



DOCUMENTATION ISG-kernel

Functional description Service Interface

Short Description:
FCT-C31

Contents

1	Overview.....	4
2	Description	5
3	Supporting CNC cycles.....	10
4	Appendix	11
4.1	Suggestions, corrections and the latest documentation.....	11

List of figures

Fig. 1:	Process in Service Interface	6
---------	------------------------------------	---

1 Overview

Task

The Service Interface is a synchronous communication mechanism to invoke an external service from a CNC program.

The external service can execute a number of different tasks e.g.:

- Process and/or archive machine and log data.
- Send emails about the successful completion of a program.
- Read out wireless dial gauges and return measured values to the CNC program.



Attention

To use the Service Interface in practice, 2 software components must be developed: 1. the external service and 2. the CNC program to invoke the service. For this reason, this functional description is only suitable for experienced users or software developers.

Requirement

The Service Interface uses Beckhoff's ADS mechanism.

The requirements are therefore:

- TwinCAT 2.* with CNC Build \geq V.2.11.2030.01, or
- TwinCAT 3.* with CNC Build \geq V.3.1.3057.03

Mandatory note on references to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-linking.

2 Description

The Service Interface consists of 5 V.G. variables. Each CNC channel has its own Service Interface.

V.G.SERVICE.UUID

The string variable "V.G.SERVICE.UUID" contains a unique identifier to identify the external service. Several services can listen on the Service Interface at the same time but only the service with the matching UUID will respond to the request.

V.G.SERVICE.REQUEST

The string variable "V.G.SERVICE.REQUEST" contains the request. The Service Interface places no format restrictions on this variable. Only the CNC program and the external service can correctly assign and interpret this variable.

The length of this variable is limited to 128 bytes.

V.G.SERVICE.REQUEST_STATE

For a listening external service, a value unequal to 0 in "V.G.SERVICE.REQUEST_STATE" indicates that a request was issued. The Service Interface ensures that the service only starts its task when the value of the variable is unequal to 0. The rows

```
N00110 ; send request  
N00120 V.G.SERVICE.REQUEST_STATE = 1
```

ensure that a listening external service with matching UUID can start its work.

The variable "V.G.SERVICE.REQUEST_STATE" in the Service Interface has no other semantic meaning. To simplify processes, this Service Interface sets this variable to 0 when the external service has finished its task.

V.G.SERVICE.RESPONSE_STATE

The variable "V.G.SERVICE.RESPONSE_STATE" is the counterpart of the "V.G.SERVICE.REQUEST_STATE" for the response. Typically, this variable is written by the external service after it has finished its task. The CNC program waits for the variable V.G.SERVICE.RESPONSE to assume a particular value before continuing its process. For example, this variable can be used to return error codes or results. In the CNC program above, the rows

```
N00140 ; wait for response  
N00150 #WAIT FOR V.G.SERVICE.RESPONSE_STATE != 0
```

ensure that the CNC program waits until the external service finishes its task. The external service must be implemented accordingly, i.e. it must return a value of unequal to 0 when it ends.

Otherwise, the variable V.G.SERVICE.RESPONSE_STATE has no other semantic meaning in the Service Interface. To simplify processes, the Service Interface sets this variable to 0 when the CNC program writes the variable "V.G.SERVICE.REQUEST_STATE".

V.G.SERVICE.RESPONSE

The string variable "V.G.SERVICE.RESPONSE" contains the response of the external service. In analogy to "V.G.SERVICE.REQUEST", "V.G.SERVICE.RESPONSE" is only limited in its length (128 bytes), not in its format.

The task of the CNC program is to correctly interpret the variable.

Process

The typical process is as follows.

1. The external service is started and listens to the Service Interface from then on.
2. The CNC program prepares the request.
3. The CNC program writes the request on the Service Interface and waits for a response.
4. After the request is received, the external service executes its task.
5. After its task is finished, the external service prepares a response for the CNC program.
6. The external service writes the response back to the Service Interface.
7. The CNC program receives the response and continues its process.

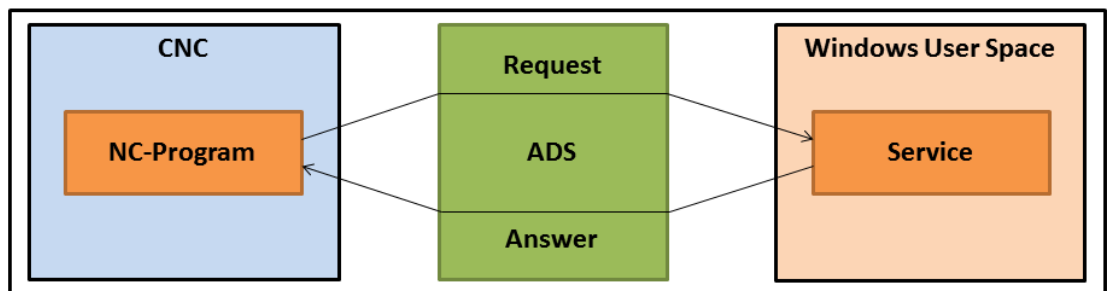


Fig. 1: Process in Service Interface



Programing Example

Process in Service Interface

```
N00010 ; move
N00020 G00 X10

N00040 ; wait for end of movement
N00050 #FLUSH CONTINUE

N00070 ; prepare request for Service Interface
N00080 V.G.SERVICE.UUID = "560AACBF-6335-44a9-895A-B27F3ED5E47"
N00090 V.G.SERVICE.REQUEST = "3"

N00110 ; send request
N00120 V.G.SERVICE.REQUEST_STATE = 1

N00140 ; wait for response
N00150 #WAIT FOR V.G.SERVICE.RESPONSE_STATE != 0

N00170 ; process response
N00180 #MSG SAVE EXCLUSIVE ["response = %s", V.G.SERVICE.RESPONSE]

N00200 ; continue processing
N00210 G00 X20

N00230 ; end
N00240 M30
```



Programing Example

Square root service - define service

```
//  
// Service function that interprets the request as numeral  
// and returns the square root of the numeral as response.  
//  
ISG_SERVICE_CODE squareRoot(ISG_SERVICE * service, char * request, ADS_UINT32 re-  
questState, char * response, size_t maxResponseSize, ADS_UINT32 * responseState)  
{  
    // interpret the request as numeral  
    double x = atof(request);  
  
    // check whether the numeral is non-negative  
    if (x >= 0)  
    {  
        // if yes, calculate the square root  
        double y = sqrt(x);  
  
        // convert root into string  
        // and write quotient in response buffer  
        sprintf_s(response, maxResponseSize, "%f", y);  
  
        // set response status to 1 to signal success  
        *responseState = 1;  
    }  
    else  
    {  
        // otherwise write error message in the  
        // response buffer  
        sprintf_s(response, maxResponseSize,  
            "ERROR: cannot calc square root of negative number %f", x);  
  
        // set response status to 99 to signal an  
        // error  
        *responseState = 99;  
    }  
  
    // the service function itself was not finished successfully  
    return ISG_SERVICE_CODE_OK;  
}
```




Programing Example

Start service, set up ADS connection and register service

```
//  
// Sets up connection to ADS and defines  
// and activates the square root service.  
//  
void exampleSquareRootService()  
{  
    // declare connection and service  
    ISG_SERVICE_CONNECTION connection;  
    ISG_SERVICE service;  
  
    // set up connection to ADS  
    // Assumption: SDA-Port = 552, COM-Port = 553  
    isgServiceConnectionCreateADSLocal(552, 553, &connection);  
  
    // define service  
    isgServiceCreate(  
  
        // this UUID must match with V.G.SERVICE.UUID  
        "560AACBF-6335-44a9-895A-8B27F3ED5E47",  
  
        // a simple description of the service  
        "a simple square root service",  
  
        // define actual service function  
        &squareRoot,  
  
        // define connection of service to ADS  
        &connection,  
  
        // service to be created, result buffer  
        &service  
    );  
  
    // poll service interface  
    isgServicePoll(&service);  
};
```

Software Development Kit (SDK)

ISG provides a simple SDK to create simple services quickly in C/C++.

The SDK contains:

- Header and library files for the Service Interface
- A sample project which implements the square root service
- A sample CNC program which invokes the service
- A batch service which listens to simple Windows batch files on the Service Interface; no C development is needed

For more information on the required ADS headers/libraries, refer to the text file "sdk/readme.txt" in the zip archive.

3 Supporting CNC cycles

SysServiceWait - Wait for external service

The SysServiceWait cycle is used in the NC program to wait for a response from the external service. Optionally a maximum waiting time can be specified.

Parameters	Description
@P1	Response code to denote continued wait The cycle waits until V.G.SERVICE.RESPONSE_STATE != @P1. optional, default value: 0
@P2	maximum waiting time, in milliseconds (optional)
@P3	ID of the timer used; must be specified if a waiting time is specified.



Programing Example

Start service, set up connection and register service

```

; set service request
V.G.SERVICE.UUID = "19719079-BA06-4740-BC02-DE5215211751"
V.G.SERVICE.REQUEST = "some-command"
V.G.SERVICE.REQUEST_STATE = 1

; wait for response, max 5s, end with timer ID 1
L CYCLE [NAME = "SysServiceWait.cyc", @P2 = 5000, @P3 = 1]
;...
;
M30/

```

4 Appendix

4.1 Suggestions, corrections and the latest documentation

Did you find any errors? Do you have any suggestions or constructive criticism? Then please contact us at documentation@isg-stuttgart.de. The latest documentation is posted in our Online Help (DE/EN):



QR code link: <https://www.isg-stuttgart.de/documentation-kernel/>

The link above forwards you to:

<https://www.isg-stuttgart.de/fileadmin/kernel/kernel-html/index.html>



Notice

Change options for favourite links in your browser;

Technical changes to the website layout concerning folder paths or a change in the HTML framework and therefore the link structure cannot be excluded.

We recommend you to save the above "QR code link" as your primary favourite link.

PDFs for download:

DE:

<https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads>

EN:

<https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads>

E-Mail: documentation@isg-stuttgart.de



© Copyright
ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart
All rights reserved
www.isg-stuttgart.de
support@isg-stuttgart.de

