmission of the digital signals are divided into two groups:

1. Four function (FCs) used for the management of the FIFO buffer
2. Three function blocks (FBs) are required for the scanning of digital signals and for the transfer of signal change messages to the S7A driver.

The functions (FCs) and function blocks (FBs) may not be changed by the user, except to the function block named **S7A_SCAN_CONTAINER (FB3)** !

2.1 FIFO Management

For the FIFO management within the PLC the four function FC1 (FIFO_INIT), FC2 (FIFO_PUT), FC3 (FIFO_GET) and FC4 (FIFO_COUNT) are implemented. These functions can handle any size and structure of data. The contents of the data element is transparently for the FIFO-functions. Certainly only data elements of the same length can be managed within the FIFO. For this particular application a FIFO data element (entry) has a size of 10 byte.

2.1.1 FC1 (FIFO_INIT)

Inputs:

Symbolic name	Type	Specification
FIFO_DB_NO	Int	Number of the DB which serve as FIFO buffer
FIFO_ENTRY_SIZE	Int	Length of one FIFO data element in bytes

Outputs:

NO_OF_ELEMENTS	Word	Number of data elements which can be stored within the FIFO DB
RET_VAL	Int	Function return value 0: function returned without error < > 0: error occurs within the function

First of all the function checks if the FIFO data block which is passed to the function by parameter *FIFO_DB_NO* already exists. In case the DB is not existing the function will be canceled with an appropriate error code at output *RET_VAL*.
If the data block exists the function calculates - based on the size of the data block and the size of the FIFO data element passed to the function by parameter *FIFO_ENTRY_SIZE*- the maximum number of data elements which can be stored inside the FIFO.The calculated number of data elements will be returned at the output *NO_OF_ELEMENTS*.

Note: The data bytes of the FIFO DB will not be initialised (set to 0). Only the header data (byte 0...9 of the FIFO DB) will be initialized with the initial values for first , last and number of elements in FIFO. In case that no errors were identified, the output *RET_VAL* returns the value 0. Otherwise it returns a general error code. For general error codes see description of SFC24 "TEST_DB".

References: This FC will be called from FB1 (S7A_TRANSFER)

2.1.2 FC2 (FIFO_PUT)

Writes a data element into the FIFO DB.

Inputs:

Symbolic name	Type	Specification
FIFO_DB_NO	Int	Number of the DB which serve as FIFO buffer
FIFO_DATA	Any	ANY- Pointer to the FIFO data element which has to be inserted into the FIFO-DB

Outputs:

RET_VAL	Int	Function return value 0: exit the function without error < > 0: error occurs within the function

Note: Before this function will be called it is mandatory to initialise the FIFO DB by calling the function FIFO_INIT (FC1).

First of all the function checks if the data block which is passed to the function by the parameter *FIFO_DB_NO* already exists. In case the DB is not existing the function will be canceled with an appropriate error code at output *RET_VAL*.

In case the DB already exists the function will check if there is sufficient space for the new data element inside the FIFO DB . If so, the data element will be copied into the FIFO. There will be no implied check whether the length of the FIFO data element (as defined in FIFO_INIT) and the specified length of the source data element (denoted inside the ANY- Pointer of parameter *FIFO_DATA*) are equal. In case of inconsistency in length, the internal call to the BLKMOV function will fail and the function will be aborted with an appropriate error code at output *RET_VAL*.. In case that the FIFO- buffer is full, the output *RET_VAL* will return the specific error code W#16#8081.

For general error codes see decsription of SFC24 "TEST_DB".

References: This FC will be called from FB2 (S7A_SCAN_32D_SIGNALS)

2.1.3 FC3 (FIFO_GET)

Reads the first data element from the FIFO DB provided that the FIFO is not empty.

Inputs:

Symbolic name	Type	Specification
FIFO_DB_NO	Int	Number of the DB which serves as FIFO buffer
FIFO_DATA	Any	ANY Pointer to the buffer for a data element which is removed from the FIFO

Outputs:

RET_VAL	Int	Function return value 0: exit the function without error < > 0: error occurs within the function

Note: Before this function will be called it is mandatory to initialise the FIFO DB by calling the function FIFO_INIT (FC1).

First of all the function checks if the data block which is passed to the function by the parameter

FIFO_DB_NO already exists. In case the DB is not existing the function will be canceled with an appropriate error code at output *RET_VAL*.
In case the DB exists the function will check if the FIFO contains at least one data element. If so the element which first came in will be copied to the target data element which is addressed by the ANY Pointer of the input parameter *FIFO_DATA*. There will be no implied check whether the length of the FIFO data element (as defined in the FIFO_INIT) and the target data element (as defined in the ANY Pointer) are equal. In case of inconsistency in length, the internal call of the BLKMOV function will fail and the function will be aborted with an appropriate error code at output *RET_VAL.*

In case that the FIFO is empty the output *RET_VAL* will return the specific error message W#16#8082. In case of errors due to an invalid source data element (invalid ANY- Pointer) it will return a generally error message.

For general error codes see decsription of SFC24 "TEST_DB".

References: This FC will be called from FB1 (S7A_TRANSFER)

2.1.4 FC4 (FIFO_COUNT)

Returns the current number of data elements within the FIFO which is specified by the parameter *FIFO_DB_NO*.

Inputs:

Symbolic name	Type	Specification
FIFO_DB_NO	Int	Number of the DB which serves as FIFO buffer

Outputs:

COUNT	Int	Number of data elements currently in the FIFO
RET_VAL	Int	Return value 0: exit the function without error < > 0: error occurs within the function

Note: Before this function will be called it is mandatory to initialise the FIFO DB by calling the function FIFO_INIT (FC1).

First of all the function checks if the data block which is passed to the function by the parameter *FIFO_DB_NO* already exists. In case the DB is not existing the function will be canceled with an appropriate error code at output *RET_VAL*.
In case it exists the actual amount of data elements will be read from the FIFO header and returned on the output parameter *COUNT*. In case of an invalid FIFO DB number the output *RET_VAL* will return a generally error message.

For general error codes see decsription of SFC24 "TEST_DB"

References: This FC will be called from FB1 (S7A_TRANSFER)

3. Signal Scanner and transfer to the S7A driver

The function blocks FB1 (S7A_TRANSFER) , FB2 (S7A_SCAN_32D_SIGNALS) and FB3 (S7A_SCAN_CONTAINER) are serving for the scanning of digital signals and for the transfer of these signals to the S7A driver.

Main FB is the FB1 (S7A_TRANSFER), which in turn calls the FB3 (S7A_SCAN_CONTAINER). The FB3 contains one or more instances of FB2 (S7A_SCAN_32D_SIGNALS).
Each instance of FB2 can handle up to 32 digital signals.
Example: To monitor 100 digital signals it is necessary to declare 4 (4 x 32 = 128) instances of FB2 within FB3. Both FBs (FB2 and FB3) have declared static data so they require an instance DB. Since instances of FB2 are declared as static data within FB3 just one instance DB for FB3 is required. This instance DB contains the static data of the single FB3 instance and furthermore the static data of all FB2 instances which are declared within the FB3. Therefore this DB is a so-called “multi instance DB”. This multi instance DB was established to reduce the amount of necessary instance DBs to a minimum.
Together with the instance DB for the FB1 (S7A_TRANSFER) only two instance DBs are required for this particular application.
Furthermore a global DB for the FIFO buffer and another global DB as transfer buffer to the S7A driver are required.

3.1 Function blocks

The following section describes all function blocks which are the main part of the entire PLC software.

3.1.1 FB1 (S7A_TRANSFER):

It is the main function block which must be called periodically (within OB1).

Note: This FB may not be changed by the users!

Inputs:

Symbolic name	Type	Specification
INIT	bool	TRUE, to initialise the function
FIFO_DB	BlockDB	DB which serves as FIFO buffer
FIFO_ELEM_SIZE	Int	length of the FIFO data element in byte
TRANSFER_DB	BlockDB	DB which serves as transfer buffer to the S7A driver

Outputs:

RTC	Int	Number of data elements inside the FIFO
RET_VAL	Int	Function return value 0: exit the function without error < > 0: error occurs within the function

The FB1 controls the transfer to the S7A driver via the transfer DB and furthermore it calls the FB3 (S7A_SCAN_CONTAINER) which in turn calls all the instances of FB2 (S7A_SCAN_32D_SIGNALS).

For the initialisation of the whole application the input *INIT* has to be set to TRUE for at least one PLC program cycle. During the initialisation a parameter check will be done first. It checks if the FIFO DB (passed to the function block by parameter *FIFO_DB*) exists. In case that the DB exists the function FC1 (*FIFO_INIT*) will be called to initialize the FIFO. The next step is to check if the

S7A Transfer DB (passed to the function block by parameter *TRANSFER_DB*) exists. In case it exists it will get filled with null values. Thereby possible pending handshake signals from or to the S7A driver would be restored to initial values. Afterwards the handshake signals are in a defined initial state.

Cyclic function:
In case that the input *INIT* is set to FALSE the main functions of the FB1 will be executed. These are:

1. Check if an interrogation command from the S7A driver was sent. If so, the interrogation command will be forwarded to the FB3 (S7A_SCAN_CONTAINER).

2. FB3 (S7A_SCAN_CONTAINER) will be called. Within the FB3 all instances of FB2 (S7A_SCAN_32D_SIGNALS) will be invoked.

3. Checks if the FIFO DB contains data elements (digital signals). If so, these signals will be removed from the FIFO and written to the Transfer DB .
   In case that the Transfer DB is not available (locked by the S7A driver) or the capacity of the Transfer DB is reached, the data elements (signals) remain in the FIFO and the function block ends.

3.1.2 FB2 (S7A_SCAN_32D_SIGNALS)

This FB monitores changes of up to 32 digital signals. If the FB has recognised a signal state change it creates a data element and writes this data element into the FIFO buffer.

Inputs:

Symbolic name	Type	Specification
In0	bool	1. digital signal input
In1	bool	2. digital signal input
...		
In31	bool	32. digital signal input
FIFO_DB_NO	Int	Number of the DB which serves as FIFO-buffer
BASIS_ADDR	Int	Signal number of the first digital signal
GENERAL_POLL	bool	TRUE if the interrogation command was requested

Output:

RTC	Int	Function return value 0: exit the function without error < > 0: error occurs within the function

Inputs *In0...In32*

The digital signal inputs *In0* to *In31* can be connected with any digital operand (E, A, M or DBX). Every time the signal changes (rising or trailing edge) the FB generates a data element (signal state and timestamp) and put it into the FIFO buffer. Furthermore the last signal state and the corresponding timestamp will be saved into an internal array (part of the instance data area within the instance DB).
The information of the last signal state and its timestamp will be needed when an interrogation command is requested.
In case that an interrogation command was request before the FB the first time has detected a signal change (e.g. after a restart of the PLC program and before the signal state has changed from FALSE to TRUE, so the signal state is still FALSE) a message with the initial signal state (FALSE) and the current timestamp will be generated and transferred into the FIFO buffer. In the

strict sense that's not the real timestamp of the signal but the timestamp represents the time as the interogation command was issued.

Input *BASIS_ADDR*

The numeric value which is passed to the function by the input *BASIS_ADDR* determines the (logically) signal number of the first digital signal *(In0)*. All further signals will obtain the accordingly consecutive numbers. For example if *BASIS_ADDR* has the value 100 then the first digital signal at input input *In0* will get the logical signal number 100 and the last digital signal at input *In31* will get the logical signal number 131. This signal number will be transferred to the S7A driver and will be used by the clients of the S7A driver as a reference to the digital signal. The supported value range of this parameter is 1 to 4065. A value outside of this range would cause an abort of the function block with an error code W#16#A225 (sector error within the 34 parameter) at the *RTC* output parameter of the FB

Note: If multiple instances of the FB2 are declared within the FB3 (S7A_SCAN_CONTAINER) each instance of FB2 has to have its individual and unique *BASIS_ADDR* value.

Input *GENERAL_POLL*

By the *GENERAL_POLL* input the function block will get signalised that an interrogation command was requested. The FB writes the current signal state and timestamp of all 32 signals into the FIFO buffer.

References: This FB will be called from FB3 (S7A_SCAN_CONTAINER)

3.1.3 FB3 (S7A_SCAN_CONTAINER)

The FB serves as a collector (container) for all instances of the real Scan FBs FB2 (S7A_SCAN_32D_SIGNALS). This FB contains one or more instances of FB2, depending on how much digital signals has to be monitored. The user has to declare this instances of FB2 as STAT variables in the interface of FB3.
See the following screen shot for an example on how and where to declare the FB2 instances within the interface of FB3:

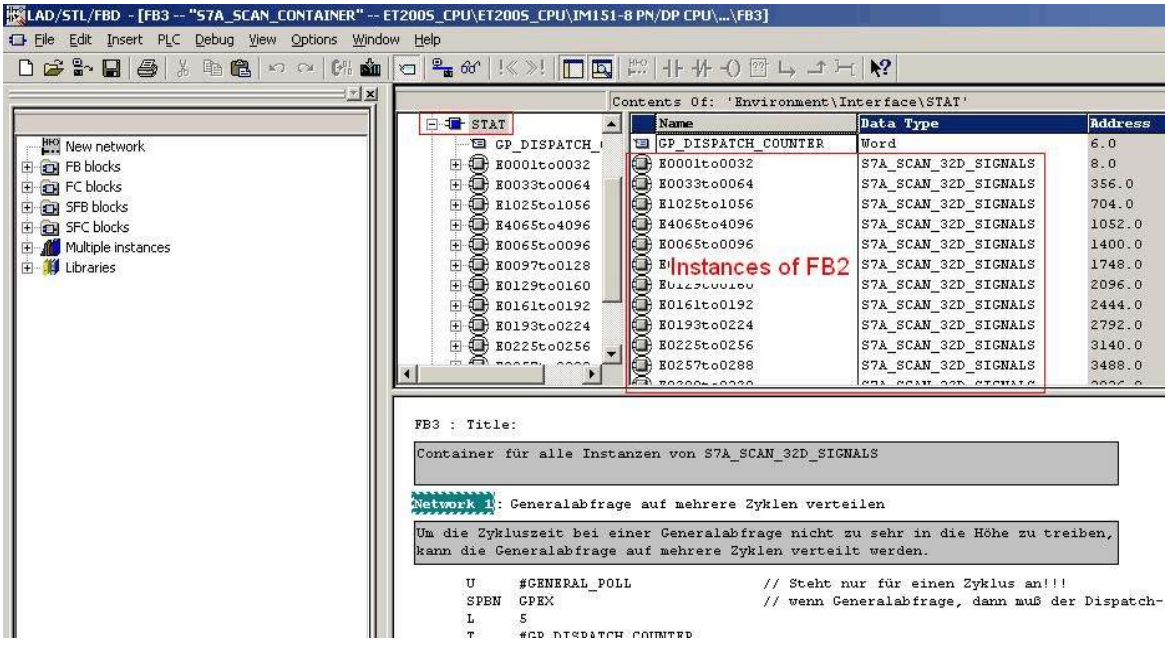


figure 3: static data declaration in FB3

Later on when the Instance DB of the FB3 is created, this Instanz DB contains -in addititon to its own instance data- the instance data of all the FB2 instances which are declared in the FB2's static interface. This multi instance DB saves storage capacity, reduces the number of required DBs and thus minimize the use of the PLC resources.

Input:

Symbolic name	**Type**	**Specification**
FIFIO_DB_NO	bool	Number of the DB which serves as FIFO buffer

Output:

RTC	Int	Function return value 0: exit the function without error < > 0: error occurs within the function

3.1.3.1 General Poll (Interrogation request) Dispatcher:

A special function of the Scan Container FB is the General Poll Dispatcher. This function serves as a program load splitter of an interrogation request for all signals because it splits this command to multiple PLC cycles.
The splitting is necessary when a large number of signal scanFBs (FB2) are declared in FB3. Otherwise an interrogation request could cause a scan cycles time-out (current scan cycle time exceeds the scan cycle monitoring time) which in turn sets the PLC CPU in STOP mode.

The reason for the increase of the scan cycle time is that on an interrogation request all signals will be scanned and thus signal change messages for all signals have to be transferred to the FIFO and furthermore transferred to the S7A transfer buffer.
On hundreds or thousand of signals this operation will generate an significant CPU load and thus a long scan cycle time. In comparison to the regular program flow this additional scan cycle time can be many times over the regular program scan cycle time.
With the Dispatcher the scan cycle load for an interrogation request would be spreaded over multiple program cycles.

Functionality of the Dispatcher:

If the input *GERNERAL_POLL* receives an interrogation request the first step (network 1) for the FB3 is to load the variable DISPATCH_COUNTER by a preset value.

Example: Assumed that FB3 contains 4 instances of the Scan FB2. By setting the DISPATCH_COUNTER to the value 4, it is possible to splitt the interrogation request to 4 scan cycles. In a single scan cycle just one instance of FB2 will be called with the interrogation request .

The trick is that the *GENERAL_POLL* input of a specific Scan FB2 instance will be set to TRUE on a specific Dispatch Counter value.
Example: The *GENERAL_POLL* input of the first instance (signal 1 to 31) of FB2 will set to TRUE if the Dispatch Counter has the value 4.
The *GENERAL_POLL* input of the second instance (signal 32 to 64) of FB2 will set to TRUE if the Dispatch Counter has the value 3.
.
.
The *GENERAL_POLL* input of the fourth instance (signal 97 to 128) of FB2 will set to TRUE if the Dispatch Counter has the value 1.

The Dispatch Counter will be decremented at the end of each scan cycle.

It is not mandatory that each Scan FB2 instance has its own Dispatch Counter value. It is also allowed that multiple Scan FB2 instances shares the same value of the Dispatch Counter.

The decision whether the interrogation command should be divided on multiple PLC scan cycles depends on the amount of Scan FB2 instances and on the performance of the PLC CPU and should decided empirically.

References: This FB will be called from FB1 (S7A_TRANSFER)

4. Data components

The specified PLC program componentes will require 4 data components. Each two of global data blocks and instance data blocks

4.1 Global data blocks:

4.1.1 FIFO buffer for the signal signal messages

The minimum size of the FIFO buffer DB is determined by the number of all digital signals which are defined in the particular application. A a general rule, the FIFO buffer DB should be large enough to host one data element per digital signal.
A data element of a signal signal has a length of 10 bytes (Reference: UDT2- definition). Furthermore the FIFO header requires 10 bytes.

Example on how to calculate the minimum size:

> 5 instances of the Scan FB2 are defined within the Scan Container FB3. The FIFO buffer DB should have a minimum size of **10** + (**5** x **32** x **10**) = 1610 byte.
> **10** bytes FIFO header + (**5 instances** of FB2, each has **32 signals**, each signal requires **10 bytes**)

This size ensures that on an interrogation command the data elements of all signals can be hosted by the FIFO buffer. Even though not all signals will be scanned in a single scan cycle (since the General Poll Dispatcher splitts it to multiple scan cycles) the FIFO buffer should have this minimum size. It can't be assumed that the signal messages will be transfered to the S7A driver immediately after the signal state change was detected. If the S7A driver is offline or in case that a large number of signal state changes occures at the same time (interrogation request) the FIFO buffer DB has to (temporarily) store the data element for all these signal messages. If the PLC provides enough memory, we suggest to configure the FIFO buffer DB twice the minimum size.

4.1.2 Transfer buffer for the S7A communication

The signal messages in the FIFO will be forwarded to the S7A driver by the so-called Transfer DB. The Transfer buffer do not have a fixed structure but it will be filled dynamically by the PLC program (FB1). Dependig on the number of signal messages in the FIFO buffer DB, the Transfer DB will be filled with these messages. The S7A driver do not poll the whole size of the Transfer DB but only the number of bytes which are currenly be used (filled) by the PLC program.

The reserved (configured) size for the Transfer DB depends on the expected amount of simultanous signal state changes. As a basic rule you can estimate 25% of the size of the FIFO Buffer DB. The maximum size of the Transfer DB is 10.000 bytes.

4.2 Instance data blocks:

The function blocks FB1 (S7A_TRANSFER) and FB3 (S7A_SCAN_CONTAINER) are requiring Instance DBs. The instance DB of FB1 (S7A_TRANSFER) has a pre defined (fixed) size. The size of the instance DB for FB3 (S7A_SCAN_CONTAINER) is dynamically and depends on the amount of instances of FB2 inside FB3. Each FB2 instance allocates 346 bytes within the instance DB of FB3.

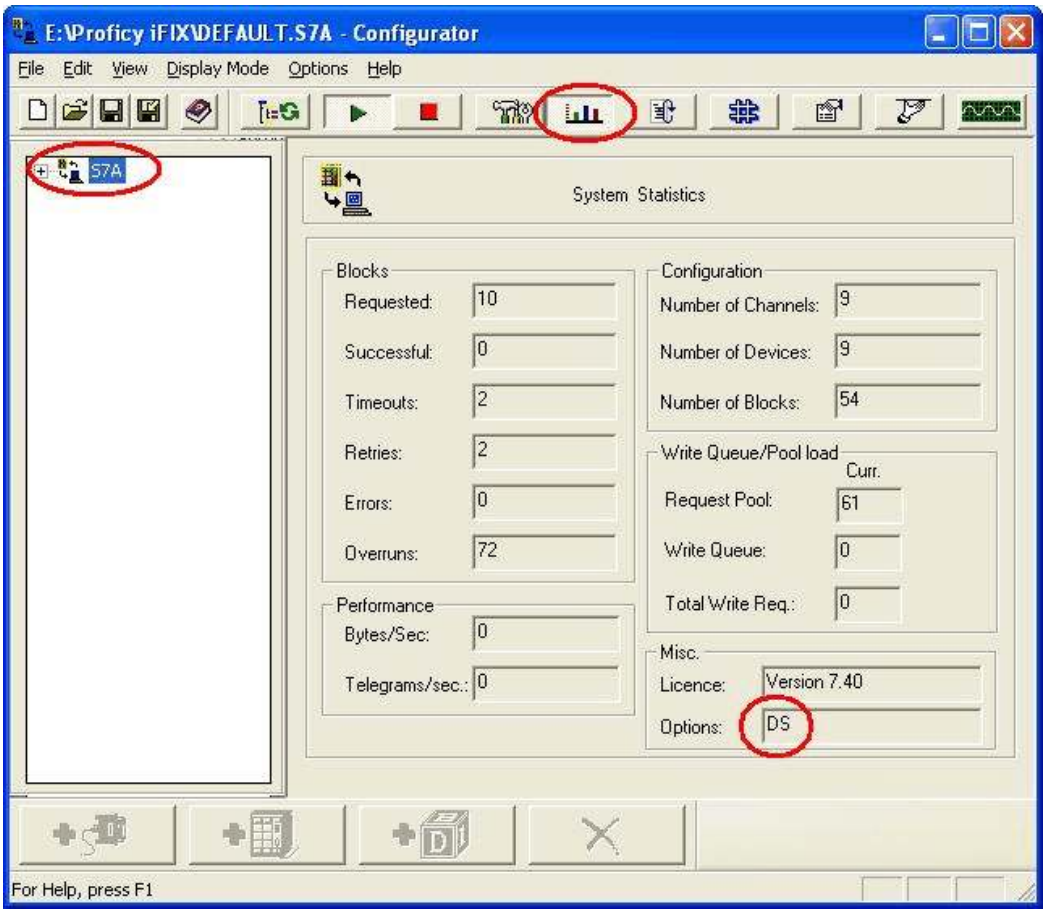
5. S7A Configuration

Note: From the license point of view, this new function is an option, means it is not available with a standard license! A standard license has to be extened (possible by a remote dongle update) to make the DS option available.

The actual license version and available options of a particular dongle can be checked on the System Statistics page in the S7A power tool.

To reach this System Statistics page, open the S7A Power tool, select the S7A root entry in the driver configuration tree view and click the “Statistics” button in the Run-time toolbar.

The following figure shows the System Statistics page of a S7A dongle for version 7.40 with the Digital Signal (DS) option available.



5.1 S7A driver configuration

The S7A driver was extended by the Memory Area “Digital Signals (DS)”. At least one DS master block has to be defined in the S7A power tool. In addition to the DS master block one or multiple so-called DS slave blocks can be added. A DS Masterblock always has a *Primary Rate* equal or greater than zero wheras a DS slave block has to have the *Primary Rate* **DISABLED**.
For both block types (master and slave) the checkbox *Enable* must be checked.

The fragmentation in master and slave blocks is necessary because a single DS block (master or slave) just can contain a maximum of 1024 digital signals. To receive the full range of 4096 digital signals a minimum of four DS blocks (one master block and three slave blocks) is necessary. The actual number and the size of the blocks are arbitrary and can be determined according to the application requirements.

Only the Master block communicates to the PLC. The Master block reads all digital signal messages from the PLC’s data block (which is defined in the field *Transfer DB)* and passes these messages to its own block or to the corresponding slave blocks. If a digital signal message will be received which can’t be found in a Master or a Slave block the message will be discarded.

The field *Starting Address* defines the logical number of the first digital signal within the block. The field *No. of Elements* defines the number of digital signals the block contains. The field *Ending Address* shows the logical number of the last digital signal within the block.

The signal ranges of the Master and Slave blocks may not overlap!

The field *Transfer DB* defines the PLC’s data block number on which the driver receives the digital signal messages. The *Transfer DB* number must be correspond to the number of the so-called S7A Transfer DB on the PLC site (see also 4.1.2). The Transfer DB number has to be defined for master block as well as for all slave blocks.

The following figures show the data block configuration of a DS master and slave block.

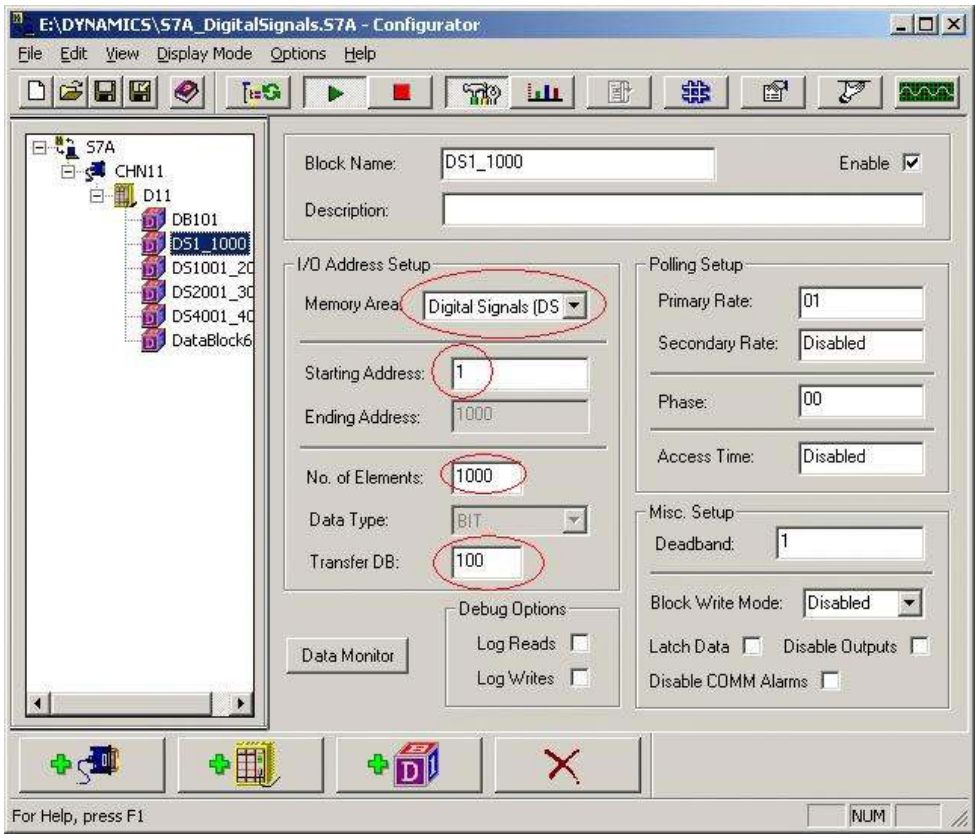


figure 4. DS master block

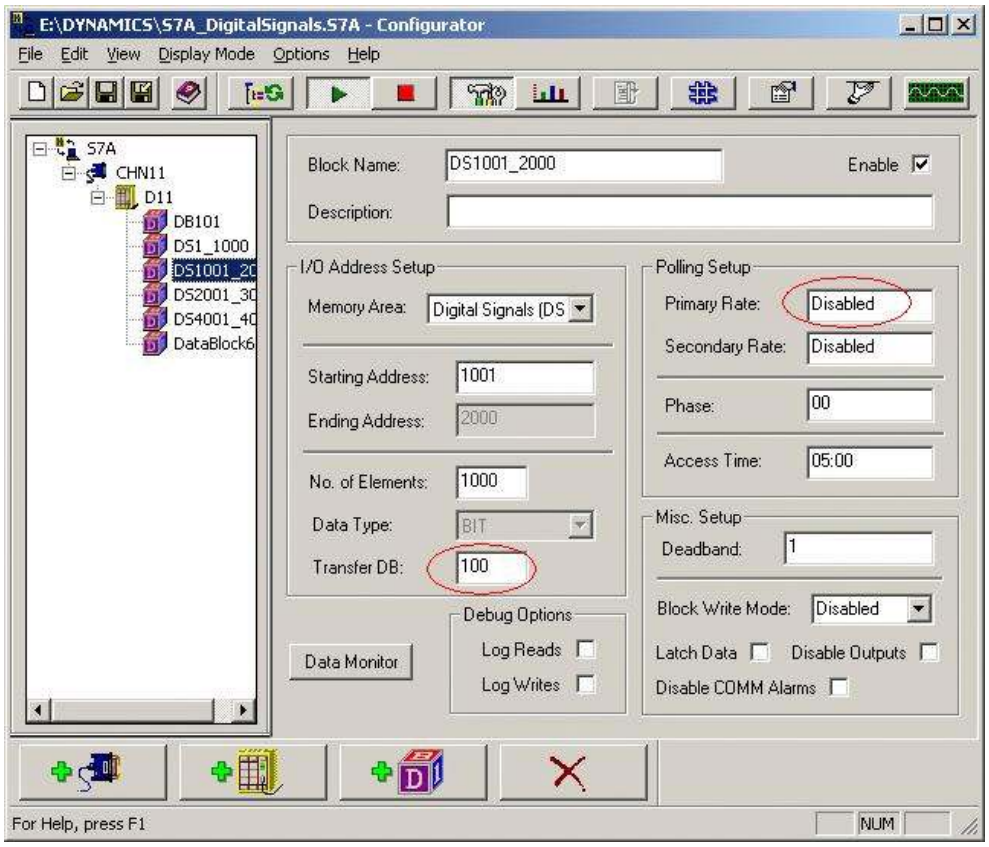


figure 5. DS Slave block

5.2 Client configuration

Digital signals can be processed eighter by the iFIX (e.g. by using the DA- Block) or by an OPC Client.
In iFIX Digital Alarm Tags can be used for DS (digital signals). The Alarm Summery windows shows the date and time of the PLC timestamp.

5.2.1 Address syntax

The address syntax of iFIX Tag I/O Addresses and OPC Item IDs for digital signals is the same:

<Device Name>:DS <Signal Number>

Example: D11:DS 1000 (for the digital signal 1000)

5.2.2 Interrogation request

The driver initiates automatically an interrogation request on each system start. Means, after each start of the S7A driver (or iFIX) all digital signals will be requested and the PLC sends the current state and timestamp of all signals.

By using a special control tag it is possible to initiate an interrogation request directed from the client application. The iFIX Tag I/O Address or rather the OPC Item-ID of the control tag has the following syntax:

!POLL:<*Block Name*>

whereas *Block Name* is the name of the DS Masterblock. Referring to the figure 3 above (image DS Master block) the control tag has the following format: "!POLL:DS1_1000"
The interrogation command will be send when the control tag's current value will be set to any value.