SPS Kommunikation Modultec

Interface Spezifikation

**Übersetzt von M. Aper (Magna Hannover)**

**Es gilt, bei Streitigkeiten, die originale, deutsche Version**

**In the event of disputes, the original, German version shall be valid.**

Version 03

13.06.2017

Copyright © 2017 SoliDat GmbH

Dokument history

|  |  |  |
| --- | --- | --- |
| Version | vom | Änderungen |
| 01 | 30.05.2014 | Initial |
| 02 | 20.06.2014 | Adressen für Key / Value Paare |
| 03 | 13.06.2017 | Status Rückmeldung und ErrorMessage hinzugefügt, Änderung der Barcodelänge von 60 auf 100 |

**Achtung: Fehlerhandling hat sich geändert. Muss in der Implementierung berücksichtigt werden!**

Inhalt

1. General 3

2. Interface: Web Services 3

1.1 Introduction 3

1.2 Prerequisites 3

1.3 Interface description 3

1.4 Using the Services 5

1.5 Test 5

3. Schnittstelle: Kommunikation über SPS Datenblöcke 6

1.6 Introduction 6

1.7 Data Blocks 6

1.8 Process discription Machine (PLC) / Server 8

# General

A communication is required between the Shopfloor (MES) system and the PLC, which is

* the transfer of process parameters from MES to PLC
* The transfer of processing status and process data from PLC to MES

realized.

The technical basics for this communication are specified in this document.

A distinction is made between two types of communication provided by the MES system:

* WEB Services
* Direct PLC communication.

If a PC supported PLC system is available, the communication can take place via WEB Services, otherwise a PLC program must be created, which realizes the direct communication via PLC data blocks.

Both communication types are described in more detail in the following sections.

# Interface: Web Services

## Introduction

A Web service is a software application provided over a network for direct machine-to-machine interaction. Each Web service has a Uniform Resource Identifier (URI) that uniquely identifies it, and an interface description in machine-readable format (as an XML artifact, usually WSDL) that defines how to interact with the Web service.

## Prerequisites

* SOAP 1.2 compatible WebService Client
* Network connection to the target computer

## Interface description

**Method: RequestForCycle**

**Input parameter**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Discription** |
| Workplace | String | Workstation |

**Output parameter**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Discription** |
| RequestForCycleReturn | Struct |  |
| PartNo | String | Member of RequestForCycleReturn |
| ProcessParams | List | Member of RequestForCycleReturn |
| Key | String | Member of ProcessParams |
| Value | String | Member of ProcessParams |
| ErrorReasonCode | Integer | 0 = Ok  1 = No Part  2 = Part Not Applicable  3 = Other |
| ErrorMessage | String |  |

**Method: CycleResult**

**Input parameter**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Discription** |
| Workplace | String | Workstation |
| PartNo | String | Partnumber |
| Status | Integer | Possible values:   1. = CompleteWithSuccess 2. = CompleteWithFailure 3. = Aborted |
| ProcessData | List |  |
| Key | String | Member of ProcessData |
| Value | String | Member of ProcessData |

**Output parameter**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Discription** |
| ErrorReasonCode | Integer | 0 = Ok  1 = No Part  2 = Part Not Applicable  3 = Other |
| ErrorMessage | String |  |

## Using the Services

For each client system (machine control), a unique workstation ID (Workplace) is specified, which the client system supplies as a parameter for both methods.

If the client system wishes to execute a work step (cycle) on a workpiece labelled with RFID, it executes the RequestForCycle method and receives the individual part number (PartNo) and the process parameters (ProcessParams). At present, exactly one process parameter is defined, which contains the key barcode and a value that corresponds to the format of the picking record number in the current system in Hannover.

After successful confirmation, the client system executes the work step and reports the result by calling the CycleResult method. For this purpose it returns the individual part number (PartNo), the processing status (status, explanation see above) and any process data (measured or test values) as a ProcessData list.

Each element of the list has a key unique to the list and a value assigned to it.

Content errors that the server reports to the client are reported differently from 0 via an ErrorReasonCode. Examples of this are:

* When processing RequestForCycle, there is no single part number (->ErrorReasonCode 1 (No Part)).
* When processing RequestForCycle, it is determined that it is not permitted to execute the current production step on the individual part (->ErrorReasonCode 2 (Part Not applicable)).

In the event of an error, the ErrorMessage parameter is filled with plain text for the error message. If the error is successful, the ErrorReasonCode is filled with the value 0 (Ok).

Non-contentual (technical) errors are delivered via SOAP Exceptions.

## Test

The MES- test system can be reached at the following address

[***http://172.20.51.180:8180/MachineService/MachineService***](http://172.20.51.180:8180/MachineService/MachineService)

The WSDL is available at

[***http://172.20.51.180:8180/MachineService/MachineService?wsdl***](http://172.20.51.180:8180/MachineService/MachineService?wsdl)

# Schnittstelle: Kommunikation über SPS Datenblöcke

## Introduction

The interface between the client (S7 based PLC program) and the MES- server is established via interacting processes. The processes communicate via the memory content of S7 data blocks as specified below.

## Data Blocks

For each PLC client (PLC application program, machine)

* The PLC address <Addr>
* The number of the data block <DB>
* An offset <n> in bytes
* Live Check Support yes/no

is specified.

For offset 100, an entry DBX20 means DBX120.

It is possible to write to several machines via one PLC, each with separately specified communication blocks.

**Data types**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data type** | **Coding** | **Alignment** | **Notation** |
| String | ASCII, 0x00 terminiert | Word | DBW<offset> |
| I16 | 16 bit Signed integer | Word | DBW<offset> |
| Bit | Bit <n> in Unsigned 16 bit integer, 0 least significant |  | DBX<offset>.<bit> |

All strings are encoded in ASCII and terminated with 0x00.

Empty strings have a termination character 0x00 in the first byte.

Data Live Check, written by both processes

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Lenght (Bytes)** | **Discription** |
| DBX0.0 | Bit |  | LiveCheck, set by Server |

Data written by Server

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Lenght(Bytes)** | **Discription** |
| DBX2.0 | Bit |  | InitialServer |
| DBX2.1 | Bit |  | InitialPLCAck |
| DBX2.2 | Bit |  | ReadyForCycle |
| DBX2.3 | Bit |  | CycleRequestAck |
| DBX2.4 | Bit |  | CycleRequestReject |
| DBX2.5 | Bit |  | CycleAck |
| DBW4 | String | 40 | PartSerial |
| DBW44 | String | 100 | ParameterBarcode |
| DBW144 | I16 | 2 | ErrorReasonCode |
| DBW146 | String | 200 | ErrorMessage |

Data written by the PLC

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Lenght(Bytes)** | **Discription** |
| DBX500.0 | Bit |  | InitialPLC |
| DBX500.1 | Bit |  | InitialServerAck |
| DBX500.2 | Bit |  | CycleRequest |
| DBX500.3 | Bit |  | CycleSuccess |
| DBX500.4 | Bit |  | CycleFailure |
| DBX500.5 | Bit |  | CycleAbort |
| DBW504 | String | 40 | PartSerialPLC |
| DBW544 | I16 | 2 | ProcessDataCount <n> |
| **For every Index <i> from 0 to <n> - 1** | | | |
| DBW800 + <i>\*100 | String | 50 | Key for ProcessData Index <i> |
| DBW850 + <i>\*100 | String | 50 | Value for ProcessData Index <i> |

## Process discription Machine (PLC) / Server

The process description is illustrated in a separate document.

Notes:

- If Live/Check Support is not configured for a PLC, the server will set the LiveCheck bit every <t> seconds, but will not check whether it has been reset by the PLC.

- In the case of a "CycleRequestReject" from the server, the field "ErrorReasonCode" is assigned a value different from 0:

* 0: Ok
* 1: No Part
* 2: Part Not Applicable
* 3: Other