SIMATIC NET

Program blocks for SIMATIC NET S7 CPs

Programming Manual

1 Preface
2 Overview and general information on handling
3 Program blocks for Industrial Ethernet
4 Program blocks for PROFINET IO (S7-300)
5 Program blocks for PROFINET CBA
6 Program blocks for PROFIBUS
7 Program blocks for PROFIBUS FMS
A Document history
B References

11/2015
C79000-G8976-C229-08
Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

<table>
<thead>
<tr>
<th>DANGER</th>
<th>indicates that death or severe personal injury will result if proper precautions are not taken.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WARNING</td>
<td>indicates that death or severe personal injury may result if proper precautions are not taken.</td>
</tr>
<tr>
<td>CAUTION</td>
<td>indicates that minor personal injury can result if proper precautions are not taken.</td>
</tr>
<tr>
<td>NOTICE</td>
<td>indicates that property damage can result if proper precautions are not taken.</td>
</tr>
</tbody>
</table>

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by personnel qualified for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

| WARNING          | Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed. |

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.
Preface

Target group and motivation
The interface to the communication services is implemented by readymade program blocks (FCs and FBs). This manual provides you with a full description of the program blocks for SIMATIC NET S7 CPs. It extends the descriptions in the online help of the configuration tools for STEP 7.

The manual is intended for programmers of STEP 7 programs and service personnel.

Structure of the manual
The manual is structured according to network types and communications services. Each program block is described in the following sections:

- Meaning
- Call interface
- Operating principle
- Explanation of the formal parameters
- Condition codes

These sections may included further specific information.

Scope of this manual
The current issue of the manual is valid for the following versions of the configuration software:

- STEP 7 V5.5 + SP4 + HF8
  Note the additions in the section “New in this release”.
- STEP 7 Professional V13.0 SP1

Note
STEP 7 - synonymous names
The term STEP 7 is used in later sections of this manual as a synonym for STEP 7 V5.5 or STEP 7 Professional

Note
CP types and communications services
Note that the program blocks each support certain communications services. Their use is therefore linked to the use of the CP types that support the particular communications service.
Meaning of "program block"

The term "program block" is used in the manual as a generic synonym for the following terms:

- **FC (function) / FB (function block)**
  
  Names used in STEP 7 V5.5 take into account the specific type of the program block. In STEP 7 Professional, these block types continue to be used unchanged, however, only the symbolic names are displayed in the libraries.

- **Instruction**
  
  Name used for system-internal program blocks in STEP 7 Professional.

**Note**

**Component of the products STEP 7 / STEP 7 Professional**

The contents of the libraries supplied with STEP 7 V5.5 and STEP 7 Professional may differ from each other.

The program block overviews in this manual specify the availability for the various device families S7-300 and S7-400. The overviews do not provide information about the availability in the various libraries of STEP 7 V5.5 or STEP 7 Professional.

New in this release

This manual release contains editorial corrections. You will find the essential changes in the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Change / addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP_CMD - universal program block for FTP services (Page 74)</td>
<td>Functional expansion of the block “FTP_CMD” for FTP client operation with the addition of the function “Passive FTP (client establishes connection)” The function is supported by the CP 443-1 Advanced with firmware version V3.2 in STEP 7 V5. In addition to the configuration software mentioned above STEP 7 V5.5, you also require the HSP 1105 and the program block library “SIMATIC NET CP” version V5.5.4.</td>
</tr>
</tbody>
</table>

You will find the changes to the previous manual information in the section Document history (Page 277).

Replaced manual issue

This manual replaces the manual release 10/2012.

**CP documentation in the Manual Collection (article number A5E00069051)**

The SIMATIC NET Manual Collection (DVD) with many each S7 CPs. This DVD is regularly updated and contains the manuals valid at the time it is created.
Information on the current program block versions (FCs/FBs)
Always use the latest block versions for new user programs. You will find information on the current block versions and the current program blocks on the Internet at the following address:


When replacing a CP, follow the instructions in the device-specific part of the device manual for your S7 CP.

Version history for program blocks and S7 CPs
The "Version History/Current Downloads for SIMATIC NET S7 CPs" provides information on all CPs available up to now for SIMATIC S7 (Industrial Ethernet, PROFIBUS and IE/PB Link) and the program blocks.

You will find the documents on the Internet at the following address:


Application examples on the topic of communication
You will a large selection of application examples and other documents on the pages of Siemens Industry Online Support at the following address:

Link: (https://support.industry.siemens.com/cs/ww/en/ps/15247/ae)

In the product tree and via the entry type you can further limit the area of application on this Internet page.

SIMATIC NET glossary
Explanations of many of the specialist terms used in this documentation can be found in the SIMATIC NET glossary.

You will find the SIMATIC NET glossary here:

- SIMATIC NET Manual Collection or product DVD
  The DVD ships with certain SIMATIC NET products.
- On the Internet under the following address:
# Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Preface</strong></td>
<td>3</td>
</tr>
<tr>
<td>1 Overview and general information on handling</td>
<td>13</td>
</tr>
<tr>
<td>1.1 Program blocks for Industrial Ethernet / PROFINET</td>
<td>13</td>
</tr>
<tr>
<td>1.2 Program blocks for PROFIBUS</td>
<td>15</td>
</tr>
<tr>
<td>1.3 Parameters for calling the program blocks</td>
<td>17</td>
</tr>
<tr>
<td>1.4 Parameters for CP and connection assignment (input parameters)</td>
<td>18</td>
</tr>
<tr>
<td>1.5 Parameters for specifying a CPU data area (input parameters)</td>
<td>19</td>
</tr>
<tr>
<td>1.6 Status information (output parameters)</td>
<td>19</td>
</tr>
<tr>
<td>2 Program blocks for Industrial Ethernet</td>
<td>21</td>
</tr>
<tr>
<td>2.1 Program blocks for open communications services (SEND/RECEIVE interface)</td>
<td>21</td>
</tr>
<tr>
<td>2.1.1 Overview of program blocks and their use</td>
<td>21</td>
</tr>
<tr>
<td>2.1.2 AG_SEND / AG_LSEND / AG_SSEND</td>
<td>25</td>
</tr>
<tr>
<td>2.1.2.1 Meaning and call - AG_SEND / AG_LSEND / AG_SSEND</td>
<td>25</td>
</tr>
<tr>
<td>2.1.2.2 How AG_SEND / AG_LSEND / AG_SSEND work</td>
<td>26</td>
</tr>
<tr>
<td>2.1.2.3 Explanation of the formal parameters - AG_SEND / AG_LSEND / AG_SSEND</td>
<td>29</td>
</tr>
<tr>
<td>2.1.2.4 AG_SEND, AG_LSEND and AG_SSEND status codes</td>
<td>30</td>
</tr>
<tr>
<td>2.1.3 AG_RECV / AG_LRECV / AG_SRECV</td>
<td>32</td>
</tr>
<tr>
<td>2.1.3.1 Meaning and call - AG_RECV / AG_LRECV / AG_SRECV</td>
<td>32</td>
</tr>
<tr>
<td>2.1.3.2 How AG_RECV / AG_LRECV / AG_SRECV work</td>
<td>33</td>
</tr>
<tr>
<td>2.1.3.3 Explanation of the formal parameters - AG_RECV / AG_LRECV / AG_SRECV</td>
<td>38</td>
</tr>
<tr>
<td>2.1.3.4 AG_RECV, AG_LRECV and AG_SRECV condition codes</td>
<td>39</td>
</tr>
<tr>
<td>2.2 Program blocks for access coordination with FETCH WRITE</td>
<td>41</td>
</tr>
<tr>
<td>2.2.1 Overview of program blocks and their use</td>
<td>41</td>
</tr>
<tr>
<td>2.2.2 AG_LOCK</td>
<td>43</td>
</tr>
<tr>
<td>2.2.2.1 Meaning and call - AG_LOCK</td>
<td>43</td>
</tr>
<tr>
<td>2.2.2.2 Explanation of the formal parameters - AG_LOCK</td>
<td>44</td>
</tr>
<tr>
<td>2.2.2.3 Condition codes of AG_LOCK</td>
<td>44</td>
</tr>
<tr>
<td>2.2.3 AG_UNLOCK</td>
<td>45</td>
</tr>
<tr>
<td>2.2.3.1 Meaning and call - AG_UNLOCK</td>
<td>45</td>
</tr>
<tr>
<td>2.2.3.2 Explanation of the formal parameters - AG_UNLOCK</td>
<td>46</td>
</tr>
<tr>
<td>2.2.3.3 Condition codes of AG_UNLOCK</td>
<td>46</td>
</tr>
<tr>
<td>2.3 Program blocks for connection and system diagnostics</td>
<td>47</td>
</tr>
<tr>
<td>2.3.1 AG_CNTRL</td>
<td>47</td>
</tr>
<tr>
<td>2.3.1.1 Meaning and call - AG_CNTRL</td>
<td>47</td>
</tr>
<tr>
<td>2.3.1.2 How AG_CNTRL works</td>
<td>49</td>
</tr>
<tr>
<td>2.3.1.3 Explanation of the formal parameters - AG_CNTRL</td>
<td>50</td>
</tr>
<tr>
<td>2.3.1.4 AG_CNTRL codes</td>
<td>51</td>
</tr>
<tr>
<td>2.3.1.5 Commands and job results - AG_CNTRL</td>
<td>52</td>
</tr>
<tr>
<td>2.3.2 AG_CNTEX</td>
<td>58</td>
</tr>
<tr>
<td>2.3.2.1 Meaning and call - AG_CNTEX</td>
<td>58</td>
</tr>
<tr>
<td>2.3.2.2 How AG_CNTEX works</td>
<td>60</td>
</tr>
</tbody>
</table>
## Table of contents

2.3.2.3  How the ping function works................................................................. 61
2.3.2.4  Explanation of the formal parameters - AG_CNTEX ............................. 63
2.3.2.5  AG_CNTEX codes .................................................................................. 64
2.3.2.6  Commands and job results - AG_CNTEX ............................................ 66
2.4  Program blocks for FTP services ................................................................. 73
  2.4.1  Overview of FTP ....................................................................................... 73
  2.4.2  FTP_CMD - universal program block for FTP services ......................... 74
        2.4.2.1  Meaning and call - FTP_CMD.......................................................... 75
        2.4.2.2  Input parameter - FTP_CMD ............................................................. 77
        2.4.2.3  Output parameters and status information - FTP_CMD ................. 82
        2.4.2.4  Migration of FC 40-44 to FTP_CMD ................................................ 85
  2.4.3  FTP_CONNECT ......................................................................................... 86
        2.4.3.1  Meaning and call - FTP_CONNECT ................................................... 86
        2.4.3.2  Explanation of the formal parameters - FTP_CONNECT ................. 87
  2.4.4  FTP_STORE ............................................................................................ 88
        2.4.4.1  Meaning and call - FTP_STORE........................................................ 88
        2.4.4.2  Explanation of the formal parameters - FTP_STORE ....................... 89
        2.4.5  FTP_RETRIEVE .................................................................................... 90
        2.4.5.1  Meaning and call - FTP_RETRIEVE .................................................. 90
        2.4.5.2  Explanation of the formal parameters - FTP_RETRIEVE ................. 91
        2.4.6  FTP_DELETE ....................................................................................... 92
        2.4.6.1  Meaning and call - FTP_DELETE ...................................................... 92
        2.4.6.2  Explanation of the formal parameter - FTP_DELETE ....................... 93
        2.4.7  FTP_QUIT .............................................................................................. 94
        2.4.7.1  Meaning and call - FTP_QUIT ........................................................... 94
        2.4.7.2  Explanation of the formal parameters - FTP_QUIT ......................... 95
        2.4.8  Parameters for CP and connection assignment (input parameters) .... 96
        2.4.9  Status information (output parameters) ............................................... 97
        2.4.10  Data block file DB .............................................................................. 99
        2.4.10.1  Structure of the data blocks (file DBs) for FTP services - FTP client mode .............................. 99
        2.4.10.2  Structure of the data blocks (file DBs) for FTP services - FTP server mode ............................ 102
  2.5  Program blocks for programmed connections and IP configuration ............ 104
  2.5.1  Operating principle .................................................................................. 105
  2.5.2  Procedure ............................................................................................... 107
  2.5.3  Configuration data block (CONF_DB) .................................................... 107
  2.5.4  Configuration data block - example ...................................................... 108
  2.5.5  Parameter field for system data (IP configuration) .................................. 112
  2.5.6  Parameter fields for connection types .................................................... 113
  2.5.6.1  Parameter field for TCP connection ................................................... 114
  2.5.6.2  Parameter field for UDP connection ................................................... 115
  2.5.6.3  Parameter field for an ISOonTCP connection ..................................... 116
  2.5.6.4  Parameter field for an Email connection ............................................. 117
  2.5.6.5  Parameter field for FTP connection .................................................... 118
  2.5.7  Subfield types ....................................................................................... 119
  2.5.8  IP_CONFIG - meaning and call .............................................................. 122
  2.5.9  How IP_CONFIG works ......................................................................... 124
  2.5.10  Explanation of the formal parameters - IP_CONFIG ............................. 126
  2.5.11  Reserved port numbers - IP_CONFIG .................................................... 127
  2.5.12  IP_CONFIG status codes ..................................................................... 127
  2.6  Program blocks for ERPC-CP ................................................................. 130
  2.6.1  LOGICAL_TRIGGER for the logical trigger ....................................... 130
# Table of contents

## 3 Program blocks for PROFINET IO (S7-300)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Overview of program blocks and their use</td>
<td>139</td>
</tr>
<tr>
<td>3.2 PROFINET IO - data transfer and interrupt evaluation</td>
<td>141</td>
</tr>
<tr>
<td>3.2.1 PNIO_SEND</td>
<td>141</td>
</tr>
<tr>
<td>3.2.1.1 Meaning and call - PNIO_SEND</td>
<td>141</td>
</tr>
<tr>
<td>3.2.1.2 Explanation of the formal parameters - PNIO_SEND</td>
<td>143</td>
</tr>
<tr>
<td>3.2.1.3 Condition codes of PNIO_SEND</td>
<td>147</td>
</tr>
<tr>
<td>3.2.2 PNIO_RECV</td>
<td>149</td>
</tr>
<tr>
<td>3.2.2.1 Meaning and call - PNIO_RECV</td>
<td>149</td>
</tr>
<tr>
<td>3.2.2.2 Explanation of the formal parameters - PNIO_RECV</td>
<td>151</td>
</tr>
<tr>
<td>3.2.2.3 Condition codes of PNIO_RECV</td>
<td>155</td>
</tr>
<tr>
<td>3.2.3 General characteristics of the FCs for PROFINET IO</td>
<td>157</td>
</tr>
<tr>
<td>3.2.4 Data consistency</td>
<td>158</td>
</tr>
<tr>
<td>3.2.5 Substitute values</td>
<td>159</td>
</tr>
<tr>
<td>3.2.6 PNIO_RW_REC</td>
<td>159</td>
</tr>
<tr>
<td>3.2.6.1 Meaning and call - PNIO_RW_REC</td>
<td>159</td>
</tr>
<tr>
<td>3.2.6.2 Explanation of the formal parameters - PNIO_RW_REC</td>
<td>160</td>
</tr>
<tr>
<td>3.2.6.3 Condition codes of PNIO_RW_REC</td>
<td>161</td>
</tr>
<tr>
<td>3.2.7 PNIO_ALARM</td>
<td>162</td>
</tr>
<tr>
<td>3.2.7.1 Meaning and call - PNIO_ALARM</td>
<td>163</td>
</tr>
<tr>
<td>3.2.7.2 Explanation of the formal parameters - PNIO_ALARM</td>
<td>164</td>
</tr>
<tr>
<td>3.2.7.3 Condition codes of PNIO_ALARM</td>
<td>166</td>
</tr>
<tr>
<td>3.3 PROFIenergy</td>
<td>167</td>
</tr>
<tr>
<td>3.3.1 PROFIenergy program blocks for the CP 300</td>
<td>168</td>
</tr>
<tr>
<td>3.3.2 PE_START_END_CP</td>
<td>170</td>
</tr>
<tr>
<td>3.3.2.1 Meaning and call - PE_START_END_CP</td>
<td>170</td>
</tr>
<tr>
<td>3.3.2.2 Explanation of the formal parameters of PE_START_END_CP</td>
<td>171</td>
</tr>
<tr>
<td>3.3.2.3 Condition codes of PE_START_END_CP</td>
<td>172</td>
</tr>
<tr>
<td>3.3.3 PE_CMD_CP</td>
<td>173</td>
</tr>
<tr>
<td>3.3.3.1 Meaning and call - PE_CMD_CP</td>
<td>173</td>
</tr>
<tr>
<td>3.3.3.2 Explanation of the formal parameters of PE_CMD_CP</td>
<td>175</td>
</tr>
<tr>
<td>3.3.3.3 Condition codes of PE_CMD_CP</td>
<td>178</td>
</tr>
<tr>
<td>3.3.4 Response data</td>
<td>179</td>
</tr>
<tr>
<td>3.3.5 PE_I_DEV_CP</td>
<td>186</td>
</tr>
<tr>
<td>3.3.5.1 Meaning and call - PE_I_DEV_CP</td>
<td>186</td>
</tr>
<tr>
<td>3.3.5.2 Explanation of the formal parameters of PE_I_DEV_CP</td>
<td>187</td>
</tr>
<tr>
<td>3.3.5.3 Condition codes of PE_I_DEV_CP</td>
<td>188</td>
</tr>
<tr>
<td>3.3.6 Supplementary program blocks for PE_I_DEV_CP</td>
<td>189</td>
</tr>
<tr>
<td>3.3.6.1 Overview of the FCs</td>
<td>189</td>
</tr>
<tr>
<td>3.3.6.2 Interconnection of the FCs with PE_I_DEV_CP</td>
<td>190</td>
</tr>
<tr>
<td>3.3.6.3 Common parameters of the FCs</td>
<td>191</td>
</tr>
<tr>
<td>3.3.6.4 Individual parameters of the FCs</td>
<td>191</td>
</tr>
<tr>
<td>3.3.7 DS3_WRITE_CP / PE_DS3_Write_ET200S_CP</td>
<td>196</td>
</tr>
<tr>
<td>3.3.7.1 Meaning and call - DS3_WRITE_CP</td>
<td>197</td>
</tr>
<tr>
<td>3.3.7.2 Explanation of the formal parameters of DS3_WRITE_CP</td>
<td>198</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.3.7.3 Condition codes of DS3_WRITE_CP</td>
<td>199</td>
</tr>
<tr>
<td>3.4 Configuration limits / resources required for the program blocks (PROFINET)</td>
<td>199</td>
</tr>
<tr>
<td>4 Program blocks for PROFINET CBA</td>
<td>201</td>
</tr>
<tr>
<td>4.1 PN_InOut / PN_InOut_Fast - meaning and call</td>
<td>201</td>
</tr>
<tr>
<td>4.2 Explanation of the formal parameters - PN_InOut / PN_InOut_Fast</td>
<td>203</td>
</tr>
<tr>
<td>4.3 Condition codes of the PN_InOut and PN_InOut_Fast blocks</td>
<td>203</td>
</tr>
<tr>
<td>4.4 Timedriven PN_InOut / PN_InOut_Fast call - recommendation on application</td>
<td>206</td>
</tr>
<tr>
<td>5 Program blocks for PROFIBUS</td>
<td>207</td>
</tr>
<tr>
<td>5.1 Program blocks for open communications services (SEND/RECEIVE interface)</td>
<td>207</td>
</tr>
<tr>
<td>5.1.1 Overview of uses</td>
<td>207</td>
</tr>
<tr>
<td>5.1.2 AG_SEND / AG_LSEND</td>
<td>209</td>
</tr>
<tr>
<td>5.1.2.1 Meaning and call - AG_SEND / AG_LSEND</td>
<td>209</td>
</tr>
<tr>
<td>5.1.2.2 How AG_SEND / AG_LSEND work</td>
<td>211</td>
</tr>
<tr>
<td>5.1.2.3 Explanation of the formal parameters - AG_SEND / AG_LSEND</td>
<td>212</td>
</tr>
<tr>
<td>5.1.2.4 AG_SEND and AG_LSEND condition codes</td>
<td>213</td>
</tr>
<tr>
<td>5.1.3 AG_RECV / AG_LRECV</td>
<td>215</td>
</tr>
<tr>
<td>5.1.3.1 Meaning and call - AG_RECV / AG_LRECV</td>
<td>215</td>
</tr>
<tr>
<td>5.1.3.2 How AG_RECV / AG_LRECV work</td>
<td>217</td>
</tr>
<tr>
<td>5.1.3.3 Explanation of the formal parameters - AG_RECV / AG_LRECV</td>
<td>218</td>
</tr>
<tr>
<td>5.1.3.4 AG_RECV and AG_LRECV condition codes</td>
<td>219</td>
</tr>
<tr>
<td>5.2 Program blocks for DP (distributed I/O) with S7-300</td>
<td>221</td>
</tr>
<tr>
<td>5.2.1 Overview of uses</td>
<td>221</td>
</tr>
<tr>
<td>5.2.2 DP_SEND</td>
<td>222</td>
</tr>
<tr>
<td>5.2.2.1 Meaning and call - DP_SEND</td>
<td>222</td>
</tr>
<tr>
<td>5.2.2.2 How DP_SEND works</td>
<td>223</td>
</tr>
<tr>
<td>5.2.2.3 Explanation of the formal parameters - DP_SEND</td>
<td>224</td>
</tr>
<tr>
<td>5.2.2.4 DP_SEND condition codes</td>
<td>225</td>
</tr>
<tr>
<td>5.2.3 DP_RECV</td>
<td>226</td>
</tr>
<tr>
<td>5.2.3.1 Meaning and call - DP_RECV</td>
<td>226</td>
</tr>
<tr>
<td>5.2.3.2 How DP_RECV works</td>
<td>227</td>
</tr>
<tr>
<td>5.2.3.3 Explanation of the formal parameters - DP_RECV</td>
<td>229</td>
</tr>
<tr>
<td>5.2.3.4 DP_RECV condition codes</td>
<td>230</td>
</tr>
<tr>
<td>5.2.3.5 DPSTATUS - DP_RECV</td>
<td>231</td>
</tr>
<tr>
<td>5.2.4 DP_DIAG</td>
<td>233</td>
</tr>
<tr>
<td>5.2.4.1 Meaning and call - DP_DIAG</td>
<td>233</td>
</tr>
<tr>
<td>5.2.4.2 How DP_DIAG works</td>
<td>234</td>
</tr>
<tr>
<td>5.2.4.3 Explanation of the formal parameters - DP_DIAG</td>
<td>236</td>
</tr>
<tr>
<td>5.2.4.4 Job types - DP_DIAG</td>
<td>237</td>
</tr>
<tr>
<td>5.2.4.5 Ring buffer for diagnostics data - DP_DIAG</td>
<td>239</td>
</tr>
<tr>
<td>5.2.4.6 DP_DIAG codes</td>
<td>239</td>
</tr>
<tr>
<td>5.2.5 DP_CTRL</td>
<td>242</td>
</tr>
<tr>
<td>5.2.5.1 Meaning and call - DP_CTRL</td>
<td>242</td>
</tr>
<tr>
<td>5.2.5.2 How DP_CTRL works</td>
<td>243</td>
</tr>
<tr>
<td>5.2.5.3 Explanation of the formal parameters - DP_CTRL</td>
<td>245</td>
</tr>
<tr>
<td>5.2.5.4 Job types - DP_CTRL</td>
<td>247</td>
</tr>
<tr>
<td>5.2.5.5 Command mode and group select - DP_CTRL</td>
<td>249</td>
</tr>
<tr>
<td>5.2.5.6 DP_CTRL condition codes</td>
<td>250</td>
</tr>
</tbody>
</table>
# Table of contents

5.3 Configuration limits / resources required for the program blocks (PROFIBUS) ................. 253

6 Program blocks for PROFIBUS FMS .................................................................................. 255

6.1 Overview of uses ............................................................................................................. 255
6.2 FMS block parameters .................................................................................................... 256
6.3 IDENTIFY .......................................................................................................................... 259
6.3.1 Meaning and call - IDENTIFY ....................................................................................... 259
6.3.2 How IDENTIFY works .................................................................................................. 260
6.4 READ ................................................................................................................................ 261
6.4.1 Meaning and call - READ .............................................................................................. 261
6.4.2 How READ works ........................................................................................................ 262
6.5 REPORT ............................................................................................................................. 263
6.5.1 Meaning and call - REPORT ......................................................................................... 263
6.5.2 How REPORT works ................................................................................................... 264
6.6 STATUS ............................................................................................................................. 265
6.6.1 Meaning and call - STATUS ......................................................................................... 265
6.6.2 How STATUS works .................................................................................................... 266
6.7 WRITE ............................................................................................................................... 268
6.7.1 Meaning and call - WRITE .......................................................................................... 268
6.7.2 How WRITE works ...................................................................................................... 269
6.8 Condition codes and error messages - FMS blocks ......................................................... 270
6.8.1 Error detected locally .................................................................................................. 271
6.8.2 Errors detected by FMS partner ................................................................................... 274
6.9 Quantity framework / resource requirements of FBs (PROFIBUS FMS) ..................... 276

A Document history ................................................................................................................ 277

B References .......................................................................................................................... 281

B.1 On configuring, commissioning and using the CP ......................................................... 281
B.2 On programming ............................................................................................................. 282

Index ........................................................................................................................................ 283
<table>
<thead>
<tr>
<th>Table of contents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Overview and general information on handling

1.1 Program blocks for Industrial Ethernet / PROFINET

How supplied - block library

The SIMATIC NET FCs program blocks are supplied with the STEP 7 configuration software unless indicated otherwise.

Note

Component of the products STEP 7 / STEP 7 Professional

The contents of the libraries supplied with STEP 7 V5.5 and STEP 7 Professional may differ from each other.

The following list shows the block numbers as they are supplied.

Under SIMATIC_NET_CP, you can also see which folders contain blocks. Please note that you must use different program blocks for the S7300 and S7400 (separate libraries).

<table>
<thead>
<tr>
<th>Communication service / functional area</th>
<th>Program block</th>
<th>Library for SIMATIC NET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SIMATIC_NET_CP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CP 300</td>
</tr>
<tr>
<td>SEND / RECEIVE (open communications services)</td>
<td>AG_SEND</td>
<td>FC5</td>
</tr>
<tr>
<td></td>
<td>AG_LSEND</td>
<td>FC50</td>
</tr>
<tr>
<td></td>
<td>AG_SSEND</td>
<td>FC53</td>
</tr>
<tr>
<td></td>
<td>AG_RECV</td>
<td>FC6</td>
</tr>
<tr>
<td></td>
<td>AG_LRECV</td>
<td>FC60</td>
</tr>
<tr>
<td></td>
<td>AG_SRECV</td>
<td>FC63</td>
</tr>
<tr>
<td></td>
<td>AG_LOCK</td>
<td>FC7</td>
</tr>
<tr>
<td></td>
<td>AG_UNLOCK</td>
<td>FC8</td>
</tr>
<tr>
<td></td>
<td>AG_CNTRL</td>
<td>FC10</td>
</tr>
<tr>
<td></td>
<td>AG_CNTEX</td>
<td>FB10</td>
</tr>
<tr>
<td>Programmed communications connections and IP configuration</td>
<td>IP_CONFIG</td>
<td>FB55</td>
</tr>
<tr>
<td>S7 communication 4)</td>
<td>BSEND</td>
<td>FB12</td>
</tr>
<tr>
<td></td>
<td>BRCV</td>
<td>FB13</td>
</tr>
<tr>
<td></td>
<td>PUT</td>
<td>FB15</td>
</tr>
<tr>
<td></td>
<td>GET</td>
<td>FB14</td>
</tr>
<tr>
<td></td>
<td>USEND</td>
<td>FB8</td>
</tr>
<tr>
<td></td>
<td>URRCV</td>
<td>FB9</td>
</tr>
<tr>
<td></td>
<td>C_CNTRL</td>
<td>FC62</td>
</tr>
</tbody>
</table>
Overview and general information on handling

1.1 Program blocks for Industrial Ethernet / PROFINET

<table>
<thead>
<tr>
<th>Communication service / functional area</th>
<th>Program block</th>
<th>Library for SIMATIC NET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SIMATIC_NET_CP</td>
<td>CP 300</td>
</tr>
<tr>
<td>FTP (advanced CPs)</td>
<td>FTP_CMD</td>
<td>FB40</td>
</tr>
<tr>
<td></td>
<td>FTP_CONNECT</td>
<td>FC40</td>
</tr>
<tr>
<td></td>
<td>FTP_STORE</td>
<td>FC41</td>
</tr>
<tr>
<td></td>
<td>FTP_RETRIEVE</td>
<td>FC42</td>
</tr>
<tr>
<td></td>
<td>FTP_DELETE</td>
<td>FC43</td>
</tr>
<tr>
<td></td>
<td>FTP_QUIT</td>
<td>FC44</td>
</tr>
<tr>
<td>PROFINET CBA</td>
<td>PN_InOut 1)</td>
<td>FB88 1)</td>
</tr>
<tr>
<td></td>
<td>PN_InOut_Fast 1)</td>
<td>FB90 1)</td>
</tr>
<tr>
<td>PROFINET IO</td>
<td>PNIO_SEND</td>
<td>FC11</td>
</tr>
<tr>
<td></td>
<td>PNIO_RECV</td>
<td>FC12</td>
</tr>
<tr>
<td></td>
<td>PNIO_RW_REC</td>
<td>FB52</td>
</tr>
<tr>
<td></td>
<td>PNIO_ALARM</td>
<td>FB54</td>
</tr>
<tr>
<td></td>
<td>PE_START_END_CP</td>
<td>FB85</td>
</tr>
<tr>
<td></td>
<td>PE_CMD_CP</td>
<td>FB86</td>
</tr>
<tr>
<td></td>
<td>PE_I_DEV_CP</td>
<td>FB87</td>
</tr>
<tr>
<td></td>
<td>DS3_WRITE_CP (STEP 7 V5.5)</td>
<td>FB53</td>
</tr>
<tr>
<td></td>
<td>PE_DS3_Write_ET200S_CP 5)</td>
<td>FB88</td>
</tr>
</tbody>
</table>

Legend:

1) FB88/FB90 is supplied along with the engineering tool SIMATIC iMap and is entered in the PROFINET system library when you install the STEP 7 addon.
2) Not to be used with the current CPs and no longer part of the current "SIMATIC_NET_CP" library.
3) Depending on the CP type
4) Described in the STEP 7 documentation
5) You will find the description of the program block PE_DS3_Write_ET200S_CP for STEP 7 Professional later in this document with the description of the program block DS3_WRITE_CP for STEP 7 V5.5 that has the same functionality.

Which block version should I use?

The following descriptions also include information on differences in behavior between the various block versions. Please check and note the version identifiers of the blocks you are using.

The SIMATIC NET block libraries installed with STEP 7 contain the block versions that were current at the time of the STEP 7 release.

Note

We recommend that you always use the latest block versions for all module types.

You will find information on the current block versions and the current blocks to download from the Internet in Siemens Industry Online Support at the following address:


This recommendation assumes that you are using the latest firmware for the particular module type.
Program blocks when modules are replaced

Module replacement means the replacement of a module with another module that may be a more recent version.

Note
Remember that if you replace a module, you must only use the blocks permitted for the configured CP type in the user program.
We recommend that you always use the latest block versions for all module types.
This recommendation assumes that you are using the latest firmware for the particular module type.

The specific manuals contain information on the compatibility of the S7-CPs and the corresponding program blocks.

1.2 Program blocks for PROFIBUS

How supplied - block library

The SIMATIC NET FCs program blocks are supplied with the STEP 7 configuration software unless indicated otherwise.

Note
Component of the products STEP 7 / STEP 7 Professional
The contents of the libraries supplied with STEP 7 V5.5 and STEP 7 Professional may differ from each other.

The following list shows the block numbers as they are supplied.
Under SIMATIC_NET_CP, you can also see which folders contain blocks. Please note that you must use different program blocks for the S7300 and S7400 (separate libraries).

<table>
<thead>
<tr>
<th>Communication service / functional area</th>
<th>Program block</th>
<th>Library for SIMATIC NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFIBUS DP</td>
<td></td>
<td>SIMATIC_NET_CP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CP 300</td>
</tr>
<tr>
<td>PROFIBUS DP</td>
<td>DP_SEND</td>
<td>FC1</td>
</tr>
<tr>
<td></td>
<td>DP_RECV</td>
<td>FC2</td>
</tr>
<tr>
<td></td>
<td>DP_DIAG</td>
<td>FC3</td>
</tr>
<tr>
<td></td>
<td>DP_CTRL</td>
<td>FC4</td>
</tr>
<tr>
<td>SEND / RECEIVE (open communications services)</td>
<td>AG_SEND</td>
<td>FC5</td>
</tr>
<tr>
<td></td>
<td>AG_LSEND</td>
<td>FC50</td>
</tr>
<tr>
<td></td>
<td>AG_RECV</td>
<td>FC6</td>
</tr>
<tr>
<td></td>
<td>AG_LRECV</td>
<td>FC60</td>
</tr>
</tbody>
</table>
Overview and general information on handling

1.2 Program blocks for PROFIBUS

<table>
<thead>
<tr>
<th>Communication service / functional area</th>
<th>Program block</th>
<th>Library for SIMATIC NET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SIMATIC_NET_CP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CP 300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CP 400</td>
</tr>
<tr>
<td>S7 communication ¹)</td>
<td>BSEND</td>
<td>FB12</td>
</tr>
<tr>
<td></td>
<td>BRCV</td>
<td>FB13</td>
</tr>
<tr>
<td></td>
<td>PUT</td>
<td>FB15</td>
</tr>
<tr>
<td></td>
<td>GET</td>
<td>FB14</td>
</tr>
<tr>
<td></td>
<td>USEND</td>
<td>FB8</td>
</tr>
<tr>
<td></td>
<td>URCV</td>
<td>FB9</td>
</tr>
<tr>
<td></td>
<td>C_CNTRL</td>
<td>FC62</td>
</tr>
<tr>
<td>PROFIBUS FMS</td>
<td>IDENTIFY</td>
<td>FB2</td>
</tr>
<tr>
<td></td>
<td>READ</td>
<td>FB3</td>
</tr>
<tr>
<td></td>
<td>REPORT</td>
<td>FB4</td>
</tr>
<tr>
<td></td>
<td>STATUS</td>
<td>FB5</td>
</tr>
<tr>
<td></td>
<td>WRITE</td>
<td>FB6</td>
</tr>
</tbody>
</table>

¹) Described in the STEP 7 documentation. Accordingly, SFBs to be used for the S7-400 are available in the STEP 7 library.

2) Can be used but has no special function with PROFIBUS.

Which block version should I use?

The following descriptions also include information on differences in behavior between the various block versions. Please check and note the version identifiers of the blocks you are using.

The SIMATIC NET block libraries installed with STEP 7 contain the block versions that were current at the time of the STEP 7 release.

Note

We recommend that you always use the latest block versions for all module types.

You will find information on the current block versions and the current blocks to download from the Internet in Customer Support under entry ID:


This recommendation assumes that you are using the latest firmware for the particular module type.
Program blocks when modules are replaced

Module replacement means the replacement of a module with another module that may be a more recent version.

Note

Remember that if you replace a module, you must only use the blocks permitted for the configured CP type in the user program.

We recommend that you always use the latest block versions for all module types.

This recommendation assumes that you are using the latest firmware for the particular module type.

The manuals contain information on the compatibility of the S7-CPs and the corresponding program blocks.

1.3 Parameters for calling the program blocks

General notes on calling and assigning parameters

Before describing the program blocks in detail, a few general comments on calling and setting parameters for program blocks will be useful at this point.

The general information below applies to the following parameter groups that exist for all program blocks:

- Parameters for CP and connection assignment (input parameters)
- Parameters for specifying a CPU data area (input parameters)
- Status information (output parameters)

NOTICE

Calling communication blocks for an S7300

The communication blocks for S7-300 (SIMATIC NET block libraries for S7300 in STEP 7) must not be called in more than one priority class! If, for example, you call a communication block in OB1 and in OB35, block execution could be interrupted by the higher priority OB.

If you call blocks in more than one OB, you must write your program so that a communication block that is currently executing cannot be interrupted by another communication block (for example by disabling/enabling SFC interrupts).
1.4 Parameters for CP and connection assignment (input parameters)

When you call a program block, you transfer the module start address of the S7 CP in the CPLADDR or LADDR parameter. You will find the module start address of the S7 CP in the configuration of the CP in "Address/Input" parameter.

With connection-oriented jobs, you must also reference the connection to be used by its connection ID. You will find this in the properties dialog of the connection under "Block parameters" (refer to the information in NetPro).

Automatically adopting block parameters (described here for STEP 7 V5.5)

To ensure correct parameter settings for the block calls, the LAD/STL/FBD editor in STEP 7 provides you with the option of accepting all the relevant parameters automatically from the hardware configuration (HW Config) and from the connection configuration (NetPro).

When assigning the parameters for the block in the user program, follow the steps outlined below:

1. Select the block call and its block parameters;
2. Right-click and select the menu command "Connections...".
3. Depending on the block type, you can now select the connection and/or module intended for the block from a list.
4. Confirm your selection; as far as possible, the available parameter values are entered in the block call.

Response to incorrect addresses

If the S7 CPU cannot communicate with the PROFIBUS CP using the specified module start address or cannot identify it as a CP, the errors described below result.

<table>
<thead>
<tr>
<th>Cause</th>
<th>Reaction / code</th>
</tr>
</thead>
<tbody>
<tr>
<td>No module can be addressed or identified at the specified CP address.</td>
<td>The CPU remains in STOP with system error state; in this case, evaluate the diagnostic buffer of the CPU.</td>
</tr>
</tbody>
</table>
| The CP address points to a different module type.                    | Possible error code in the STATUS parameter of the communication block:  
  8184H System error  
  80B0H The module does not recognize the data record.  
  80C0H The data record cannot be read.  
  80C3H Resources (memory) occupied.  
  80D2H Logical base address is wrong. |

Note

If you inadvertently address not a CP but another module type, errors occur that cannot be indicated by the error messages of the program blocks themselves.
1.5 Parameters for specifying a CPU data area (input parameters)

Specifying the data area on the CPU

When you call a program block, you transfer the address and length of the data area on the CPU in which the user data is available or will be stored or which can contain further parameter information.

The ANY pointer data type is used to address this area. You will find more detailed information on this data type in the STEP 7 online help.

1.6 Status information (output parameters)

Evaluating status codes

For status evaluation, the following parameters must be evaluated in the user program:

- **DONE** or **NDR**
  These parameters (DONE with send jobs and NDR with receive jobs) signal (successful) completion of the job.

- **ERROR**
  This indicates that the job could not be executed errorfree.

- **STATUS**
  This parameter supplies detailed information about the execution of the job. Status codes can be returned during execution of the job (DONE=0 and ERROR=0).

**Note**

Remember that the status codes DONE, NDR, ERROR, STATUS are updated at each block call.

Status codes during CP startup

With a complete restart or restart of the Ethernet CP (for example after activating a switch on the module), the output parameters of the FC are reset as follows:

- **DONE** = 0
- **NDR** = 0
- **ERROR** = 0
- **STATUS** =
  - 8180h for AG_RECV / AG_LRECV
  - 8181h for AG_SRECV
  - 8181h for AG_SEND / AG_LSEND / AG_SSEND
Overview and general information on handling

1.6 Status information (output parameters)
2.1 Program blocks for open communications services (SEND/RECEIVE interface)

2.1.1 Overview of program blocks and their use

Overview

The following program blocks are available for transferring data on the SEND/RECEIVE interface:

<table>
<thead>
<tr>
<th>Program block</th>
<th>Can be used with 1)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S7-300</td>
<td>S7-400</td>
</tr>
<tr>
<td>AG_SEND (FC5)</td>
<td>x</td>
<td>for sending data</td>
</tr>
<tr>
<td>AG_RECV (FC6)</td>
<td>x</td>
<td>for receiving data</td>
</tr>
<tr>
<td>AG_LSEND (FC50)</td>
<td></td>
<td>for sending data</td>
</tr>
<tr>
<td>AG_LRECV (FC60)</td>
<td>x</td>
<td>for receiving data</td>
</tr>
<tr>
<td>AG_SSEND (FC53)</td>
<td>x</td>
<td>for sending data</td>
</tr>
<tr>
<td>AG_SRECV (FC63)</td>
<td>x</td>
<td>for receiving data</td>
</tr>
</tbody>
</table>

1) Notes on the program blocks for an S7300 and S7400

- **S7300:**
  - With the latest versions of the Ethernet CPs, only program blocks AG_SEND and AG_RECV are used; data with a length of up to 8192 bytes can be transferred.
  - With S7-300 CPs (up to 6GK7 343-1EX10-0XE0 with firmware version V2.2), use FC60 on TCP connections instead of FC6. For the CP 343-1 (EX10), you can use FC5/FC6 up to block version V3.0.

- **S7-400:**
  - With AG_SEND / AG_RECV program blocks, the data length per job is restricted to <=240 bytes. Longer data records (up to 8192 bytes) can be transferred with FCS AG_LSEND or AG_LRECV.
  - The AG_SSEND and AG_SRECV program blocks are for accelerated transfer of data by using optimized block communication between CPU and CP in the S7 station. The fast communication has no effect on LAN communication.
  - On an S7-400, FC6 cannot be used on TCP connections but only FC60 or FC63.
Further information

Please check the supported data area for the S7-CP you are using in the manual for the specific device. You will find an overview of the versions of the FCs/FBs in the SIMATIC NET block history.

Application

The following diagram illustrates the use of the program blocks described here for bi-directional data transfer on a configured connection.

Note

Unless specifically stated otherwise, the information on this and the following pages refers to the AG_SEND / AG_LSEND / AG_SSEND or AG_RECV / AG_LRECV / AG_SRECV blocks.

Sample programs

Please note that the following sample programs are also available on the Internet under the following entry ID

- Sample program for the SEND/RECEIVE interface with the blocks FC5 (AG_SEND) and FC6 (AG_RECV) for S7-300:

- Sample program for the SEND/RECEIVE interface with the functions FC50 (AG_LSEND) and FC60 (AG_LRECV) for S7-400:
Specifying the data area on the CPU

When you call an FC, you transfer the address and length of the data area in the CPU. Remember, that the maximum length of the data area depends on the block type and block version being used.

- **AG_SEND and AG_RECV**
  Up to version V3.0 of these blocks, a maximum of 240 bytes can be sent or received. The current block versions allow a data area of up to 8192 bytes for an S7-300. With an S7-400, the FCs AG_LSEND / AG_LRECV must still be used for larger data areas.

- **AG_LSEND / AG_LRECV**
  Using the CPs of the S7400 and with earlier versions of the S7300, larger data areas can only be transferred with the FCs AG_LSEND or AG_LRECV. Please check the length of the data area in the product information of the CP.

- **AG_SSEND / AG_SRECV**
  With CPs of the S7–400 that support PROFINET communication in conjunction with CPUs as of version 5.1, data can be transferred at higher transmission speeds with the FCs AG_SSEND or AG_SRECV (does not apply to the CP 443–1 Advanced 6GK7 443–1EX41–0XE0).
  You can check which CP types are supported by CPUs as of version 5.1 in the manual of your CP (Section "Requirements for use").

The following table shows the limit values of the various connection types.

<table>
<thead>
<tr>
<th>FC</th>
<th>ISO transport</th>
<th>ISO-on-TCP</th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG_LSEND (S7-400)</td>
<td>8192 bytes</td>
<td>8192 bytes</td>
<td>8192 bytes</td>
<td>2048 bytes</td>
</tr>
<tr>
<td>AG_SEND (S7-300)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AG_SEND (S7-400)</td>
<td>240 bytes</td>
<td>240 bytes</td>
<td>240 bytes</td>
<td>240 bytes</td>
</tr>
<tr>
<td>AG_LRECV (S7-400)</td>
<td>8192 bytes</td>
<td>8192 bytes</td>
<td>8192 bytes</td>
<td>2048 bytes</td>
</tr>
<tr>
<td>AG_RECV (S7-400)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AG_SSEND (S7-400)</td>
<td>1452 bytes</td>
<td>1452 bytes</td>
<td>1452 bytes</td>
<td>1452 bytes</td>
</tr>
<tr>
<td>AG_SRECV (S7-400)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note**

For information on the length of the data area you can transfer with older versions of the Ethernet CPs, refer to the product information / manual of the Ethernet CP you are using.

**Use without job header**

On specified connections, the address and job parameters are specified by the connection configuration. The user program only provides the user data in the UDP data area when sending with AG_SEND / AG_LSEND / AG_SSEND or receives the data with AG_RECV / AG_LRECV / AG_SRECV.
Use with header

Free UDP connections require a job header in the user data area. The following schematic illustrates the structure of the job buffer and the meaning and location (high byte / low byte) of the parameters in the job header.

- In the diagram (entries in hexadecimal) the following IP address is assumed as an example: 142.11.40.35;
- For the port address 1003, the following would be entered: For high byte: 03H; For low byte: EBH.
- The user data area can be up to 2048 bytes. Up to 2042 bytes of user data can be transferred. 6 bytes are reserved for the job header. Please note that the data length specified in the block call (LEN parameter) must include the header and the user data!

Change call parameters only after job confirmation

Note

Once the job has been triggered, you can only change the call parameters of the call interface of the AG_SEND or AG_RECV program blocks after the FC has confirmed completion of the job with DONE=1 or with ERROR=1.

If you do not keep to this rule, it is possible that the job will be aborted with an error.

Status display on the FC call interface; Special case with FC versions (only for S7-300) *)

With the FCs AG_SEND (FC5) and AG_RECV (FC6), you will receive the codes shown below in the following situations:
- CP is in STOP;
- Connection is not configured;
2.1 Program blocks for open communications services (SEND/RECEIVE interface)

- Connection is not established
- Connection is aborted;

Codes:
- **AG_SEND:**
  - DONE=0; ERROR=1; Status=8183H
- **AG_RECV:**
  - DONE=0; ERROR=0; Status=8180H
  - or
  - DONE=0; ERROR=1; Status=8183H

*) applies to FCs as of version 4.0

2.1.2 AG_SEND / AG_LSEND / AG_SSEND

2.1.2.1 Meaning and call - AG_SEND / AG_LSEND / AG_SSEND

Meaning of the block
The program block AG_SEND / AG_LSEND / AG_SSEND passes data to the Ethernet CP for transfer over a configured connection.

The selected data area can be a memory bit area or a data block area.

Errorfree execution of the function is indicated when the entire user data area could be sent over Ethernet.

Note:
Unless otherwise stated, all the following information applies equally to the FCs AG_SEND, AG_LSEND and AG_SSEND.

Call interface
Call interface in FBD representation
2.1 Program blocks for open communications services (SEND/RECEIVE interface)

Example in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call fc 5 (</td>
<td>//Block call</td>
</tr>
<tr>
<td>ACT := M 10.0,</td>
<td>//Job triggered by memory bit</td>
</tr>
<tr>
<td>ID := MW 12,</td>
<td>//Connection ID acc. to configuration</td>
</tr>
<tr>
<td>LADDR := W#16#0100,</td>
<td>//LADDR 256 dec. in HW Config</td>
</tr>
<tr>
<td>SEND := P#db99.dbx10.0 byte 240,</td>
<td>//Buffer with send data</td>
</tr>
<tr>
<td>LEN := MW 14,</td>
<td>//Length for send data</td>
</tr>
<tr>
<td>DONE := M 10.1,</td>
<td>//Execution code</td>
</tr>
<tr>
<td>ERROR := M 10.2,</td>
<td>//Error code</td>
</tr>
<tr>
<td>STATUS := MW 16 );</td>
<td>//Status code</td>
</tr>
</tbody>
</table>

Note
If you want to use FC5 AG_SSEND, you will need to select the "SPEED SEND/RECV" mode in the connection properties during configuration of the connection.

2.1.2.2 How AG_SEND / AG_LSEND / AG_SSEND work

How It works
The following diagrams illustrate the normal sequence of data transmission triggered in the user program using AG_SEND.

The way in which the FC functions depends on the CP type you are using.

- Case 1: Sequence with FC5, FC50, FC53 in S7-400 CPs
  In the S7–400, the transfer of the entire data area regardless of its length is handled by the CP after the first block call.

- Case 2: Sequence with FC5 in S7–300 CPs
  In the S7-300, the transfer takes place several data segments (each with 240 bytes of user data) and requires several FC calls to transfer the entire data.

Case 1: Sequence with FC5, FC50, FC53 in S7–400 CPs

The send job is executed as soon as the parameter ACT = 1 is passed.
Following this, the parameter ACT = 0 must be passed in at least one further call.

The status code in the output parameters DONE, ERROR and STATUS is updated in each block call and can be evaluated. To update the status code without starting a new send job, start a new block call with the parameter ACT = 0.
Case 2: Sequence with FC5 in S7–300 CPs

The send job is started as soon as the parameter ACT = 1 is passed.
In contrast to case 1, the protocol used here to transfer the data segments (each 240 bytes of user data) requires the FC to be called again for each segment.
Depending on the length of the user data, you must therefore continue to call the FC with ACT=0 until transfer of the entire data record is indicated; at least one further call is necessary. The data is transferred to the communication partner in segments of 240 bytes.

The status code in the output parameters DONE, ERROR and STATUS is updated in each block call and can be evaluated.
Note

In principle, it is possible to call the FC more than once within the CPU cycle to speed up the handling of the job. Do not forget, however, that this increases the load during the CPU cycle (the load differs depending on the CPU type)!

1) Parameter transfer DONE, ERROR, STATUS
### Explanation of the formal parameters - AG_SEND / AG_LSEND / AG_SSEND

The following table explains all the formal parameters for the AG_SEND / AG_LSEND / AG_SSEND functions:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>INPUT</td>
<td>BOOL</td>
<td>0, 1</td>
<td>If an FC is called with ACT=1, LEN bytes are sent from the ISO transport data area specified with the SEND parameter. If an FC is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.</td>
</tr>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>INT</td>
<td>1, 2...64 (S7-400)</td>
<td>The connection number of the connection is specified in the parameter ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1, 2...16 (S7-300)</td>
<td></td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>Module start address</td>
<td>When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.</td>
</tr>
<tr>
<td>SEND</td>
<td>INPUT</td>
<td>ANY</td>
<td>Specifies the address and length</td>
<td>The address of the data area points to one of the alternatives:</td>
</tr>
<tr>
<td>LEN</td>
<td>INPUT</td>
<td>INT</td>
<td>On ISO transport and ISOonTCP / TCP: 1, 2...8192 (or up to &quot;length specified for SEND parameter&quot;)</td>
<td>Number of bytes to be sent from the data area with this job. The possible values range from 1 to length specified for the SEND parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>On UDP: 1, 2...2048 (or up to &quot;length specified for SEND parameter&quot;)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note the block type:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- For S7-300</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The current versions of FC AG_SEND allow up to 8192 bytes (2048 bytes for UDP) to be transferred.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- For S7-400</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With FC AG_SEND, the data area is restricted to a maximum of 240 bytes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note the following with an S7-400:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Improved performance with shorter data records: Transfer of data records up to 240 bytes results in better performance! This applies regardless of the block type used (AG_SEND/AG_LSEND).</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- With AG_SSEND, the data area is restricted to a maximum of 1452 bytes.</td>
<td></td>
</tr>
</tbody>
</table>
### Program blocks for Industrial Ethernet

#### 2.1 Program blocks for open communications services (SEND/RECEIVE interface)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
</table>
| DONE      | OUTPUT      | BOOL      | 0: Job active  
1: Job done | The status parameter indicates whether or not the job was completed without errors.  
As long as DONE = 0, no further job can be triggered.  
DONE is set to 0 by the CP when it accepts a new job.  
For the meaning in conjunction with the ERROR and STATUS parameters, refer to AG_SEND, AG_LSEND and AG_SSEND status codes (Page 30) |
| ERROR     | OUTPUT      | BOOL      | 0: -  
1: Error | Error code  
For the meaning in conjunction with the DONE and STATUS parameters, refer to AG_SEND, AG_LSEND and AG_SSEND status codes (Page 30) |
| STATUS    | OUTPUT      | WORD      | Status code | Status code  
For the meaning in conjunction with the DONE and ERROR parameters, refer to AG_SEND, AG_LSEND and AG_SSEND status codes (Page 30) |

#### 2.1.2.4 AG_SEND, AG_LSEND and AG_SSEND status codes

##### Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>Job completed without errors.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0000H</td>
<td>No job being executed.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Job active.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>7000H</td>
<td>The condition code is possible only with S7-400: The FC was called with ACT=0; the job has not yet been processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>No configuration or the ISO/TCP service has not yet started on the Ethernet CP.</td>
</tr>
</tbody>
</table>
| 0    | 1     | 8184H  | - Illegal data type specified for the SEND parameter.  
- System error (the source data area is incorrect). |
| 0    | 1     | 8185H  | LEN parameter longer than SEND source area. |
| 0    | 1     | 8186H  | ID parameter invalid.  
- ID != 1, 2....16 (S7-300)  
- ID != 1, 2....64,(S7-400) |

Note

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.
# Program blocks for Industrial Ethernet

## 2.1 Program blocks for open communications services (SEND/RECEIVE interface)

### Table of Errors

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8302H</td>
<td>No receive resources on the destination station; the receiving station cannot process received data quickly enough or has not prepared any receive resources.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8304H</td>
<td>The connection is not established. The send job should only be attempted again after waiting for at least 100 ms.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8311H</td>
<td>The destination station cannot be obtained under the specified Ethernet address.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8312H</td>
<td>Ethernet error on the CP.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F22H</td>
<td>Source area invalid, e.g.: Area does not exist in the DB LEN parameter &lt; 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F24H</td>
<td>Area error when reading a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F28H</td>
<td>Alignment error reading a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F32H</td>
<td>Parameter contains a DB number that is too high.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F33H</td>
<td>DB number error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F3AH</td>
<td>Area not loaded (DB).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F42H</td>
<td>Acknowledgment timeout reading a parameter from the I/O area.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F44H</td>
<td>Access to a parameter to be read during block execution is prevented.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F7FH</td>
<td>Internal error, e.g. illegal ANY reference e.g. parameter LEN=0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8090H</td>
<td>• Module with this module start address does not exist; • The FC being used does not match the system family being used (remember to use different FCs for S7300 and S7400).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8091H</td>
<td>Module start address not at a doubleword boundary.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8092H</td>
<td>In the ANY reference, a type other than BYTE is specified. (S7-400 only)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80A4H</td>
<td>The communication bus connection between the CPU and CP is not established. (With newer CPU versions)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B0H</td>
<td>The module does not recognize the data record.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B1H</td>
<td>The specified length (in the LEN parameter) is incorrect.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B2H</td>
<td>The communication bus connection between the CPU and CP is not established.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C0H</td>
<td>The data record cannot be read.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C1H</td>
<td>The specified data record is currently being processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C2H</td>
<td>There are too many jobs pending.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C3H</td>
<td>CPU resources (memory) occupied.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C4H</td>
<td>Communication error (occurs temporarily and a repetition in the user program will often remedy the problem)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80D2H</td>
<td>Module start address incorrect.</td>
</tr>
</tbody>
</table>

### See also

/5/ (Page 282)
2.1.3 AG_RECV / AG_LRECV / AG_SRECV

2.1.3.1 Meaning and call - AG_RECV / AG_LRECV / AG_SRECV

Meaning of the block

The AG_RECV / AG_LRECV / AG_SRECV program block receives the data transferred on a configured connection from the Ethernet CP.

The data area specified for the receive data can be a memory bit area or a data block area.

Errorfree execution is indicated when the data could be received from the Ethernet CP.

Note:
Unless otherwise stated, all the following information applies to both the FCs AG_RECV and AG_LRECV / AG_SRECV.

Call

Call interface in FBD representation

Example in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>call fc 6 (</td>
<td>//Block call</td>
</tr>
<tr>
<td>ID := MW 40,</td>
<td>//Connection ID acc. to configuration</td>
</tr>
<tr>
<td>LADDR := W#16#0100,</td>
<td>//=LADDR 256 dec. in HW Config</td>
</tr>
<tr>
<td>RECV := P#M 0.0 BYTE 100,</td>
<td>//Buffer for received data</td>
</tr>
<tr>
<td>NDR := DB 110.DBX 0.6,</td>
<td>//Receive code</td>
</tr>
<tr>
<td>ERROR := DB 110.DBX 0.7,</td>
<td>//Error code</td>
</tr>
<tr>
<td>STATUS := DB 110.DBW 2,</td>
<td>//Status code</td>
</tr>
<tr>
<td>LEN := DB 110.DBW 4 );</td>
<td>//Received data length</td>
</tr>
</tbody>
</table>

Note

If you want to use FC63 AG_SRECV, you will need to select the "SPEED SEND/RECV" mode in the connection properties during configuration of the connection.
2.1.3.2 How AG_RECV / AG_LRECV / AG_SRECV work

How it works

The following diagrams illustrate the normal sequence of data acceptance triggered by an AG_RECV in the user program.

Each AG_RECV job in the user program is acknowledged by the Ethernet CP with an entry in the output parameters NDR, ERROR and STATUS.

The way in which the FC functions depends on the CP type you are using and the connection types.

- **Case 1: Sequence with FC6 in S7-300 CPs**
  
  With the newer CP types, optimized data transfer on the SEND/RECEIVE interface is available. In particular with longer data records, this allows a much higher data throughput on the interface between the CPU and CP.

- **Case 2: Sequence with FC6 and FC60 in S7-400 CPs**
  
  With FC6 / FC60 AG_RECV, the response on the S7-400 depends on the protocol used.
  - Case 2a: Sequence with ISO transport, ISO-on-TCP, UDP connections
    
    With these connection types, the transfer is handled by the CP with one or more FC6 / FC60 calls depending on the length of the data area.
  
  - Case 2b: Sequence with TCP connections
    
    On a TCP connection, the length specified in the ANY pointer of the RECV parameter is the deciding factor. An FC6 / FC60 job is completed with the condition code NDR=1, as soon as an amount of data corresponding to the specified length has been written to the receive buffer.

- **Case 3: Sequence with FC63 in S7-400 CPs**
  
  With FC63 AG_SRECV, the response on the S7-400 depends on the protocol used.
  - Case 3a: Sequence with ISO transport, ISO-on-TCP, UDP connections
    
    With these connection types, the transfer of the entire data area regardless of its length is started by the CP after the first block call.
  
  - Case 3b: Sequence with TCP connections
    
    On a TCP connection, the data on the CP is accepted up to the maximum specified job length with every call.

    The call must be repeated until a data record has been entered completely and consistently in the receive buffer. The reception of the completed data record is indicated in one of the later FC calls with the parameter NDR=1.

**Case 1: Sequence with FC6 in S7-300 CPs**

When FC6 is called, the user program prepares the buffer to receive data and instructs the CP to enter the received data there.

The protocol used here to transfer the data to the receive buffer requires the FC to be called again for each segment (240 bytes of user data).
Depending on the length of the user data, the FC must be called repeatedly until the complete transfer is indicated by parameter NDR=1.

The status code in the output parameters NDR, ERROR and STATUS is updated in each block call and can be evaluated.

**Case 2a: Sequence with FC6 and FC60 in S7-400 CPs (with ISO transport, ISO-on-TCP, UDP connections)**

When the FC is called, the user program prepares the buffer to receive data and instructs the CP to enter all available data there.

As soon as a data record has been entered fully and consistently in the receive buffer, this is indicated by the parameter NDR=1 in one of the next FC calls.

The status code in the output parameters NDR, ERROR and STATUS is updated in each block call and can be evaluated.
Case 2b: Sequence with FC6 / FC60 in S7-400 CPs (only for TCP connections)

On a TCP connection, the length specified in the ANY pointer of the RECV parameter is the deciding factor. An FC6 /FC60 job is completed with the condition code NDR=1, as soon as an amount of data corresponding to the specified length has been written to the receive buffer.

The sequence example shows a situation in which the length in the ANY pointer was set to 400 bytes for an FC60.
Case 3a: Sequence with FC63 in S7-400 CPs (with ISO transport, ISO-on-TCP, UDP connections)

When the FC is called, the user program prepares the buffer for the received data and instructs the CP to enter all data until the end of the transfer there; in other words, when the length specified in the ANY pointer is reached.

As soon as the data record has been entered fully and consistently in the receive buffer, this is indicated by the parameter NDR=1 in one of the next FC calls.

The maximum length for received data is 1452 bytes. The size of the receive buffer must always be set to this value.

The status code in the output parameters NDR, ERROR and STATUS is updated in each block call and can be evaluated.
Case 3b: Sequence with FC63 in S7-400 CPs (only for TCP connections)

When the FC is called, the user program prepares the buffer to receive data and instructs the CP to enter the currently available data there. Until the data has been completely entered in the receive buffer, the message "Job active" (8181H) is sent.

With each new FC call, the currently available data is entered in the receive buffer. When the data record has been entered fully and consistently in the receive buffer, this is indicated by the parameter NDR=1 in one of the next FC calls.

The maximum length for received data is 1452 bytes. The size of the receive buffer must always be set to this value.

The status code in the output parameters NDR, ERROR and STATUS is updated in each block call and can be evaluated.
2.1.3.3 Explanation of the formal parameters - AG_RECV / AG_LRECV / AG_SRECV

The following table explains all the formal parameters for the AG_RECV / AG_LRECV / AG_SRECV function:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>INT</td>
<td>1, 2...64 (S7-400) 1, 2...16 (S7-300)</td>
<td>The connection number of the ISO transport connection is specified in the ID parameter.</td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>Module start address</td>
<td>When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.</td>
</tr>
</tbody>
</table>
| RECV      | INPUT       | ANY       | Specifies the address and length | The address of the data area points to one of the alternatives:  
- Memory bit area  
- Data block area  
Note on length:  
Performance is improved when transferring data records up to 212 bytes if you also restrict the length to 212 bytes at the RECV parameter.  
Note on FC63 AG_SRECV:  
With FC63 AG_SRECV, always set RECV to the maximum receive buffer length of 1452 bytes. Otherwise, the following error can occur in certain situations:  
NDR=0; ERROR=1; STATUS=8185
|
### 2.1 Program blocks for open communications services (SEND/RECEIVE interface)

#### 2.1.3.4 AG_RECV, AG_LRECV and AG_SRECV condition codes

**Condition codes**

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

**Note**

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>New data accepted.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8180H</td>
<td>There is no data available yet (not with AG_SRECV).</td>
</tr>
</tbody>
</table>
### 2.1 Program blocks for open communications services (SEND/RECEIVE interface)

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Job active.</td>
</tr>
</tbody>
</table>
| 0   | 1     | 8183H  | The configuration is missing;  
• The ISO transport service has not yet started on the Ethernet CP;  
• The connection is not established. |
| 0   | 1     | 8184H  | Illegal type specified for the RECV parameter;  
• System error. |
| 0   | 1     | 8185H  | Destination buffer (RECV) is too short. |
| 0   | 1     | 8186H  | ID parameter invalid.  
ID ! = 1, 2,...16 (S7-300).  
ID ! = 1, 2,...64.(S7-400) |
| 0   | 1     | 8304H  | The connection is not established. The receive job should only be attempted again after waiting for at least 100 ms. |
| 0   | 1     | 8F23H  | Source area invalid, e.g.:  
Area does not exist in the DB. |
| 0   | 1     | 8F25H  | Range error when writing a parameter. |
| 0   | 1     | 8F29H  | Alignment error writing a parameter |
| 0   | 1     | 8F30H  | Parameter is in the writeprotected first current data block. |
| 0   | 1     | 8F31H  | Parameter is in the writeprotected second current data block. |
| 0   | 1     | 8F32H  | Parameter contains a DB number that is too high. |
| 0   | 1     | 8F33H  | DB number error. |
| 0   | 1     | 8F3AH  | Destination area not loaded (DB). |
| 0   | 1     | 8F43H  | Timeout writing a parameter to the I/O area. |
| 0   | 1     | 8F45H  | Address of the parameter to be written is disabled in the access track. |
| 0   | 1     | 8F7FH  | Internal error, e.g. illegal ANY reference. |
| 0   | 1     | 8090H  | No module with this module start address exists or the CPU is in STOP mode;  
• The FC being used does not match the system family being used (remember to use different FCs for S7300 and S7400). |
| 0   | 1     | 8091H  | Module start address not at a doubleword boundary. |
| 0   | 1     | 8092H  | In the ANY reference, a type other than BYTE is specified.  
(S7-400 only) |
| 0   | 1     | 80A0H  | Negative acknowledgment reading from the module. |
| 0   | 1     | 80A4H  | The communication bus connection between the CPU and CP is not established. |
| 0   | 1     | 80B0H  | The module does not recognize the data record. |
| 0   | 1     | 80B1H  | Possible causes:  
• The destination area is invalid.  
• The destination area is too short.  
• The destination area for the received data was adequately dimensioned.  
Remedy: Run another receive call with maximum receive buffer size. This applies regardless of the connection type (unicast / multicast / broadcast) and the device family (S7-300 / S7-400). |
| 0   | 1     | 80B2H  | The communication bus connection between the CPU and CP is not established. |
### 2.2 Program blocks for access coordination with FETCH/WRITE

#### 2.2.1 Overview of program blocks and their use

**Overview**

The following program blocks are available for the FETCH/WRITE function to coordinate access:

<table>
<thead>
<tr>
<th>Program block</th>
<th>can be used with:</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S7-300</td>
<td>S7-400</td>
</tr>
<tr>
<td>AG_LOCK (FC7)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>AG_UNLOCK (FC8)</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

**Caution when Configuring**

If you use program blocks AG_LOCK and AG_UNLOCK, specify the following information for CPs in S7400 stations in the configuration:

- Under "Properties > Addresses"
  The "Address setting for LOCK/UNLOCK" option must be selected if the selection is available.

---

### NDR, ERROR, STATUS

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>80C0H</td>
<td>The data record cannot be read.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C1H</td>
<td>The specified data record is currently being processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C2H</td>
<td>There are too many jobs pending.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C3H</td>
<td>CPU resources (memory) occupied.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C4H</td>
<td>Communications error occurs temporarily and a repetition in the user program will often remedy the problem.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80D2H</td>
<td>Module start address incorrect.</td>
</tr>
</tbody>
</table>

**See also**

/5/ (Page 282)
How It works

With these program blocks, you can coordinate access to system memory areas so that no inconsistent data is created and transferred. The control is from the user program in the S7 CPU that can, if necessary, disable an external FETCH/WRITE access using an AG_LOCK call. After a certain time or after the local write/read access is completed, an AG_UNLOCK job can be used to enable external access again.

Another advantage is that this access lock only applies to the FETCH/WRITE connection specified in the call. If more than one FETCH/WRITE connection is configured, these can, for example, be used for certain specific system areas and a selective access coordination can be implemented.

The following diagram illustrates the usual chronological sequence of memory access coordination controlled in the user program with AG_LOCK and AG_UNLOCK.

The lock job must first be monitored in the user program using the code in the return parameter LOCKED. As long as LOCKED=0 is indicated, it must be assumed that there is still an external FETCH/WRITE access active.
If LOCKED=1 is indicated, this shows that the lock is active; data can now be modified by the user program.

The status code is updated at each block call.

### 2.2.2 AG_LOCK

#### 2.2.2.1 Meaning and call - AG_LOCK

**Meaning of the block**

Using the AG_LOCK block the data exchange using FETCH or WRITE on the connection selected with the parameter ID is disabled. The LOCKED output indicates whether or not the lock was successful. If the lock was not successful, the job must be triggered again in a later CPU cycle.

The STATUS output indicates the status of the CP for this connection.

**Call**

- **Call interface in FBD representation**

```
AG_LOCK
```

- **Example in STL representation**

```
call fc 7 {                                  //Block call
ID := DB 100.DBW 2,                         //Connection ID acc. to configuration
LADDR := W#16#0100,                        //LADDR 256 dec. in HW Config
LOCKED := DB 100.DBX 0.6,                  //Status code of access lock
STATUS := DB 100.DBW 4 );                 //Status code
```
2.2.2.2 **Explanation of the formal parameters - AG_LOCK**

**Explanation of the formal parameters**

The following table explains all the formal parameters for the AG_LOCK function:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>INT</td>
<td>1,2...16 for S7300 1,2...64 for S7400</td>
<td>The connection number of the connection is specified in the parameter ID.</td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.</td>
</tr>
<tr>
<td>LOCKED</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: not (yet) locked 1: locked</td>
<td>Shows the status of the access lock requested on the specified FETCH/WRITE connection.</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td></td>
<td>Status code For the meaning, refer to Condition codes of AG_LOCK (Page 44)</td>
</tr>
</tbody>
</table>

2.2.2.3 **Condition codes of AG_LOCK**

**Condition codes**

The following table shows the STATUS code that must be evaluated by the user program.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>7000H</td>
<td>CP is not processing a job</td>
</tr>
<tr>
<td>7001H</td>
<td>FETCH active</td>
</tr>
<tr>
<td>7002H</td>
<td>WRITE active</td>
</tr>
<tr>
<td>8183H</td>
<td>FETCH/WRITE not configured for this connection (S7-400 only)</td>
</tr>
<tr>
<td>8186H</td>
<td>ID number not in permitted range (e.g. 1...64 for S7-400 Industrial Ethernet CPs)</td>
</tr>
<tr>
<td>80A4H</td>
<td>The communication bus connection between the CPU and CP is not established (with newer CPU versions)</td>
</tr>
<tr>
<td>80B9H</td>
<td>The module does not recognize the data record.</td>
</tr>
<tr>
<td>80B1H</td>
<td>The specified length (in the LEN parameter) is incorrect.</td>
</tr>
<tr>
<td>80B2H</td>
<td>The communication bus connection between the CPU and CP is not established.</td>
</tr>
<tr>
<td>80C0H</td>
<td>The data record cannot be read.</td>
</tr>
<tr>
<td>80C1H</td>
<td>The specified data record is currently being processed.</td>
</tr>
<tr>
<td>80C2H</td>
<td>There are too many jobs pending.</td>
</tr>
<tr>
<td>80C3H</td>
<td>CPU resources (memory) occupied.</td>
</tr>
<tr>
<td>80C4H</td>
<td>Communications error occurs temporarily and a repetition in the user program will often remedy the problem.</td>
</tr>
<tr>
<td>80D2H</td>
<td>Module start address incorrect.</td>
</tr>
</tbody>
</table>
2.2.3 AG_UNLOCK

2.2.3.1 Meaning and call - AG_UNLOCK

**Meaning of the block**

With the aid of the AG_UNLOCK block, you enable external access to user memory areas of the S7-CPU. With FETCH or WRITE, access via the connection selected with the ID parameter is then possible.

The AG_UNLOCK follows an access lock with AG_LOCK.

**Call**

Call interface in FBD representation

```
AG_UNLOCK

ID           STATUS
LADDR
```

**Example in STL representation**

```stl
STL | Explanation
--- |------------
call fc 8 { //Block call
  ID := DB 100.DBW 2, //Connection ID acc. to configuration
  LADDR := W#16#0100, //LADDR 256 dec. in HW Config
  STATUS := DB 100.DBW 4 ); //Status code
```

**How it works**

To release the connection again, the FC must clear the LOCK request bit again. The FC also shows the current status using error messages.
2.2.3.2 Explanation of the formal parameters - AG_UNLOCK

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_UNLOCK function:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>INT</td>
<td>1,2...16 for S7300, 1,2...64 for S7400</td>
<td>The connection number of the connection is specified in the parameter ID. (See Configuration)</td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>Module start address</td>
<td>Module start address When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Status code: For the meaning, refer to Condition codes of AG_UNLOCK (Page 46)</td>
<td></td>
</tr>
</tbody>
</table>

2.2.3.3 Condition codes of AG_UNLOCK

Condition codes

The following table shows the STATUS code that must be evaluated by the user program.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>7000H</td>
<td>CP is not processing a job</td>
</tr>
<tr>
<td>7001H</td>
<td>FETCH active</td>
</tr>
<tr>
<td>7002H</td>
<td>WRITE active</td>
</tr>
<tr>
<td>8183H</td>
<td>FETCH/WRITE not configured for this connection (S7-400 only)</td>
</tr>
<tr>
<td>8186H</td>
<td>ID number not in permitted range (e.g. 1...64 for S7-400 Industrial Ethernet CPs)</td>
</tr>
<tr>
<td>80A4H</td>
<td>The communication bus connection between the CPU and CP is not established (with newer CPU versions)</td>
</tr>
<tr>
<td>80B0H</td>
<td>The module does not recognize the data record.</td>
</tr>
<tr>
<td>80B1H</td>
<td>The specified length (in the LEN parameter) is incorrect.</td>
</tr>
<tr>
<td>80B2H</td>
<td>The communication bus connection between the CPU and CP is not established.</td>
</tr>
<tr>
<td>80C0H</td>
<td>The data record cannot be read.</td>
</tr>
<tr>
<td>80C1H</td>
<td>The specified data record is currently being processed.</td>
</tr>
<tr>
<td>80C2H</td>
<td>There are too many jobs pending.</td>
</tr>
<tr>
<td>80C3H</td>
<td>CPU resources (memory) occupied.</td>
</tr>
<tr>
<td>80C4H</td>
<td>Communications error</td>
</tr>
<tr>
<td></td>
<td>occurs temporarily and a repetition in the user program will often remedy the problem</td>
</tr>
<tr>
<td>80D2H</td>
<td>Module start address incorrect.</td>
</tr>
</tbody>
</table>
2.3 Program blocks for connection and system diagnostics

2.3.1 AG_CNTRL

2.3.1.1 Meaning and call - AG_CNTRL

How It works

With the AG_CNTRL program block, you can diagnose connections. When necessary, you can initialize connection establishment again using AG_CNTRL.

Note

AG_CNTRL and AG_CNTEX

The AG_CNTEX program block provides expanded functionality compared with the AG_CNTRL program block.
All the functions of AG_CNTRL are included in AG_CNTEX and they can be used identically in the user program.

The following actions are possible by setting commands:

- Reading out connection information
  
  Based on status information, you can decide whether or not it would be useful to reset all or individual connections of the CP.
- Resetting configured connections
  
  You can reset individual connections or all connections of a CP.
- Aborting the active connection and establishing it again

The commands of the AG_CNTRL function (FC) are permitted only for SEND/RECV connections based on the ISO / RFC / TCP / UDP protocols.

Call interface

Call interface in FBD representation

```
AG_CNTRL

BOOL
INT
WORD
INT

ACT
ID
LADDR
CMD

DONE
ERROR
STATUS
RESULT 1

BOOL
BOOL
WORD
DWORD

RESULT 2
```
Example in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call fc 10 (</td>
<td>//AG_CNTRL block call</td>
</tr>
<tr>
<td>ACT := M1.0,</td>
<td>//Job trigger</td>
</tr>
<tr>
<td>ID := MW8,</td>
<td>//Connection ID acc. to configuration</td>
</tr>
<tr>
<td>LADDR := W#16#100,</td>
<td>//Module address acc. to HW Config</td>
</tr>
<tr>
<td>CMD := MW6,</td>
<td>//Command ID</td>
</tr>
<tr>
<td>DONE := M20.1,</td>
<td>//Execution code</td>
</tr>
<tr>
<td>ERROR := M20.2,</td>
<td>//Error code</td>
</tr>
<tr>
<td>STATUS := MW22,</td>
<td>//Status code</td>
</tr>
<tr>
<td>RESULT1 := MD24,</td>
<td>//Job result 1</td>
</tr>
<tr>
<td>RESULT2 := MD28 );</td>
<td>//Job result 2</td>
</tr>
</tbody>
</table>

See also

FAQ under entry ID 33414377
2.3.1.2 How AG_CNTRL works

Operating principle

The following diagram shows a typical sequence of AG_CNTRL jobs in the user program.

The diagram shows how the connection status is initially queried and then, in a second job, how the connection termination is triggered with the reset command.

The reset ID (bit 15 in RESULT1) is set on the CP. If there is a status query later, it is therefore clearly recognizable whether the connection has been reset due to a reset job. The reset ID is cleared on the CP only after this status query (or as a result of an explicit CN_CLEAR_RESET command).
2.3 Program blocks for connection and system diagnostics

Note

The block must be called with ACT = 1; if it is called with ACT=0, there is no function call and the block is exited again immediately.

Since the job result of FC10 is obtained synchronous to the call, it can be called again in the same cycle.

2.3.1.3 Explanation of the formal parameters - AG_CNTRL

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_CNTRL function:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>INPUT</td>
<td>BOOL</td>
<td>0, 1</td>
<td>The FC must be called with ACT=1. If it is called with ACT=0, there is no function call and the block is exited again immediately.</td>
</tr>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>INT</td>
<td>• 1, 2, .., n, or • 0</td>
<td>The connection number of the connection is specified in the parameter ID. The connection number can be found in the configuration. n is the maximum number of connections and is dependent on the product (S7-300 or S7-400). If the call addresses all connections (_ALL function with CMD 3 or 4), 0 must be specified as the ID.</td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.</td>
</tr>
<tr>
<td>CMD</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>Command to FC AG_CNTRL.</td>
</tr>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: Job still being processed or not yet triggered 1: Job done</td>
<td>This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to AG_CNTRL codes (Page 51) Note: If DONE=1, RESULT can be evaluated</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: No error 1: Error situation</td>
<td>Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to AG_CNTRL codes (Page 51)</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td></td>
<td>Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to AG_CNTRL codes (Page 51)</td>
</tr>
</tbody>
</table>
### Parameter Declaration Data type Possible values Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>OUTPUT</td>
<td>DWORD</td>
<td></td>
<td>Information returned according to the command sent to FC AG_CNTRL.</td>
</tr>
<tr>
<td>RESULT2</td>
<td>OUTPUT</td>
<td>DWORD</td>
<td></td>
<td>Only to be evaluated for S7-400: Part 2 of information returned according to the command sent to FC AG_CNTRL.</td>
</tr>
</tbody>
</table>

#### 2.3.1.4 AG_CNTRL codes

**Condition codes**

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

The command results in the RESULT1/2 parameters must also be evaluated according to "Commands and job results - AG_CNTRL (Page 52) ".

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>A job (CMD) was transferred successfully to the CP (for example RESET) or a status was read successfully from the CP. The RESULT1/2 parameters can be evaluated.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0000H</td>
<td>There has been no block call yet or the block was called with ACT=0.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Job active The block call must be repeated with the same parameters until DONE or ERROR is signaled.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>No configuration or the service has not yet started on the Ethernet CP.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8186H</td>
<td>The ID parameter is invalid. The permitted ID depends on the selected command.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8187H</td>
<td>The CMD parameter is invalid.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8188H</td>
<td>Sequence error in the ACT control (Note: this code does not occur in the product version of the CP / firmware).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8189H</td>
<td>The CP version / firmware used does not support FC10. The code is set when you call a CP 3431-EX20 with firmware as of V1.3.9; with other CP types, the code 80B0H is set instead. Note: FC10 in version V1.0 is supported by the CPs as of CP 343-1EX21/GX21; this code does not occur with these modules.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8090H</td>
<td>No module with this module start address exists. or The FC being used does not match the system family being used (remember to use different FCs for S7300 and S7400). or The function is not supported by this module.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8091H</td>
<td>The module start address is not at a doubleword boundary.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B0H</td>
<td>The module does not recognize the data record.</td>
</tr>
</tbody>
</table>
### 2.3 Program blocks for connection and system diagnostics

#### 2.3.1.5 Commands and job results - AG_CNTRL

#### Commands and evaluating the job results

The following table shows you the possible commands and the results that can be evaluated in the RESULT1/2 parameters.

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NOP – no operation</td>
</tr>
</tbody>
</table>

The block executes without a job being sent to the CP.

#### Table 2-5 Commands to FC AG_CNTRL

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NOP – no operation</td>
</tr>
</tbody>
</table>

The block executes without a job being sent to the CP.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001H</td>
<td>Executed without error</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000H</td>
<td>Default</td>
</tr>
</tbody>
</table>

#### Note

**Command evaluation with older CP types or firmware versions**

The commands described below are supported by the current CP types or firmware versions. You should also check the more detailed information under the following entry ID:

### CMD

| CMD | Meaning
|-----|-------------|
| 1   | **CN_STATUS** – connection status  
This command returns the status of the connection selected with the ID.  
The CP is selected in the LADDR parameter.  
If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5). |

#### RESULT (for CMD = 1)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Bit</th>
<th>Value / meaning</th>
</tr>
</thead>
</table>
| RESULT1   | 0000 000*H     | Bits 0-3: Codes for the send direction  
(excluded values: 0x2) |
|           |                | Bit 0 | Connection type |
|           |                | • 0: no send and receive connection  
• 1: Connection reserved for send and receive jobs |
|           |                | Bit 1 | Status of current job |
|           |                | • 0: No send job being executed  
• 1: Send job being executed |
|           |                | Bits 2+3 | Previous job: |
|           |                | • 00: No information available on previous send job  
• 01: previous send job completed successfully  
• 10: previous send job not completed successfully |

---

**CMD=1 continued**

This command returns the status of the connection selected with the ID.  
The CP is selected in the LADDR parameter.  
If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).
### CN_STATUS – connection status (CMD=1 continued)

This command returns the status of the connection selected with the ID.

The CP is selected in the LADDR parameter.

If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CN_STATUS – connection status</td>
</tr>
</tbody>
</table>

#### RESULT (for CMD = 1)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0*00H</td>
<td></td>
</tr>
</tbody>
</table>

- Bits 8-11: Codes for FETCH/WRITE (excluded values: 0x3, 0x7, 0x8, 0xB, 0xF)
- **Bit 8** Connection type:
  - 0: No FETCH connection
  - 1: Connection reserved for FETCH jobs
- **Bit 9** Connection type:
  - 0: No WRITE connection
  - 1: Connection reserved for WRITE jobs
- **Bit 10** Job status (FETCH/WRITE):
  - 0: Job status OK
  - 1: Job status NOT OK
  - This ID is set in the following situations:
    - The job was acknowledged negatively by the CPU
    - The job could not be forwarded to the CPU because the connection was in the "LOCKED" status.
    - The job was rejected because the FETCH/WRITE header did not have the correct structure.
- **Bit 11** Status of FETCH/WRITE job
  - 0: No job active
  - 1: Job from LAN active
### CN_STATUS – connection status (CMD=1 continued)

This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter.

If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).

#### RESULT (for CMD = 1)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Bit</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 *000H</td>
<td>Bits 12-15: General CP information (excluded values: 0x3, 0xB)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 12 + 13: Information on connection status: (only available for SEND/RECV connections based on the ISO/RFC/TCP protocols, with UDP, the corresponding internal information is output)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 00: Connection is terminated</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 01: Connection establishment active</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 10: Connection termination active</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 11: Connection is established</td>
<td></td>
</tr>
<tr>
<td>RESULT1</td>
<td>**** 0000H</td>
<td>Bit 14: CP information:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: CP in STOP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: CP in RUN</td>
<td></td>
</tr>
<tr>
<td>RESULT1</td>
<td>0000 0000H</td>
<td>Bit 15: Reset ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: FC10 has not yet reset a connection or the reset ID was cleared.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: The control block has executed a connection reset.</td>
<td></td>
</tr>
</tbody>
</table>

| RESULT2   | 0000 0000H     | Bits 16-31: Reserved |
|           |                | 0 – reserved for later expansions |

### CN_RESET – connection reset

This command resets the connection selected with ID.

The CP is selected in the LADDR parameter.

Resetting the connection means that a connection is aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted.

An entry is also generated in the diagnostics buffer in which the job result can be found.

#### RESULT (for CMD = 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001H</td>
<td>The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment were triggered.</td>
</tr>
</tbody>
</table>
### 2.3 Program blocks for connection and system diagnostics

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
<th>Hex value</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0002H</td>
<td>The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).</td>
<td>RESULT2</td>
<td>0000 0000H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
<th>Hex value</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>CN_STATUS_ALL – all connections status</td>
<td>RESULT1</td>
<td>**** ****H</td>
</tr>
<tr>
<td></td>
<td>This command returns the connection status of all connections (established/terminated) in the RESULT1/2 parameters (at total of 8 bytes of group information).</td>
<td>RESULT2</td>
<td>**** ****H</td>
</tr>
<tr>
<td></td>
<td>The ID parameter must be set to &quot;0&quot; (checked for 0).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The CP is selected in the LADDR parameter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>When necessary, you can obtain detailed information about a terminated or unconfigured connection using a further connection status call with CMD=1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RESULT (for CMD = 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parameter</td>
<td>Hex value/range</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RESULT1</td>
<td>**** ****H</td>
<td>32 bits: Connection 1 - 32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 – connection terminated / not configured</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 – connection established</td>
</tr>
<tr>
<td></td>
<td>RESULT2</td>
<td>**** ****H</td>
<td>32 bits: Connection 33 - 64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0 – connection terminated / not configured</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1 – connection established</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
<th>Hex value</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>CN_RESET_ALL – all connections reset:</td>
<td>RESULT1</td>
<td>0000 0001H</td>
</tr>
<tr>
<td></td>
<td>This command resets all connections.</td>
<td>RESULT1</td>
<td>0000 0002H</td>
</tr>
<tr>
<td></td>
<td>The ID parameter must be set to &quot;0&quot; (checked for 0).</td>
<td>RESULT2</td>
<td>0000 0000H</td>
</tr>
<tr>
<td></td>
<td>The CP is selected in the LADDR parameter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Resetting the connections means that connections are aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>An entry is also generated in the diagnostics buffer in which the job result can be found.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RESULT (for CMD = 4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parameter</td>
<td>Hex value/range</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RESULT1</td>
<td>0000 0001H</td>
<td>The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment of all connections were triggered.</td>
</tr>
<tr>
<td></td>
<td>RESULT1</td>
<td>0000 0002H</td>
<td>The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).</td>
</tr>
<tr>
<td></td>
<td>RESULT2</td>
<td>0000 0000H</td>
<td>Default</td>
</tr>
</tbody>
</table>
### CMD 5  
**CN_CLEAR_RESET** – Clear the reset ID

This command resets the reset ID (bit 15 in RESULT1) for the connection selected with ID. The CP is selected in the LADDR parameter.

This job executes automatically when the connection status is read (CMD=1); the separate job described here is therefore only required in special situations.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001H</td>
<td>The clear job was transferred to the CP successfully.</td>
</tr>
<tr>
<td>RESULT1</td>
<td>0000 0002H</td>
<td>The Clear job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000H</td>
<td>Default</td>
</tr>
</tbody>
</table>

### CMD 6  
**CN_DISCON** – connection disconnect

This command resets the connection selected with ID and LADDR. Resetting the connection is achieved by aborting the connection. Any data in the stack is lost without any message being displayed. The connection is not established again automatically afterwards. The connection can be established again with the CN_STARTCON control job. A diagnostics buffer entry is created in which you will find the job result.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001H</td>
<td>The job was transferred to the CP successfully. The connection abort was initiated.</td>
</tr>
<tr>
<td>RESULT1</td>
<td>0000 0002H</td>
<td>The job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000H</td>
<td>Default</td>
</tr>
</tbody>
</table>

### CMD 7  
**CN_STARTCON** – start connection

This command establishes a connection selected with ID and LADDR and aborted earlier with the control job CN_DISCON. A diagnostics buffer entry is created in which you will find the job result.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001H</td>
<td>The connection establishment job was transferred to the CP successfully. The connection establishment was initiated.</td>
</tr>
<tr>
<td>RESULT1</td>
<td>0000 0002H</td>
<td>The connection establishment job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000H</td>
<td>Default</td>
</tr>
</tbody>
</table>
See also

You will find further information in the FAQ under the following entry ID: 33414377 (https://support.industry.siemens.com/cs/ww/en/view/33414377)

2.3.2 AG_CNTEX

2.3.2.1 Meaning and call - AG_CNTEX

Significance and how it works

With the AG_CNTEX program block, it is possible to run diagnostics on connections and to address devices using the ping command via the network. When necessary, you can initialize connection establishment again using AG_CNTEX.

Note

AG_CNTRL and AG_CNTEX

The AG_CNTEX program block provides expanded functionality compared with the AG_CNTRL program block.

All the functions of AG_CNTRL are included in AG_CNTEX and they can be used identically in the user program.

The following actions are possible by setting commands:

- Reading out connection information
  - Based on status information, you can decide whether or not it would be useful to reset all or individual connections of the CP.
- Resetting configured connections
  - You can reset individual connections or all connections of a CP.
- Aborting the active connection and establishing it again
- Reading out connection types configured on the CP (expanded functionality compared with AG_CNTRL)
- Send PING command (expansion compared with AG_CNTRL)
  - You can check whether a specific node is reachable in the network.

The commands of the AG_CNTEX program block are permitted only for SEND/RECV connections based on the ISO / RFC / TCP / UDP protocols.
Note

Availability in the block library

If the program block AG_CNTEX is not yet available in the SIMATIC_NET_CP block library, install the current SIMATIC NET block library. You will find this under the following entry ID on the Internet:


Call interface

Call interface in FBD representation

```
AG_CNTEX

BOOL ACT DONE BOOL
INT ID ERROR BOOL
WORD LADDR STATUS WORD
INT CMD RESULT1 DWORD
ANY PING RESULT2 DWORD
```

Example in STL representation

```
call fb 10, DB10 ( //AG_CNTEX block call
ACT := DB11.DBX0.0, //Job trigger
ID := DB11.DBW2, //Connection ID acc. to configuration
LADDR := DB11.DBW4, //module address acc. to hardware configuration
CMD := DB11.DBW6, //Command ID
PING := P#DB11.DBX8.0, // Data area with ping data
DONE := DB11.DBX16.0, //Execution code
ERROR := DB11.DBX16.1, //Error code
STATUS := DB11.DBW18, //Status code
RESULT1 := DB11.DBW20, //Job result 1
RESULT2 := DB11.DBW24); //Job result 2
```

2.3.2.2 How AG_CNTEX works

Operating principle

The following diagram shows a typical sequence of AG_CNTEX jobs in the user program. Below you will find examples of the sequence of ping commands.

The diagram shows how the connection status is initially queried and then, in a second job, how the connection termination is triggered with the reset command.

The reset ID (bit 15 in RESULT1) is set on the CP. If there is a status query later, it is therefore clearly recognizable whether the connection has been reset due to a reset job. The

1) Parameter transfer DONE, ERROR, STATUS and RESULT1/2
reset ID is cleared on the CP only after this status query or as a result of an explicit CN_CLEAR_RESET command

**Note**

When the block is called, ACT = 1 must be set. If it is called with ACT=0, there is no function call and the block is exited again immediately.

Since the job result of AG_CNTEX is obtained synchronous to the call, AG_CNTEX can be called again in the same cycle.

### 2.3.2.3 How the ping function works

**Mode of operation / call sequence**

Below there is a description of how to use the commands CMD=8 and CMD=9 to send a ping command.

With the CMD=8 ping command, you instruct the CP to send 4 successive ping requests over the network to the IP address specified in the job. The ping echo is expected by the CP within the period of time you set in the ping job field.

The CP registers the response times and enters these in the RESULT 1/2 parameters.

The RESULT 1/2 parameters can be queried with ping command CMD=9. As soon as the 4 ping requests have been replied to, or their set monitoring time has been exceeded, execution is confirmed in the DONE=1 parameter. The ping result can then be queried within a maximum of 30 seconds; afterwards, the RESULT entries become invalid.

**Note**

**Ping is only possible over a configured connection**

The ping command is only possible if at least one connection for the SEND/RECEIVE interface (TCP, ISO-on-TCP, ISO-Transport, UDP) is configured.

The following schematic shows a typical sequence of a ping request with the corresponding ping result request.
Several ping requests at the same time

You can send up to 4 ping requests at the same time to different IP addresses. To do this, you must use the same instance DB for the ping requests. Further ping requests are possible only after completion of at least one of the current PING requests.

If too many ping requests are sent at the same time, an error message to this effect is output (STATUS parameter = 828AH).
When are ping requests completed?

Ping requests count as being completed as soon as one of the following conditions is met:
- The ping result was read out:
- The ping result was not read out but 30 seconds have elapsed since the ping result was available.

Replies of the SIMATIC NET CPs to ICMP frames

To reduce the activity of the CPs to ping requests of AG_CNTEX, SIMATIC NET CPs for S7-300/S7-400 reply to ICMP frames a maximum of ten times within 100 milliseconds.

2.3.2.4 Explanation of the formal parameters - AG_CNTEX

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_CNTEX function:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>INPUT</td>
<td>BOOL</td>
<td>0, 1</td>
<td>The FB must be called with ACT=1. If it is called with ACT=0, there is no function call and the block is exited again immediately.</td>
</tr>
</tbody>
</table>
| ID        | INPUT       | INT       | • 1, 2, .., n, or • 0 | The connection number of the connection is specified in the parameter ID. The connection number can be found in the configuration. n is the maximum number of connections and is dependent on the product (S7-300 or S7-400). For a call that addresses all connections, 0 must be specified as the ID. This affects:
- the functions CN_STATUS_ALL (CMD3) and CN_RESET_ALL (CMD4)
- Ping command with CMD 8 or CMD 9 |
| LADDR     | INPUT       | WORD      | Module start address When you configure the CP with STEP 7, the module start address is displayed. Specify this address here. |
| CMD       | INPUT       | INT       | Command to FB AG_CNTEX |
| PING      | INPUT       | ANY       | References a block of data (for example DB) that contains the data structure for the ping command. The block of data contains the IP address and optional information about the time monitoring and the number of bytes to be transferred in the ping request. Data structure, see below |
2.3 Program blocks for connection and system diagnostics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: Job still being processed or not yet triggered 1: Job done</td>
<td>This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to AG_CNTEX codes (Page 64) Note: If DONE=1, RESULT can be evaluated</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: No error 1: Error situation</td>
<td>Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to AG_CNTEX codes (Page 64)</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Status code</td>
<td>Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to AG_CNTEX codes (Page 64)</td>
</tr>
<tr>
<td>RESULT1</td>
<td>OUTPUT</td>
<td>DWORD</td>
<td>Information returned to AG_CNTEX according to the command.</td>
<td></td>
</tr>
<tr>
<td>RESULT2</td>
<td>OUTPUT</td>
<td>DWORD</td>
<td>Part 2 of the information returned to AG_CNTEX according to the command.</td>
<td></td>
</tr>
</tbody>
</table>

PING block of data

The PING parameter references a block of data with the following data structure:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Range of values</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address</td>
<td>ARRAY [1..4] of Byte</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>INT</td>
<td>1..60000 ms</td>
<td>Can be specified as an option; default value = 1000 ms</td>
</tr>
<tr>
<td>Size</td>
<td>INT</td>
<td>1..1000 bytes</td>
<td>Can be specified as an option; default value = 32 bytes</td>
</tr>
</tbody>
</table>

2.3.2.5 AG_CNTEX codes

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.
The command results in the RESULT1/2 parameters must also be evaluated according to Commands and job results - AG_CNTEX (Page 66).

Table 2-6 AG_CNTEX codes

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>A job (CMD) was transferred successfully to the CP (for example RESET) or a status was read successfully from the CP. The RESULT1/2 parameters can be evaluated.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0000H</td>
<td>There has been no block call yet or the program block was called with ACT=0.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Job active. The block call must be repeated with the same parameters until DONE or ERROR is signaled.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>No configuration or the service has not yet started on the Ethernet CP.</td>
</tr>
</tbody>
</table>
| 0    | 1     | 8184H  | System error or wrong parameter type. The cause can be:  
- Data type of the ANY pointer not correct for the PING parameter.  
- The ANY pointer references an odd bit address. |
| 0    | 1     | 8186H  | The ID parameter is invalid. The permitted ID depends on the selected command. |
| 0    | 1     | 8187H  | The CMD parameter is invalid. |
| 0    | 1     | 8090H  | Possible meanings:  
- No module with this module start address exists;  
- The program block being used does not match the system family being used (remember to use different program blocks for S7300 and S7400);  
- The function is not supported by this module. |
| 0    | 1     | 8091H  | The module start address is not at a doubleword boundary. |
| 0    | 1     | 8092H  | The module start address is incorrect. |
| 0    | 1     | 80B0H  | The module does not recognize the data record. |
| 0    | 1     | 80B2H  | The communication bus connection between the CPU and CP is not established. The corresponding CPU in the H system is in STOP mode. |
| 0    | 1     | 80C0H  | The data record cannot be read. |
| 0    | 1     | 80C1H  | The specified data record is currently being processed. |
| 0    | 1     | 80C2H  | There are too many jobs pending. |
| 0    | 1     | 80C3H  | CPU resources (memory) occupied. |
| 0    | 1     | 80C4H  | Communication error. The error occurs temporarily; it is usually best to repeat the job in the user program. |
| 0    | 1     | 8286H  | The value for the "Timeout" in the PING data block is outside the valid range of values. |
| 0    | 1     | 8287H  | The IP address specified in the ping DB is reserved and therefore not permitted. |
| 0    | 1     | 8288H  | The display occurs only with the PING result request command. Possible meanings:  
- The IP address was not reached by the ping command (for example because the CP was changed to STOP mode after the ping request command was sent);  
- The ping result has already been read out;  
- The ping result was not read out within the maximum time of 30 seconds. |
2.3 Program blocks for connection and system diagnostics

### Commands and job results - AG_CNTEX

#### Commands and evaluating the job results

The following table shows you the possible commands and the results that can be evaluated in the RESULT1/2 parameters.

Note

Command evaluation with older CP types or firmware versions

The commands described below are supported by the current CP types or firmware versions. You should also check the more detailed information under the following entry ID: 33414377 ([https://support.industry.siemens.com/cs/ww/en/view/33414377](https://support.industry.siemens.com/cs/ww/en/view/33414377))

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0   | NOP – no operation  
The block executes without a job being sent to the CP. |

<table>
<thead>
<tr>
<th>RESULT (for CMD = 0)</th>
<th>Hex value/range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001_H</td>
<td>Executed without error</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000_H</td>
<td>Default</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8289_H</td>
<td>The data volume for the ping request has exceeded the permitted range (maximum 1000 bytes; see data structure for the PING command)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>828A_H</td>
<td>There are already 4 ping requests being processed. New requests are only possible again after processing the existing requests.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>828B_H</td>
<td>There is already a PING request being processed for the specified IP address. Use the PING result request to complete the current processing.</td>
</tr>
</tbody>
</table>
### CMD Meaning

1 **CN_STATUS – connection status**

   This command returns the status of the connection selected with the ID.
   The CP is selected in the LADDR parameter.
   If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).

#### RESULT (for CMD = 1)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Bit</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 000*H</td>
<td></td>
<td>Bits 0-3: Codes for the send direction (excluded values: 0x2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 0: Connection type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: no send and receive connection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: Connection reserved for send and receive jobs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 1: Status of current job</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: No send job being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: Send job being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bits 2+3: Previous job:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 00: No information available on previous send job</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 01: previous send job completed successfully</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10: previous send job not completed successfully</td>
</tr>
</tbody>
</table>

(continued for CMD=1)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Bit</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 000*H</td>
<td></td>
<td>Bits 4-7: Codes for the receive direction (excluded values: 0x2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 4: Connection type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: No send and receive connection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: Connection reserved for send and receive jobs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 5: Status of current job</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: No receive job being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: Receive job being executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bits 6+7: Previous job:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 00: No information available on previous receive job</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 01: previous receive job completed successfully</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10: previous receive job not completed successfully</td>
</tr>
</tbody>
</table>
2.3 Program blocks for connection and system diagnostics

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CN_STATUS – connection status (continued for CMD=1)</td>
</tr>
</tbody>
</table>

This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter.

If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).

RESULT (for CMD = 1) | Value / meaning
--- | ---
| **Parameter** | **Hex value/range** | **Bit** |
| RESULT1 | 0000 0*00H | Bits 8-11: Codes for FETCH/WRITE (excluded values: 0x3, 0x7, 0x8, 0xB, 0xF) |
| Bit 8 | Connection type: |
| | • 0: No FETCH connection |
| | • 1: Connection reserved for FETCH jobs |
| Bit 9 | Connection type: |
| | • 0: No WRITE connection |
| | • 1: Connection reserved for WRITE jobs |
| Bit 10 | Job status (FETCH/WRITE): |
| | • 0: Job status OK |
| | • 1: Job status NOT OK |
| | This ID is set in the following situations: |
| | – The job was acknowledged negatively by the CPU |
| | – The job could not be forwarded to the CPU because the connection was in the "LOCKED" status. |
| | – The job was rejected because the FETCH/WRITE header did not have the correct structure. |
| Bit 11 | Status of FETCH/WRITE job |
| | • 0: No job active |
| | • 1: Job from LAN active |
2.3 Program blocks for connection and system diagnostics

### CMD 1: CN_STATUS – connection status (continued for CMD=1)

This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter.

If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).

<table>
<thead>
<tr>
<th>Parameter (for CMD = 1)</th>
<th>Value / meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 *000h</td>
</tr>
<tr>
<td>Aug 12-15: General CP information</td>
<td>Bits 12-15: General CP information (excluded values: 0x3, 0xB)</td>
</tr>
<tr>
<td>Bit 12 + 13</td>
<td>Information on connection status: (only available for SEND/RECV connections based on the ISO/RFC/TCP protocols, with UDP, the corresponding internal information is output)</td>
</tr>
<tr>
<td>Bit 13</td>
<td>• 00: Connection is terminated</td>
</tr>
<tr>
<td>Bit 14</td>
<td>• 01: Connection establishment active</td>
</tr>
<tr>
<td>Bit 15</td>
<td>• 10: Connection termination active</td>
</tr>
<tr>
<td>Bit 15</td>
<td>• 11: Connection is established</td>
</tr>
<tr>
<td>RESULT1</td>
<td>**** 0000h</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000h</td>
</tr>
</tbody>
</table>

---

### CMD 2: CN_RESET – connection reset

This command resets the connection selected with ID. The CP is selected in the LADDR parameter.

Resetting the connection means that a connection is aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted.

An entry is also generated in the diagnostics buffer in which the job result can be found.

<table>
<thead>
<tr>
<th>Parameter (for CMD = 2)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001h</td>
</tr>
<tr>
<td>The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment were triggered.</td>
<td></td>
</tr>
</tbody>
</table>
### Program blocks for SIMATIC NET S7 CPs

#### 2.3 Program blocks for connection and system diagnostics

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0002H</td>
<td>The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>CN_STATUS_ALL – all connections status</td>
</tr>
<tr>
<td></td>
<td>This command returns the connection status of all connections (established/terminated) in the RESULT1/2 parameters (at total of 8 bytes of group information).</td>
</tr>
<tr>
<td></td>
<td>The ID parameter must be set to “0” (checked for 0).</td>
</tr>
<tr>
<td></td>
<td>The CP is selected in the LADDR parameter.</td>
</tr>
<tr>
<td></td>
<td>When necessary, you can obtain detailed information about a terminated or unconfigured connection using a further connection status call with CMD=1.</td>
</tr>
</tbody>
</table>

#### RESULT (for CMD = 3)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>**** ****H</td>
<td>32 bits with the following validity:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For S7-400: Bits 0-31 for connections 1 - 32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For S7-300: Bits 0-15 for connections 1 - 16</td>
</tr>
<tr>
<td>RESULT2</td>
<td>**** ****H</td>
<td>32 bits with the following validity:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For S7-400: Bits 0-31 for connections 33 - 64</td>
</tr>
</tbody>
</table>

#### CMD 4

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>CN_RESET_ALL – all connections reset</td>
</tr>
<tr>
<td></td>
<td>This command resets all connections.</td>
</tr>
<tr>
<td></td>
<td>The ID parameter must be set to “0” (checked for 0).</td>
</tr>
<tr>
<td></td>
<td>The CP is selected in the LADDR parameter.</td>
</tr>
<tr>
<td></td>
<td>Resetting the connections means that connections are aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted.</td>
</tr>
<tr>
<td></td>
<td>An entry is also generated in the diagnostics buffer in which the job result can be found.</td>
</tr>
</tbody>
</table>

#### RESULT (for CMD = 4)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001H</td>
<td>The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment of all connections were triggered.</td>
</tr>
<tr>
<td>RESULT1</td>
<td>0000 0002H</td>
<td>The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000H</td>
<td>Default</td>
</tr>
</tbody>
</table>
### Program blocks for SIMATIC NET S7 CPs

#### 2.3 Program blocks for connection and system diagnostics

**CMD**  **Meaning**

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><strong>CN_CLEAR_RESET</strong> - Clear the reset ID</td>
</tr>
<tr>
<td></td>
<td>This command resets the reset ID (bit 15 in RESULT1) for the connection selected with ID.</td>
</tr>
<tr>
<td></td>
<td>The CP is selected in the LADDR parameter.</td>
</tr>
<tr>
<td></td>
<td>This job executes automatically when the connection status is read (CMD=1); the separate job described here is therefore only required in special situations.</td>
</tr>
</tbody>
</table>

**RESULT (for CMD = 5)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001H</td>
<td>The clear job was transferred to the CP successfully.</td>
</tr>
<tr>
<td>RESULT1</td>
<td>0000 0002H</td>
<td>The Clear job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000H</td>
<td>Default</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td><strong>CN_DISCON</strong> - connection disconnect</td>
</tr>
<tr>
<td></td>
<td>This command resets the connection selected with ID and LADDR.</td>
</tr>
<tr>
<td></td>
<td>Resetting the connection is achieved by aborting the connection.</td>
</tr>
<tr>
<td></td>
<td>Any data in the stack is lost without any message being displayed. The connection is not established again automatically afterwards. The connection can be established again with the <strong>CN_STARTCON</strong> control job. A diagnostics buffer entry is created in which you will find the job result.</td>
</tr>
</tbody>
</table>

**RESULT (for CMD = 6)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001H</td>
<td>The job was transferred to the CP successfully. The connection abort was initiated.</td>
</tr>
<tr>
<td>RESULT1</td>
<td>0000 0002H</td>
<td>The job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000H</td>
<td>Default</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMD</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><strong>CN_STARTCON</strong> - start connection</td>
</tr>
<tr>
<td></td>
<td>This command establishes a connection selected with ID and LADDR and aborted earlier with the control job <strong>CN_DISCON</strong>. A diagnostics buffer entry is created in which you will find the job result.</td>
</tr>
</tbody>
</table>

**RESULT (for CMD = 7)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Hex value/range</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT1</td>
<td>0000 0001H</td>
<td>The connection establishment job was transferred to the CP successfully. The connection establishment was initiated.</td>
</tr>
<tr>
<td>RESULT1</td>
<td>0000 0002H</td>
<td>The connection establishment job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0000H</td>
<td>Default</td>
</tr>
</tbody>
</table>
## Program blocks for SIMATIC NET S7 CPs

### 2.3 Program blocks for connection and system diagnostics

#### CMD 8: PING_REQUEST - Send a ping request
This command sends a ping command to the CP. The CP then initiates 4 ping echo requests to the specified IP address.

<table>
<thead>
<tr>
<th>RESULT (for CMD = 8)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Hex value/range</td>
</tr>
<tr>
<td>RESULT1</td>
<td>0000 0001H</td>
</tr>
<tr>
<td>RESULT2</td>
<td>0000 0002H</td>
</tr>
</tbody>
</table>

#### CMD 9: PING_RESULT - Query ping result
This command sends a ping result request to the CP. The CP transfers the results of the 4 executed ping echo requests in the RESULT parameter.

The call is successful when the 4 ping echo requests have been completed on the part of the CP.

<table>
<thead>
<tr>
<th>RESULT (for CMD = 9)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Hex value/range</td>
</tr>
<tr>
<td>RESULT1</td>
<td>**** ****H</td>
</tr>
<tr>
<td>RESULT2</td>
<td>**** ****H</td>
</tr>
</tbody>
</table>

Range of values for data words in RESULT1 / RESULT 2:

- 0000H: not used
- 0001H … EA60H: Reply time in ms 0001H = 1 ms EA60H = 60000 ms
- EA61H … FFFEH: not used
- FFFFH: Timeout: No echo within the specified monitoring time.
**Program blocks for FTP services**

2.4 Program blocks for FTP services

### Overview of FTP

FB and FCs for FTP services (FTP client)

The following list shows the program blocks available for the FTP client services.

**Note**

Note that the FTP client services of older SIMATIC S7-300 CPUs, for example the CPU 312 or CPU 315-1AF01, cannot be used because they do not support SFC24.

The following applies to S7-300s when using FC40 - FC44: The CP requires the supplied FC5 (AG_SEND) to start the FTP FCs; do not rename the FC5 for this application.
# Program blocks for Industrial Ethernet

## 2.4 Program blocks for FTP services

<table>
<thead>
<tr>
<th>Program block</th>
<th>can be used with:</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S7-300</td>
<td>S7-400</td>
</tr>
<tr>
<td>FTP_CMD (FB40)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Handling of complete FTP job sequences</td>
</tr>
<tr>
<td>FTP_CONNECT (FC40)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Establishing an FTP connection from client to server.</td>
</tr>
<tr>
<td>FTP_STORE (FC41)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transferring a DB from client to server.</td>
</tr>
<tr>
<td>FTP_RETRIEVE (FC42)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transferring a file from client to server.</td>
</tr>
<tr>
<td>FTP_DELETE (FC43)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deleting a file on the server.</td>
</tr>
<tr>
<td>FTP_QUIT (FC44)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Terminating a connection established with the ID.</td>
</tr>
</tbody>
</table>

### Using the program blocks

With the latest CP module types, you should preferably use FB40. As alternatives you can use FC40...FC44 if necessary and if they are available in the STEP 7 library.

### Requirement - configured FTP connection

To manage an FTP request sequence between the S7 station as FTP client and an FTP server, you will need to configure an FTP connection. To do this, first configure an unspecified TCP connection with the additional "Use FTP protocol" attribute.

### See also

Migration of FC 40-44 to FTP_CMD (Page 85)

### 2.4.2 FTP_CMD - universal program block for FTP services

#### Validity

FB40 can be used as of the following module types:

- As of CP 343-1 Advanced - GX30*) / GX31)
- As of CP 443-1 Advanced - GX20*) / GX30

*) The commands 33 (CONNECT_PASSIVE), 17 (CONNECT_TLS_PRIVATE) and 49 (CONNECT_TLS_PRIVATE_PASSIVE) cannot be used.

**) The commands 33 (CONNECT_PASSIVE) and 49 (CONNECT_TLS_PRIVATE_PASSIVE) cannot be used

The commands 33 "CONNECT_PASSIVE" and 49 "CONNECT_TLS_PRIVATE_PASSIVE" are only supported by the CP 443-1 Advanced as of firmware version V3.2 and as of block version 2.2.
2.4.2.1 Meaning and call - FTP_CMD

Meaning

Using FB40, you can establish FTP connections and transfer files from and to an FTP server.

FB40 replaces the FTP functions FC40 to FC44 used previously. The differences in these functions are mapped to a command parameter in FB40.

The advantages of FB40 are as follows:

- Simplification in the user program by using a command variable instead of different function calls
- The AG_SEND (FC5) function is not required here.
- The following commands with the parameter “CMD” provide additional functions:
  - APPEND
    The command allows data to be appended to an existing file.
  - RETR_PART
    The command allows selected data areas to be read from a file.
  - CONNECT_PASSIVE
    Command for establishing an FTP connection
    The client establishes the data connection, the server is in passive mode.
  - CONNECT_TLS_PRIVATE
    Command for establishing an SSL secured FTP connection
    The server actively establishes the FTP data connection.
  - CONNECT_TLS_PRIVATE_PASSIVE
    Command for establishing an SSL secured FTP connection
    The client establishes the data connection, the server is in passive mode.
Call interface

Call interface in FBD representation

<table>
<thead>
<tr>
<th>Type</th>
<th>ID</th>
<th>DONE</th>
<th>INT</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>LADDR</td>
<td>ERROR</td>
<td>BOOL</td>
<td>WORD</td>
</tr>
<tr>
<td>BYTE</td>
<td>CMD</td>
<td>STATUS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANY</td>
<td>NAME_STR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>FILE_DB_NR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DWORD</td>
<td>OFFSET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DWORD</td>
<td>LEN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2- 8   Example of a call in STL representation

```
CALL FB 40, DB 40 (        // Block call
  ID := 4,                 // FTP connection ID acc. to configuration
  LADDR := W#16#3FFD,     // Module address acc. to configuration
  CMD := B#16#3,          // The FTP command to be executed
  NAME_STR := P#DB44.DBX 170.0 BYTE 220, // Target data area address and length
  FILE_DB_NR := 42,       // Number of the data block
  OFFSET := DW#16#0,      // (not relevant in example)
  LEN := DW#16#0,        // (not relevant in example)
  DONE := M 420.1,       // Status parameter
  ERROR := M 420.2,      // Error information
  STATUS := MW 422);     // Status information
```

System functions called

The following system functions are called by the program block FTP_CMD:

SFC 1, SFC 20, SFC 24, SFC 58, SFC 59

Note

Note that the FTP client services of old SIMATIC S7-300 CPUs, for example the CPU 312 or CPU 315-1AF01, cannot be used because they do not support SFC24.
2.4.2.2 **Input parameter - FTP_CMD**

**Explanation of the input parameters**

Each FTP block call must be supplied with the following input parameters:

Table 2-9  Formal parameters of FB40 (FTP_CMD) - input parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Type</th>
<th>Range of values</th>
<th>Meaning / remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>INT</td>
<td>For S7-300: 1 ... 16</td>
<td>The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For S7-400: 1 ... 64</td>
<td></td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.</td>
<td></td>
</tr>
<tr>
<td>CMD</td>
<td>INPUT</td>
<td>BYTE</td>
<td>See table below - FTP commands in the &quot;CMD&quot; parameter</td>
<td>FTP commands executed when FB40 is called. You will find further information following the table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If a command is not supported by the CP firmware, an error message with STATUS = 8F6BH is output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Examples of FTP commands:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• RETRIEVE: B#16#3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• CONNECT_TLS_PRIVATE: B#16#11</td>
</tr>
<tr>
<td>NAME_STR</td>
<td>INPUT</td>
<td>ANY</td>
<td>Only &quot;BYTE&quot; is permitted as VARTYPE.</td>
<td>The address references a data block area. Here, you specify the address and length of the data area in which the target data is entered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• When CMD = 1, 33, 17, 49:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>With this command, the &quot;NAME_STR&quot; parameter specifies the FTP server to be addressed over the FTP connection with the following attributes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- IP address of the FTP server</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- User name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Password for the login</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>These values must be specified as consecutive strings in the destination range of the ANY pointer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• When CMD = 2, 3, 4, 6, 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>With this command, the &quot;NAME_STR&quot; parameter specifies the file name on the FTP server, in other words, the data source or data destination. The file name is specified as a string in the destination range of the ANY pointer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• When CMD = 5: Parameter not relevant</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>You will find example of content further below.</td>
</tr>
</tbody>
</table>
### 2.4 Program blocks for FTP services

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Type</th>
<th>Range of values</th>
<th>Meaning / remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_DB_NR</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>The data block specified here contains the file DB to be read / written.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The parameter is relevant only when CMD = 2, 3, 6 and 7.</td>
</tr>
<tr>
<td>OFFSET</td>
<td>INPUT</td>
<td>DWORD</td>
<td></td>
<td>Only when CMD = 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Offset in bytes starting at which the file will be read.</td>
</tr>
<tr>
<td>LEN</td>
<td>INPUT</td>
<td>DWORD</td>
<td></td>
<td>Only when CMD = 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sublength in bytes that is read starting at the value specified in &quot;OFFSET&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Special features:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• If &quot;DW#16#FFFFFFFF&quot; is specified, the available rest of the file will be read.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Result OK (DONE = 1, STATUS = 0) if no other error occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• When OFFSET &gt; length of the original file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The length of the destination file is displayed in this case in the ACT_LENGTH parameter in the file DB: 0 bytes on the CPU.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Result OK (DONE = 1, STATUS = 0) if no other error occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• When OFFSET + LEN &gt; length of the original file (and LEN ≠ 0xFFFFFFFF):</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The length of the destination file is displayed in this case in the ACT_LENGTH parameter in the file DB: Available bytes starting at &quot;OFFSET&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Result OK (DONE = 1, STATUS = 0) if no other error occurred.</td>
</tr>
</tbody>
</table>

### FTP commands in the "CMD" parameter

The following table explains the meaning of the commands of the "CMD" parameter and which input parameters need to be supplied. The ID and LADDR parameters must always be set to identify the connection.
Table 2- 10 FTP commands in the "CMD" parameter

<table>
<thead>
<tr>
<th>Command</th>
<th>Relevant input parameters (in addition to ID and LADDR)</th>
<th>Meaning / handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (NOOP)</td>
<td>-</td>
<td>The called FC does not execute any action. The status codes are set as follows when these parameters are supplied: DONE=1; ERROR=0; STATUS=0</td>
</tr>
<tr>
<td>1 (CONNECT)</td>
<td>NAME_STR</td>
<td>With this command, the FTP client establishes an FTP connection to an FTP server (active FTP). The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user. Sequence (shortened): 1. The client sends a query to the control port 21 of the server. The client also sends the number of the data port to the server and opens it. 2. The server acknowledges the query to the client. 3. The server establishes the data connection to the client from its port 20.</td>
</tr>
<tr>
<td>33 (CONNECT_PASSIVE)</td>
<td>NAME_STR</td>
<td>With this command, the FTP client establishes an FTP connection to an FTP server (passive FTP). The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user. The use of this command is recommended if the client is located in a protected (internal) subnet. Sequence (shortened): 1. The client sends a query to the control port 21 of the server. The client sends the &quot;PASV&quot; command to the server that does not then establish a data connection. 2. The server acknowledges the query and sends the client its port number for the data connection. 3. The client establishes the data connection to the server with the port number specified by the server.</td>
</tr>
<tr>
<td>2 (STORE)</td>
<td>NAME_STR, FILE_DB_NR</td>
<td>This function call transfers a data block (file DB) from the FTP client (S7-CPU) to the FTP server. Caution: If the file (file DB) already exists on the FTP server, it will be overwritten.</td>
</tr>
<tr>
<td>3 (RETRIEVE)</td>
<td>NAME_STR, FILE_DB_NR</td>
<td>This function call transfers a file from the FTP server to the FTP client (S7-CPU). Caution: If the data block (file DB) on the FTP client already contains a file, it will be overwritten.</td>
</tr>
<tr>
<td>4 (DELETE)</td>
<td>NAME_STR</td>
<td>With this function call, you delete a file on the FTP server.</td>
</tr>
<tr>
<td>5 (QUIT)</td>
<td>-</td>
<td>With this function call, you establish the FTP connections selected with the ID.</td>
</tr>
<tr>
<td>Command</td>
<td>Relevant input parameters (in addition to ID and LADDR)</td>
<td>Meaning / handling</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>6 (APPEND)</td>
<td>NAME_STR, FILE_DB_NR</td>
<td>Similar to &quot;STORE&quot;, the &quot;APPEND&quot; command saves a file on the FTP server. With &quot;APPEND&quot;, the file on the FTP server is, however, not overwritten. The new content is appended to the existing file. If the file (file DB) does not exist on the FTP server, it will be created.</td>
</tr>
<tr>
<td>7 (RETR_PART)</td>
<td>NAME_STR, FILE_DB_NR, OFFSET, LEN</td>
<td>Using the &quot;RETR_PART&quot; command (retrieve part), you can request a section of a file from the FTP server. If very large files are involved, this allows you to restrict the read to the part you currently require. To do this, you need to know the structure of the file. Enter the required part of the file using the two parameters &quot;OFFSET&quot; and &quot;LEN&quot; in FB40.</td>
</tr>
<tr>
<td>17 (CONNECT_TLS_PRIVATE)</td>
<td>NAME_STR</td>
<td>With the &quot;CONNECT_TLS_PRIVATE&quot; command, the FTP client sets up an active SSL-secured FTP connection (FTPS) to the FTP server (active FTP). The data of the control connection and the data connection is transferred encrypted. The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user. Requirements: The FTPS server certificate must be imported into the CP configuration (Security). Sequence (shortened): 1. The client sends a query to the control port 21 of the server. The client also sends the number of the data port to the server and opens it. 2. The server acknowledges the query to the client. 3. The server establishes the data connection to the client from its port 20.</td>
</tr>
<tr>
<td>49 (CONNECT_TLS_PRIVATE_PASSIVE)</td>
<td>NAME_STR</td>
<td>With the &quot;CONNECT_TLS_PRIVATE&quot; command, the FTP client sets up an SSL-secured FTP connection (FTPS) to the FTP server (passive FTP). The data of the control connection and the data connection is transferred encrypted. The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user. Requirements: The FTPS server certificate must be imported into the CP configuration (Security). Sequence (shortened): 1. The client sends a query to the control port 21 of the server. The client sends the “PASV” command to the server that does not then establish a data connection. 2. The server acknowledges the query and sends the client its port number for the data connection. 3. The client establishes the data connection to the server with the port number specified by the server.</td>
</tr>
</tbody>
</table>
Examples of the content of the "NAME_STR" parameter

The parameter record has the following content:

Table 2-11 Content of the parameter record for CMD = 1, 17, 33, 49

<table>
<thead>
<tr>
<th>Relative address ²</th>
<th>Name</th>
<th>Type ¹</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>ip_address</td>
<td>STRING[100]</td>
<td>'142.11.25.135'</td>
<td>IP address of the FTP server</td>
</tr>
<tr>
<td>102.0</td>
<td>username</td>
<td>STRING[32]</td>
<td>'user'</td>
<td>User name for the login on the FTP server</td>
</tr>
<tr>
<td>136.0</td>
<td>password</td>
<td>STRING[32]</td>
<td>'password'</td>
<td>Password for the login on the FTP server</td>
</tr>
</tbody>
</table>

1) The maximum possible string length is specified
2) The specified values relate to the string lengths specified in "Type".

Table 2-12 Content of the parameter record for CMD = 2, 3, 4, 6, 7

<table>
<thead>
<tr>
<th>Relative address ²</th>
<th>Name</th>
<th>Type ¹</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>170.0</td>
<td>filename</td>
<td>STRING[220]</td>
<td>'plant1/tank2/press.dat'</td>
<td>Name of the destination or source file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or STRING[212]</td>
<td></td>
<td>Note: When CMD=7 (RETR_PART), the maximum length of the file name is limited to 212 characters.</td>
</tr>
</tbody>
</table>

1) The maximum possible string length is specified
2) The specified values relate to the string lengths specified in "Type".
2.4.2.3 Output parameters and status information - FTP_CMD

Introduction

For status evaluation, the following parameters must be evaluated in the user program:

Table 2-13 Formal parameters of FB40 (FTP_CMD) - output parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Type</th>
<th>Range of values</th>
<th>Meaning / remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Job done</td>
<td>This parameter indicates whether or not the job was completed without errors.</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Error/fault</td>
<td>Error code This parameter signals that the job could not be executed error-free.</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>See following table</td>
<td>Status code This parameter supplies detailed information about the execution of the job.</td>
</tr>
</tbody>
</table>

The DONE, ERROR and STATUS parameters are updated at every block call.

Example

During job execution, the FB40 returns the following codes:

- DONE=0
- ERROR=0
- STATUS=8181H

Meaning: Job still running.

Evaluating status codes

Note

For entries coded with 8FxxH in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 2-14 FB 40: Meaning of the STATUS parameter in conjunction with DONE and ERROR

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0000H</td>
<td>No job being executed.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td>DONE</td>
<td>ERROR</td>
<td>STATUS</td>
<td>Meaning</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Job active. If 8181H is indicated permanently: The CP is not released for FB40. An illegal command was for the firmware version was called: CMD 6 / CMD 7 / CMD 17 / CMD 33 / CMD 49</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8090H</td>
<td>• No module with this module start address exists.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The block being used does not match the system family being used (remember to use different blocks for S7300 and S7400).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8091H</td>
<td>Module start address not at a doubleword boundary</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8092H</td>
<td>Type information in the ANY pointer is not byte</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80A4H</td>
<td>The communication bus connection between the CPU and CP is not established (with newer CPU versions). This can, for example, be caused by the following: • No connection configuration • Maximum number of CPs operating at the same time was exceeded</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B0H</td>
<td>The module does not recognize the data record.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B1H</td>
<td>Destination area invalid; for example, destination area &gt; 240 bytes.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B2H</td>
<td>The communication bus connection between the CPU and CP is not established (with older CPU versions). (with newer CPU versions, see 80A4H)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C0H</td>
<td>The data record cannot be read.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C1H</td>
<td>The specified data record is currently being processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C2H</td>
<td>There are too many jobs pending.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C3H</td>
<td>Resources occupied (memory).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C4H</td>
<td>Communication error (occurs temporarily, it is usually best to repeat the job in the user program).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80D2H</td>
<td>Module start address incorrect.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8180H</td>
<td>There is no data available yet.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>The configuration does not match the job parameters.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8184H</td>
<td>Illegal data type specified for the NAME_STR parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8185H</td>
<td>Destination buffer (LEN) is too small.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8186H</td>
<td>ID parameter invalid. ID = 1, 2,...,64</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8302H</td>
<td>No receive resources available on the destination station.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8304H</td>
<td>FTP connection not established. Wrong or non-existent connection ID.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F22H</td>
<td>Source area invalid, for example: Area does not exist in the DB</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F23H</td>
<td>Area length error writing a parameter (e.g. DB too short).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F24H</td>
<td>Range error when reading a parameter</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F28H</td>
<td>Alignment error reading a parameter</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F32H</td>
<td>Parameter contains a DB number that is too high.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F33H</td>
<td>DB number error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F3AH</td>
<td>Area not loaded (DB)</td>
</tr>
</tbody>
</table>
### Program blocks for FTP services

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8F50H</td>
<td>File DB DB 0 or DB does not exist</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F51H</td>
<td>Specified file DB data area larger than existing data area</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F52H</td>
<td>File DB in write-protected memory</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F53H</td>
<td>File DB max. length &lt; current length</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F54H</td>
<td>File DB does not contain any valid data</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F55H</td>
<td>Header status bit: Locked</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F56H</td>
<td>The NEW bit in the file DB header was not reset</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F57H</td>
<td>The FTP client does not have write access to the file DB but rather the FTP server (header status bit: WriteAccess).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F60H</td>
<td>Bad user data, for example bad IP address of the FTP server</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F61H</td>
<td>FTP server not obtainable</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F62H</td>
<td>Possible meanings:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Job not supported or rejected by FTP server</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The FTP server does not support SSL-secured connections.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F63H</td>
<td>File transfer aborted by the FTP server</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F64H</td>
<td>Error on the FTP control connection; data could not be sent or received; the FTP control connection must be established again after such an error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F65H</td>
<td>Error on the FTP data connection; data could not be sent or received.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The job must be called again.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This error can, for example, be caused by RETRIEVE (CMD=3) when the addressed file is already open on the FTP server.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F66H</td>
<td>Error reading/writing data from/to the CPU (for example DB does not exist or too short)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F67H</td>
<td>Error in the FTP client on the CP; for example attempting to open more than the maximum number of FTP connections.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F68H</td>
<td>The job was rejected by the FTP client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This error can, for example, be caused by RETRIEVE (CMD=3) when the value for the parameter MAX_LENGTH was selected too low in the file DB header.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F69H</td>
<td>The FTP connection in an incorrect status, for example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The connection is called without a previous connection termination (with the same connection ID)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• There is a connection termination for a connection that has already been terminated;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• A STORE command was sent on a connection that is not established.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F6AH</td>
<td>The connection could not be established due to a temporary resource bottleneck.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Remedy: Repeat the block call.</td>
</tr>
</tbody>
</table>
### Program blocks for Industrial Ethernet

#### 2.4 Program blocks for FTP services

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8F6BH</td>
<td>Possible causes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Wrong value for the CMD parameter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• An FB40 command is not supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Possible cause: Wrong firmware on the CP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Remedy: Firmware update (with older CPs, use the functions FC 40...FC 44 instead of FB 40.)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F6CH</td>
<td>A value &gt; 7FFF FFFH was set in the OFFSET parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F6DH</td>
<td>The FTP client does not support SSL-secured connections.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F6EH</td>
<td>The signature of the certificate is invalid.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F6FH</td>
<td>Possible causes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The certificate contains an invalid value for &quot;notBefore&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The certificate is invalid: The &quot;notBefore&quot; entry contains a time after the current time.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F70H</td>
<td>Possible causes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The certificate contains an invalid value for &quot;notAfter&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The certificate has expired: The &quot;notAfter&quot; entry contains a time before the current time.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F71H</td>
<td>The CA certificate of a non-trustworthy device certificate could not be found.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F72H</td>
<td>The original CA certificate is invalid. This is either not a CA certificate or its expansions are not consistent with the intended purpose.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F73H</td>
<td>The original CA certificate is marked as not trustworthy for the specified purpose.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F74H</td>
<td>Other errors occurred during the verification of a certificate.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F7FH</td>
<td>Internal error, for example illegal ANY reference</td>
</tr>
</tbody>
</table>

#### 2.4.2.4 Migration of FC 40-44 to FTP_CMD

**Comparison of the function block FB40 with older functions FC40...44**

All CPs with FTP functionality support the functions FC40...44. This means that existing user programs can be used unchanged.

If you want to convert from the FTP functions FC40...44 to FB40, you will need to modify your user program.

The following table shows the FB40 commands used to replace the functions FC40...44.

- Correlation is indicated by "X".
- Where there is no correlation, this is indicated by ",".
Program blocks for SIMATIC NET S7 CPs

Programming Manual, 11/2015, C79000-G8976-C229-08

2.4 Program blocks for FTP services

### Commands of the "CMD" parameter of FTP_CMD

<table>
<thead>
<tr>
<th>Older FTP functions FC40...44</th>
<th>CMD = 1</th>
<th>CMD = 2</th>
<th>CMD = 3</th>
<th>CMD = 4</th>
<th>CMD = 5</th>
<th>CMD = 6 / 7 / 17 / 33 / 49</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC40</td>
<td>X 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC41</td>
<td></td>
<td>X 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC42</td>
<td></td>
<td></td>
<td>X 3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC43</td>
<td></td>
<td></td>
<td></td>
<td>X 4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

1) ...4) The parameters of FC40...43 and CMD 1...4 (FB40) are not identical (see table below).

The corresponding parameters that specify a particular function in the functions FC40...FC43 or in the commands of FB40 are listed in the following table.

<table>
<thead>
<tr>
<th>Parameters of the FC</th>
<th>Parameters in FTP_CMD (with CMD 1...4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC40: LOGIN</td>
<td>CMD = 1: NAME_STR</td>
</tr>
<tr>
<td>FC41: FILE_NAME</td>
<td>CMD = 2: NAME_STR</td>
</tr>
<tr>
<td>FC42: FILE_NAME</td>
<td>CMD = 3: NAME_STR</td>
</tr>
<tr>
<td>FC43: FILE_NAME</td>
<td>CMD = 4: NAME_STR</td>
</tr>
<tr>
<td>FC40...43: BUFFER_DB_NR</td>
<td>Omitted (replaced by instance DB)</td>
</tr>
</tbody>
</table>

2.4.3 FTP_CONNECT

2.4.3.1 Meaning and call - FTP_CONNECT

**Meaning**

With this function call, the FTP client establishes an FTP connection to an FTP server.

The IP address of the FTP server, the user identification (username) and (if necessary) the password for user identification must be transferred to the FTP server.

All further access by the FTP client is then based on this user identification if you use the same FTP connection ID. Data is then exchanged with the FTP server specified for this user.
Call interface

### Example of a call in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call fc40 {</td>
<td></td>
</tr>
<tr>
<td>ACT := M 420.0,</td>
<td>//Call FTP_CONNECT</td>
</tr>
<tr>
<td>ID := 4,</td>
<td>//Job triggered by memory bit</td>
</tr>
<tr>
<td>LADDR := W#16#3FFD,</td>
<td>//FTP connection ID acc. to configuration</td>
</tr>
<tr>
<td>LOGIN := P#DB40.DBX 0.0 BYTE 170,</td>
<td>//Module address acc. to configuration</td>
</tr>
<tr>
<td>BUFFER_DB_NR := 9,</td>
<td>//Information for LOGIN in DB40</td>
</tr>
<tr>
<td>DONE := M 420.1,</td>
<td>//Buffer area for FTP service</td>
</tr>
<tr>
<td>ERROR := M 420.2,</td>
<td></td>
</tr>
<tr>
<td>STATUS := MW 422);</td>
<td></td>
</tr>
</tbody>
</table>

### 2.4.3.2 Explanation of the formal parameters - FTP_CONNECT

**Explanation of the general call parameters**

The general parameters have the same significance in every FTP function call; they are therefore described in one section.
Explanation of the formal parameters specific to the call

Table 2-15  Formal parameters for FTP_CONNECT

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGIN</td>
<td>INPUT</td>
<td>ANY</td>
<td>This parameter specifies the FTP server to be accessed on the FTP connection.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(only the following are permitted as VARTYPE: BYTE)</td>
<td>(for further details, refer to the following table)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Here, you specify the address and length of the data area in which the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>target data are entered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The address references a data block area.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The ANY pointer data type is used to address this area. For more detailed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>information on this data type, refer to the STEP 7 online help under the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>appendix topic &quot;Format of the parameter type ANY&quot;.</td>
</tr>
<tr>
<td>BUFFER_DB_NR</td>
<td>INPUT</td>
<td>INT</td>
<td>Here, you enter a data block required as a buffer area by the FTP client</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>for FTP transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can use the same data block as the buffer area for all FTP jobs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: The length of the reserved DB must be at least 255 bytes!</td>
</tr>
</tbody>
</table>

LOGIN parameter

This parameter record has the following content for FTP_CONNECT:

<table>
<thead>
<tr>
<th>Relative address 2)</th>
<th>Name</th>
<th>Type 1)</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>ip_address</td>
<td>STRING[100]</td>
<td>'142.11.25.135'</td>
<td>IP address of the FTP server.</td>
</tr>
<tr>
<td>102.0</td>
<td>username</td>
<td>STRING[32]</td>
<td>'user'</td>
<td>User name for the login on the FTP server.</td>
</tr>
<tr>
<td>136.0</td>
<td>password</td>
<td>STRING[32]</td>
<td>'password'</td>
<td>Password for the login on the FTP server.</td>
</tr>
<tr>
<td>170.0</td>
<td>filename</td>
<td>STRING[220]</td>
<td>'plant1/tank2/press.dat'</td>
<td>Name of the destination or source file 3)</td>
</tr>
</tbody>
</table>

1) in each case, the maximum possible string length is specified
2) The specified values relate to the string lengths specified in "Type".
3) These rows are irrelevant for this call.

2.4.4 FTP_STORE

2.4.4.1 Meaning and call - FTP_STORE

Meaning

This function call transfers a data block (file DB) from the FTP client (S7-CPU) to the FTP server.
You need to specify the data block that contains the file. You will also have to select the path/file name under which the file will be created on the FTP server.

If the file (file DB) already exists on the FTP server, it will be overwritten.

**Call interface**

<table>
<thead>
<tr>
<th>FTP_STORE</th>
<th>FTP_STORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL ACT</td>
<td>BOOL DONE</td>
</tr>
<tr>
<td>INT ID</td>
<td>INT ERROR</td>
</tr>
<tr>
<td>WORD LADDR</td>
<td>WORD STATUS</td>
</tr>
<tr>
<td>ANY FILE_NAME</td>
<td>INT BUFFER_DB_NR</td>
</tr>
<tr>
<td>INT FILE_DB_NR</td>
<td></td>
</tr>
</tbody>
</table>

**Example of a call in STL representation**

```stl
STL                                      Explanation
CALL fc41 {
ACT := M 420.0,
ID := 4,
LADDR := W#16#3FFD,
FILE_NAME := P#DB40.DBX 170.0 BYTE 220,
BUFFER_DB_NR := 9,
FILE_DB_NR := 42,
DONE := M 420.1,
ERROR := M 420.2,
STATUS := MW 422};
```

**2.4.4.2 Explanation of the formal parameters - FTP_STORE**

**Explanation of the general call parameters**

The general parameters have the same significance in every FTP function call; they are therefore described in one section.
Explanation of the formal parameters specific to the call

Table 2-16  Formal parameters for FTP_STORE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_NAME</td>
<td>INPUT</td>
<td>ANY</td>
<td>This parameter specifies the data destination. Here, you specify the address and length of the data area in which the target data are entered. The address references a data block area. The ANY pointer data type is used to address this area. For more detailed information on this data type, refer to the STEP 7 online help under the appendix topic &quot;Format of the parameter type ANY&quot;.</td>
</tr>
<tr>
<td>BUFFER_DB_NR</td>
<td>INPUT</td>
<td>INT</td>
<td>Here, you enter a data block required as a buffer area by the FTP client for FTP transfer. You can use the same data block as the buffer area for all FTP jobs. Note: The length of the reserved DB must be at least 255 bytes!</td>
</tr>
<tr>
<td>FILE_DB_NR</td>
<td>INPUT</td>
<td>INT</td>
<td>The data block specified here contains the file DB to be read.</td>
</tr>
</tbody>
</table>

FILE_NAME parameter

This parameter record has the following content for FTP_STORE:

<table>
<thead>
<tr>
<th>Relative address 2)</th>
<th>Name</th>
<th>Type 1)</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>ip_address</td>
<td>STRING[100]</td>
<td>'142.11.25.135'</td>
<td>IP address of the FTP server. 3)</td>
</tr>
<tr>
<td>102.0</td>
<td>username</td>
<td>STRING[32]</td>
<td>'user'</td>
<td>User name for the login on the FTP server. 3)</td>
</tr>
<tr>
<td>136.0</td>
<td>password</td>
<td>STRING[32]</td>
<td>'password'</td>
<td>Password for the login on the FTP server. 3)</td>
</tr>
<tr>
<td>170.0</td>
<td>filename</td>
<td>STRING[220]</td>
<td>'plant1/tank2/press.dat'</td>
<td>Name of the destination or source file</td>
</tr>
</tbody>
</table>

1) in each case, the maximum possible string length is specified
2) The specified values relate to the string lengths specified in "Type".
3) These rows are irrelevant for this call.

2.4.5 FTP_RETRIEVE

2.4.5.1 Meaning and call - FTP_RETRIEVE

Meaning

This function call transfers a file from the FTP server to the FTP client (S7-CPU).
You need to specify the data block in which the file will be entered. You will also have to select the path/file name under which the file is located on the FTP server.

If the data block (file DB) on the FTP client already contains a file, it will be overwritten.

Call interface

Example of a call in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call fc42 {</td>
<td>//Call FTP RETRIEVE</td>
</tr>
<tr>
<td>ACT := M 420.0,</td>
<td>//Job triggered by memory bit</td>
</tr>
<tr>
<td>ID := 4,</td>
<td>//FTP connection ID acc. to configuration</td>
</tr>
<tr>
<td>LADDR := W#16#3FFD,</td>
<td>//Module address acc. to configuration</td>
</tr>
<tr>
<td>FILE_NAME := P#DB40.DBX 170.0 BYTE 220,</td>
<td>//Information for source file in DB40</td>
</tr>
<tr>
<td>BUFFER_DB_NR := 9,</td>
<td>//Buffer area for FTP service</td>
</tr>
<tr>
<td>FILE_DB_NR := 42,</td>
<td>DB no. in target file</td>
</tr>
<tr>
<td>DONE := M 420.1,</td>
<td></td>
</tr>
<tr>
<td>ERROR := M 420.2,</td>
<td></td>
</tr>
<tr>
<td>STATUS := MW 422);</td>
<td></td>
</tr>
</tbody>
</table>

2.4.5.2 Explanation of the formal parameters - FTP RETRIEVE

Explanation of the general call parameters

The general parameters have the same significance in every FTP function call; they are therefore described in one section.
Explanation of the formal parameters specific to the call

Table 2-17 Formal parameters for FTP RETRIEVE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_NAME</td>
<td>INPUT</td>
<td>ANY</td>
<td>This parameter specifies the data source. (for further details, refer to the following table) Here, you specify the address and length of the data area in which the target data are entered. The address references a data block area. The ANY pointer data type is used to address this area. For more detailed information on this data type, refer to the STEP 7 online help under the appendix topic &quot;Format of the parameter type ANY&quot;.</td>
</tr>
<tr>
<td>BUFFER_DB_NR</td>
<td>INPUT</td>
<td>INT</td>
<td>Here, you enter a data block required as a buffer area by the FTP client for FTP transfer. You can use the same data block as the buffer area for all FTP jobs. Note: The length of the reserved DB must be <strong>at least 255 bytes</strong>!</td>
</tr>
<tr>
<td>FILE_DB_NR</td>
<td>INPUT</td>
<td>INT</td>
<td>The data block specified here contains the file DB to be written (data destination).</td>
</tr>
</tbody>
</table>

**FILE_NAME parameter**

This parameter record has the following content for FTP RETRIEVE:

<table>
<thead>
<tr>
<th>Relative address</th>
<th>Name</th>
<th>Type</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>ip_address</td>
<td>STRING[100]</td>
<td>'142.11.25.135'</td>
<td>IP address of the FTP server.</td>
</tr>
<tr>
<td>102.0</td>
<td>username</td>
<td>STRING[32]</td>
<td>'user'</td>
<td>User name for the login on the FTP server.</td>
</tr>
<tr>
<td>136.0</td>
<td>password</td>
<td>STRING[32]</td>
<td>'password'</td>
<td>Password for the login on the FTP server.</td>
</tr>
<tr>
<td>170.0</td>
<td>filename</td>
<td>STRING[220]</td>
<td>'plant1/tank2/press.dat'</td>
<td>Name of the destination or source file</td>
</tr>
</tbody>
</table>

1) in each case, the maximum possible string length is specified
2) The specified values relate to the string lengths specified in "Type".
3) These rows are irrelevant for this call.

2.4.6 FTP_DELETE

2.4.6.1 Meaning and call - FTP_DELETE

**Meaning**

With this function call, you delete a file on the FTP server.
Call interface

<table>
<thead>
<tr>
<th>Call interface</th>
<th>FTP_DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL ACT</td>
<td>BOOL DONE</td>
</tr>
<tr>
<td>INT ID</td>
<td>BOOL ERROR</td>
</tr>
<tr>
<td>WORD LADDR</td>
<td>WORD STATUS</td>
</tr>
<tr>
<td>ANY FILE_NAME</td>
<td>INT BUFFER_DB_NR</td>
</tr>
</tbody>
</table>

Example of a call in STL representation

```
STL | Explanation
---|--------------
call fc43 {   | //Call FTP_DELETE
  ACT := M 420.0, | //Job triggered by memory bit
  ID := 4, | //FTP connection ID acc. to configuration
  LADDR := W#16#3FFD, | //Module address acc. to configuration
  FILE_NAME := P#DB40.DBX 170.0 BYTE 220, | //Information for target file in DB40
  BUFFER_DB_NR := 9, | //Buffer area for FTP service
  DONE := M 420.1, |
  ERROR := M 420.2, |
  STATUS := MW 422); |
```

2.4.6.2 Explanation of the formal parameter - FTP_DELETE

Explanation of the general call parameters

The general parameters have the same significance in every FTP function call; they are therefore described in one section.
2.4 Program blocks for FTP services

Explanation of the formal parameters specific to the call

Table 2-18 Formal parameters for FTP_DELETE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_NAME</td>
<td>INPUT</td>
<td>ANY</td>
<td>This parameter specifies the data destination. (for further details, refer to the following table) Here, you specify the address and length of the data area in which the target data are entered. The address references a data block area. The ANY pointer data type is used to address this area. For more detailed information on this data type, refer to the STEP 7 online help under the appendix topic &quot;Format of the parameter type ANY&quot;.</td>
</tr>
<tr>
<td>BUFFER_DB_NR</td>
<td>INPUT</td>
<td>INT</td>
<td>Here, you enter a data block required as a buffer area by the FTP client for FTP transfer. You can use the same data block as the buffer area for all FTP jobs. Note: The length of the reserved DB must be at least 255 bytes!</td>
</tr>
</tbody>
</table>

LOGIN parameter

This parameter record has the following content for FTP_DELETE:

<table>
<thead>
<tr>
<th>Relative address 2)</th>
<th>Name</th>
<th>Type 1)</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>ip_address</td>
<td>STRING[100]</td>
<td>'142.11.25.135'</td>
<td>IP address of the FTP server. 3)</td>
</tr>
<tr>
<td>102.0</td>
<td>username</td>
<td>STRING[32]</td>
<td>'user'</td>
<td>User name for the login on the FTP server. 3)</td>
</tr>
<tr>
<td>136.0</td>
<td>password</td>
<td>STRING[32]</td>
<td>'password'</td>
<td>Password for the login on the FTP server. 3)</td>
</tr>
<tr>
<td>170.0</td>
<td>filename</td>
<td>STRING[220]</td>
<td>'plant1/tank2/press.dat'</td>
<td>Name of the destination or source file</td>
</tr>
</tbody>
</table>

1) in each case, the maximum possible string length is specified
2) The specified values relate to the string lengths specified in "Type".
3) These rows are irrelevant for this call.

2.4.7 FTP_QUIT

2.4.7.1 Meaning and call - FTP_QUIT

Meaning

With this function call, you establish the FTP connections selected with the ID.
2.4 Program blocks for FTP services

Call interface

Example of a call in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call fc44 {</td>
<td>//Call FTP.Quit</td>
</tr>
<tr>
<td>ACT := M 420.0,</td>
<td>//Job triggered by memory bit</td>
</tr>
<tr>
<td>ID := 4,</td>
<td>//FTP connection ID acc. to configuration</td>
</tr>
<tr>
<td>LADDR := W#16#3FFD,</td>
<td>//Module address acc. to configuration</td>
</tr>
<tr>
<td>DONE := M 420.1,</td>
<td></td>
</tr>
<tr>
<td>ERROR := M 420.2,</td>
<td></td>
</tr>
<tr>
<td>STATUS := MW 422);</td>
<td></td>
</tr>
</tbody>
</table>

Note

The output of FC44 must be assigned a memory word as value. If you enter DBx.DWy, an error message is displayed (applies only to S7-300).

2.4.7.2 Explanation of the formal parameters - FTP.Quit

Explanation of the general call parameters

The general parameters have the same significance in every FTP function call; they are therefore described in one section.
2.4.8 Parameters for CP and connection assignment (input parameters)

Parameters for CP and connection assignment (input parameters)

Apart from the input parameters specific to the jobs started with each FTP block call, the following general input parameters must also have values supplied to them:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Type 1)</th>
<th>Possible values</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>INPUT</td>
<td>BOOL</td>
<td>0.1</td>
<td>The parameter contains the initialization bit for triggering the job.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• If ACT = 1, the job is executed. During job execution, the FC returns the following codes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- DONE=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- ERROR=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- STATUS=8181H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• If ACT = 0, the called FC does not execute any actions; the status codes are then as follows for these parameters settings:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- DONE=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- ERROR=1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- STATUS=8F70H</td>
</tr>
<tr>
<td>Note / recommendation:</td>
<td></td>
<td></td>
<td></td>
<td>You should execute the FTP calls conditionally in your application, for example by evaluating the codes. It is not a good idea to control the call using the ACT bit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The ACT bit must be set to 1 until the execution is signaled by the DONE bit.</td>
</tr>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>INT</td>
<td>1,2...64</td>
<td>The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.</td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When you call an FC, you transfer the module start address of the ADVANCED-CP in the LADDR parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>You will find the module start address of the ADVANCED-CP in the configuration of the properties of the ADVANCED-CP in &quot;Addresses &gt; Inputs&quot;.</td>
</tr>
</tbody>
</table>

**NOTICE**

Make sure that only one FTP client program block is called per user ID as long as ACT = 1 is set.

For example, FTP_STORE and FTP_RETRIEVE must not be running on the same FTP connection at the same time. This requirement corresponds to normal FTP functionality.

Otherwise you cannot rely on the correctness of the output parameters (DONE bit, ERROR bit and STATUS word).
2.4.9 Status information (output parameters)

Status information (output parameters)

For status evaluation, the following parameters must be evaluated in the user program:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Type</th>
<th>Possible values</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Job done</td>
<td>This parameter indicates whether or not the job was completed without errors.</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Error situation</td>
<td>Error code This parameter signals that the job could not be executed error-free.</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>See following table</td>
<td>Status code This parameter supplies detailed information about the execution of the job.</td>
</tr>
</tbody>
</table>

Note

For FC FTP_QUIT, use only the data type memory word for the STATUS parameter (applies only to the CP 343-1 IT).

Example

During job execution, the FC returns the following codes:

- DONE=0
- ERROR=0
- STATUS=8181H

Evaluating status codes

Remember that the status codes DONE, ERROR, STATUS are updated at each block call.

Note

For entries coded with 8FxxH in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0000H</td>
<td>No job being executed.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Job active.</td>
</tr>
<tr>
<td>DONE</td>
<td>ERROR</td>
<td>STATUS</td>
<td>Meaning</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| 0    | 1     | 8090H  | • No module with this module start address exists.  
|      |       |        | • The FC being used does not match the system family being used (remember to use different FCs for S7300 and S7400). |
| 0    | 1     | 8091H  | Module start address not at a doubleword boundary. |
| 0    | 1     | 8092H  | Type information in the ANY pointer is not byte. |
| 0    | 1     | 80A4H  | The communication bus connection between the CPU and CP is not established.  (with newer CPU versions).  
|      |       |        | This can, for example, be caused by the following:  
|      |       |        | • No connection configuration;  
|      |       |        | • Maximum number of CPs operating at the same time was exceeded. |
| 0    | 1     | 80B0H  | The module does not recognize the data record. |
| 0    | 1     | 80B1H  | Destination area invalid.  
|      |       |        | for example, destination area > 240 bytes. |
| 0    | 1     | 80B2H  | The communication bus connection between the CPU and CP is not established (with older CPU versions; otherwise 80A4H; for further information, refer to this code). |
| 0    | 1     | 80C0H  | The data record cannot be read. |
| 0    | 1     | 80C1H  | The specified data record is currently being processed. |
| 0    | 1     | 80C2H  | There are too many jobs pending. |
| 0    | 1     | 80C3H  | Resources occupied (memory). |
| 0    | 1     | 80C4H  | Communication error (occurs temporarily, it is usually best to repeat the job in the user program). |
| 0    | 1     | 80D2H  | Module start address incorrect. |
| 0    | 1     | 8183H  | The configuration does not match the job parameters. |
| 0    | 1     | 8184H  | Bad data type specified for the FILE_NAME / LOGIN parameter. |
| 0    | 1     | 8186H  | ID parameter invalid. ID != 1,2,...64. |
| 0    | 1     | 8F22H  | Source area invalid, for example:  
|      |       |        | Area does not exist in the DB |
| 0    | 1     | 8F24H  | Area error reading a parameter. |
| 0    | 1     | 8F28H  | Alignment error reading a parameter. |
| 0    | 1     | 8F32H  | Parameter contains a DB number that is too high. |
| 0    | 1     | 8F33H  | DB number error. |
| 0    | 1     | 8F3AH  | Area not loaded (DB). |
| 0    | 1     | 8F50H  | File DB DB 0 or DB does not exist |
| 0    | 1     | 8F51H  | Specified file DB data area larger than existing data area |
| 0    | 1     | 8F52H  | File DB in write-protected memory |
| 0    | 1     | 8F53H  | File DB max. length < current length |
| 0    | 1     | 8F54H  | File DB does not contain any valid data |
| 0    | 1     | 8F55H  | Header status bit: Locked |
| 0    | 1     | 8F56H  | The NEW bit in the file DB header was not reset |
| 0    | 1     | 8F57H  | The FTP client does not have write access to the file DB but rather the FTP server (header status bit: WriteAccess) |
| 0    | 1     | 8F5AH  | Buffer DB DB 0 or DB does not exist |
| 0    | 1     | 8F5BH  | Buffer DB data area too short |
2.4.10 Data block file DB

2.4.10.1 Structure of the data blocks (file DBs) for FTP services - FTP client mode

Procedure

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)
File DB header for FTP client mode

Note: The file DB header described here is largely identical to the file DB header for server mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value / meaning</th>
<th>Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIST</td>
<td>BOOL</td>
<td>The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1.</td>
<td>The dele FTP command sets EXIST=0; The stor FTP command sets EXIST=1;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: The file DB does not contain valid user data (&quot;file does not exist&quot;).</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: The file DB contains valid user data (&quot;file exists&quot;).</td>
<td></td>
</tr>
<tr>
<td>LOCKED</td>
<td>BOOL</td>
<td>The LOCKED bit is used to restrict access to the file DB.</td>
<td>The stor and retr FTP commands set LOCKED=1 when they are executed. The following function is also possible when writing from the user program: The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: The file DB can be accessed.</td>
<td>1. Check LOCKED bit; if = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: The file DB is locked.</td>
<td>2. Set WRITEACCESS bit = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Check LOCKED bit; if = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4. Set LOCKED bit = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5. Write data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6. Set LOCKED bit = 0</td>
</tr>
</tbody>
</table>
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value / meaning</th>
<th>Supply</th>
</tr>
</thead>
</table>
| NEW                | BOOL         | The NEW bit indicates whether data has been modified since the last read access.  
  - 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification.  
  - 1: The user program of the S7 CPU has not yet registered the last write access. | After execution, the stor FTP command sets NEW=1  
  After reading the data, the user program in the S7-CPU must set NEW=0 to allow a new retr command.                                                                                     |
| WRITE_ACCESS       | BOOL         | 0: The user program (FTP client blocks) has write access rights for the file DBs on the S7 CPU.  
  1: The user program (FTP client blocks) has no write access rights for the file DBs on the S7 CPU.                                                                                     | During the configuration of the DB, the bit is set to an initialization value.  
  Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.                                                   |
| ACT_LENGTH         | DINT         | Current length of the user data area. The content of this field is only valid when EXIST = 1.                                                                                                                   | The current length is updated following write access.                                                                                         |
| MAX_LENGTH         | DINT         | Maximum length of the user data area (length of the entire DB less 20 bytes header).                                                                                                                          | The maximum length should be specified during configuration of the DB.  
  The value can also be modified by the user program during operation.                                                                             |
| FTP_REPLY_CODE     | INT          | Unsigned integer (16-bit) containing the last reply code from FTP as a binary value. The content of this field is only valid when EXIST = 1.                                                                     | This is updated by the FTP client when the FTP command is executed.                                                                            |
| DATE_TIME          | DATE_AND_TIME| Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.                                                                                                   | The current date is updated following a write access.  
  If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on.  
  If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the IT-CP (the initialization value is 1.1.1994 0.0 (midnight)). |
2.4.10.2 Structure of the data blocks (file DBs) for FTP services - FTP server mode

Procedure

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length (20 bytes) and structure)
- Section 2: User data (has a variable length and structure)

File DB header for FTP server mode

Note: The file DB header described here is largely identical to the file DB header for client mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value / meaning</th>
<th>Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIST</td>
<td>BOOL</td>
<td>The EXIST bit indicates whether the user data area contains valid data.</td>
<td>The dele FTP command sets EXIST=0; The store FTP command sets EXIST=1;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The retrieve FTP command executes the job only when EXIST=1.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: The file DB does not contain valid user data (&quot;file does not exist&quot;).</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: The file DB contains valid user data (&quot;file exists&quot;).</td>
<td></td>
</tr>
</tbody>
</table>
### Parameter blocks for FTP services

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value / meaning</th>
<th>Supply</th>
</tr>
</thead>
</table>
| LOCKED          | BOOL  | The LOCKED bit is used to restrict access to the file DB.  
• 0: The file DB can be accessed.  
• 1: The file DB is locked.                                                                                                                                  | The stor and retr FTP commands set LOCKED=1 when they are executed.  
The following function is also possible when writing from the user program:  
The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency.  
Recommended sequence in the user program:  
1. Check LOCKED bit;  
   if = 0  
2. Set WRITEACCESS bit = 0  
3. Check LOCKED bit;  
   if = 0  
4. Set LOCKED bit = 1  
5. Write data  
6. Set LOCKED bit = 0                                                                                                                                      |
| NEW             | BOOL  | The NEW bit indicates whether data has been modified since the last read access.  
• 0: The content of the file DB is unchanged since the last write access.  
The user program of the S7 CPU has registered the last modification.  
• 1: The user program of the S7 CPU has not yet registered the last write access.                                                             | After execution, the stor FTP command sets NEW=1  
After reading the data, the user program on the S7-CPU must set NEW=0 to allow store to be used again or to be able to delete the file with the dele FTP command. |
| WRITE_ACCESS    | BOOL  | 0: The FTP client on the PG/PC has no write access rights for the file DBs on the S7 CPU.  
1: The FTP client on the PG/PC has write access rights for the file DBs on the S7 CPU.                                                                      | During the configuration of the DB, the bit is set to an initialization value.  
Recommendation:  
Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible. |
| ACT_LENGTH      | DINT  | Current length of the user data area.  
The content of this field is only valid when EXIST = 1.                                                                                                     | The current length is updated following write access.                                                                                                                  |
Program blocks for Industrial Ethernet

2.5 Program blocks for programmed connections and IP configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value / meaning</th>
<th>Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_LENGTH</td>
<td>DINT</td>
<td>Maximum length of the user data area (length of the entire DB less 20 bytes header).</td>
<td>The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.</td>
</tr>
<tr>
<td>FTP_REPLY_CODE</td>
<td>INT</td>
<td>This parameter is irrelevant in FTP server mode.</td>
<td>Is set to &quot;0&quot; by the FTP server.</td>
</tr>
<tr>
<td>DATE_TIME</td>
<td>DATE_AND_TIME</td>
<td>Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.</td>
<td>The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the IT-CP (the initialization value is 1.1.1994 0.0 (midnight)).</td>
</tr>
</tbody>
</table>

2.5 Program blocks for programmed connections and IP configuration

In certain areas of application, it is an advantage to set up the communications connections or to create the IP configuration program-controlled in a special application instead of via the configuration interface of STEP 7.

Typical users who will find this useful are, for example, mass producers of machines who want to offer their customers a simple user interface but need to adapt the communication services to the operator input. The end user should not need knowledge of STEP 7.

For such applications, function block FB55 is available for setting up connections on the SEND/RECEIVE interface and for IP configuration. FB55 allows the flexible transfer of data blocks with configuration data to an Ethernet CP.

Note

Remember that the functions described here depend on the characteristics (supported connection types) of the CP type you are using. You will find information on this in the manuals.

Further information

You will find further information on the following topics in /1/ (Page 281):

- Properties of the configurable connection types;
- Information on configuring IP access protection;
- Information on amounts of data and configuration limits.
2.5.1 Operating principle

Interplay between programming and configuration

You configure connections on the SEND/RECEIVE interface and the IP configuration of a CP either with STEP 7 or you configure them during runtime of the S7 station via the user program. Mixing these variants on a CP is not possible!

Principle of programmed configuration

Configuration data for communication connections and the IP configuration can be transferred to the CPU using the function block called in the user program.

![Configuration data block](image)

The configuration DB can be loaded on the CP at any time. The previously applicable connections and configuration data (IP address, subnet mask, default router, NTP time server and other parameters) are overwritten.

Based on the configuration data, the Ethernet CP recognizes that the communication connections must be set up by the user program.

**Note**

The functions can only be executed if "Not locked" was configured for the module access protection. Refer to the "Options" tab in the properties dialog of the CP (not available for every CP).

The "Set IP address in user program" option must also be enabled (see properties dialog of the CP or the Ethernet interface of the CP, "IP Configuration" tab).

Based on the configuration data, the Ethernet CP recognizes that the communication connections must be set up by the user program.

**Note**

As soon as the user program transfers the connection data via FB55 IP_CONFIG, the CPU switches the CP briefly to STOP. The CP receives the system data (including the IP address) and the new connection data and processes them during startup (RUN).
Quantity framework

A maximum of 64 connections can be specified in program block CP_CONFIG. The most important factor, however, is the maximum number of connections supported by the CP type you are using.

Special features / restrictions

- Consistency check only with STEP 7
  The connection configuration in STEP 7 involves consistency checks that are not possible or only possible with restrictions when using the programmed configuration!

- Connection configuration required on the partner
  When configuring specified connections in STEP 7, you implicitly create the connection for the partner; with a programmed configuration, this is not possible! In this case, you must configure suitable connections for the partner.

- Configuring IP access protection
  Using IP access protection gives you the opportunity of restricting communication over the CP of the local S7 station to partners with specific IP addresses. This parameter assignment also applies to programmed communications connections. You either disable IP access protection in STEP 7 (= default) or authorize the communications partner.

- DHCP / DNS is supported
  With a programmed configuration, IP addressing is also possible using DHCP (and DNS for the mail service).
  The use of a DHCP server is defined in this case in FB55 (not in the configuration).

- No connection information when uploading
  When you upload the S7 station data in STEP 7, this does not contain the data of the programmed configuration.

- Configuring connections for CPs with several interfaces
  If you are using CPs with several interfaces (for example with a gigabit interface), check the device manual to see whether or not the connection configuration is supported for both interfaces.

- PROFINET IO is not possible at the same time
  On a device you intend to operate as a PROFINET IO controller or IO device, it is not possible to set up the connection using FB55 as described here.

- No use of IP_CONFIG when operating the CP with fault-tolerant S7 connections
  If you configure fault-tolerant S7 connections via the CP, you cannot use the IP_CONFIG program block for IP configuration of the CP.
2.5.2 Procedure

Initial situation
The steps described here assume the following:

- You have created the local S7 station and the required partner stations in your STEP7 projects.
- You have clarified the other station types with which connections must be established. You create substitute objects for these station types in your STEP 7 projects.

Configuring CP properties
Set the CP properties in "IP Configuration": Select the "Set IP address in user program" option.

Programming connection setup
The basic procedure for setting up connections via the user program is as follows:
1. Create the subfields for system and connection data in the configuration DB.
2. Set the connection properties in the configuration DB.
3. Program the FB55 interface in the user program.
4. Use the FCs of the SEND/RECEIVE interface for open communications services in the user program.

2.5.3 Configuration data block (CONF_DB)

Meaning
The configuration data block (CONF_DB) contains all the connection data and configuration data (IP address, subnet mask, default router, NTP time server and other parameters) for an Ethernet CP. The configuration data block is transferred to the CP with function block FB55.
Block and Data Structure

The graphic below shows the following:

- Structure resulting from parameter fields and subfields
  - The connections and specific system data are described by an identically structured parameter field.
  - Individual parameters are characterized by subfields.

- Offset range
  The CONF_DB can start at any point within a data block as specified by an offset range. The address (or offset) must simply be an even number.

---

1. Parameter fields are described below in Parameter field for system data (IP configuration) (Page 112)
2. Subfield types are described below in Parameter fields for connection types (Page 113)

See also

Subfield types (Page 119)

---

2.5.4 Configuration data block - example

Below, you will find a sample of a configuration data block with the parameter field for system data and a parameter field for a TCP connection.
CONF_DB

STL

DATA_BLOCK DB271
TITLE=IP_CONFIG for 1 active nTCP connection,
AUTHOR : SIMATIC //CP-Daten : IP=200.12.1.144, Router=200.12.1.80
FAMILY : A3300 //Connection data: Destination IP-Addr=200.12.1.99,
NAME : ipconf //Local port = 4001, remote port = 5001, estab=active
VERSION : 1.0 //07-June-2005

STRUCT
  DB_TYP : INT := 1;
  // ----------------------------------------------------------- System Data ------
  sys_pb : INT:=0; // Subfield type: System data for CP
  sys_id : INT:= 0; // System parameter ID, always 0
  sys_sb_cnt: INT:= 3; // Number of subfields in the system parameter field
  ip_addr: SUB_IP_V4; // IP address of the CP
  ip_netmask: SUB_NETMASK; // Subnet mask of the CP
  ip_router: SUB_DEF_ROUTER; // Default router

  // ----------------------------------------------------------- tcp VB 01 --------
  tcp_pb_01 : int := 1; // Subfield type: TCP connection
  tcp_id_01 : int := 1; // 1. TCP_VB
  tcp_sb_cnt_01 : int := 6; // 6 elements per TCP connection
  tcp_vb_ip_01 : SUB_IP_V4; // IP address of the partner
  tcp_loc_01 : SUB_LOC_PORT; //
  tcp_rem_01 : SUB_REM_PORT; //
  tcp_vb_01_name : CON_NAME_L; //
  tcp_vb_01_kbus : SUB_KBUS_ADDR; // only relevant for S7-400
  rq_01 : ACT_CN_REQ; //

  //--------------------------------------------------------------
END_STRUCT ;
BEGIN
  tcp_loc_01.port := 4001; // Definition of the ports, when the value of
  tcp_rem_01.port := 5001; // of the predefinition is different !
END_DATA_BLOCK

The type definitions used in the sample DB follow.
Type definitions for the "system data" subfield

```
// Data structure IP Config
TYPE "SUB_IP_V4"
STRUCT
  id : int := 1; // ID for IP, V4-Addr.
  len: int := 8; // Sub Block Length
  b_3 : BYTE := b#16#C8; // IP_High 200.
  b_2 : BYTE := b#16#0C; // IP_12.
  b_1 : BYTE := b#16#01; // IP_1.
  b_0 : BYTE := b#16#90; // IP_Low 144
END_STRUCT;
END_TYPE

TYPE "SUB_NETMASK"
STRUCT
  id : int := 2; // ID for Sub Net Mask
  len: int := 8; // Sub Block Length
  b_3 : BYTE := b#16#FF; // SNM_High
  b_2 : BYTE := b#16#FF; // SNM_
  b_1 : BYTE := b#16#FF; // SNM_
  b_0 : BYTE := b#16#00; // SNM_Low
END_STRUCT;
END_TYPE

TYPE "SUB_DEF_ROUTER"
STRUCT
  id : int := 8; // ID_4_Router
  len: int := 8; // Sub Block Length
  r_3 : BYTE := b#16#C8; // R_High
  r_2 : BYTE := b#16#0C; // R_
  r_1 : BYTE := b#16#01; // R_
  r_0 : BYTE := b#16#50; // R_Low
END_STRUCT;
END_TYPE
```
Type definitions for the subfield "TCP connection"

STL

TYPE "SUB_LOC_PORT"
  STRUCT
    id : int := 9; // ID_4_LOC_PORT
    len: int := 6; // Sub Block Length
    port: int := 2001; // Loc. Port
  END_STRUCT;
END_TYPE

TYPE "SUB_REM_PORT"
  STRUCT
    id : int := 10; // ID_4_REM_PORT
    len: int := 6; // Sub Block Length
    port: int := 2002; // Rem. Port
  END_STRUCT;
END_TYPE

TYPE "CON_NAME_L" // 24 characters
  STRUCT
    id : int := 18; // ID for CON Name
    len: int := 28; // 4+len(n[0..x]
    c : ARRAY [1..24] of CHAR := 'V','B','_','N','a','m','e','_','2','4','C','h','a','r','a','c','t','e','r','s','_','0','0','1';
  END_STRUCT;
END_TYPE

TYPE "SUB_KBUS_ADDR"
  STRUCT
    id : int := 21; // ID for KBUS-Address
    len: int := 5; //
    addr: BYTE := B#16#04; // =R0/S4
  END_STRUCT
END_TYPE

TYPE "ACT_CN_REQ"
  STRUCT
    id : int := 22; // ID for CON REQ Mode
    len: int := 5; // Sub Block Length
    w : BYTE := b#16#1; // = Active
  END_STRUCT;

Note:
The structures listed here must also be entered in the symbol table.
Example of SUB_IP_V4 entry:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB_IP_V4</td>
<td>UDT 100</td>
<td>UDT 100</td>
</tr>
</tbody>
</table>

See also

Subfield types (Page 119)

2.5.5 Parameter field for system data (IP configuration)

Meaning

Below, you will find the parameter field for system data relevant to the IP configuration of the CP and the subfields that need to be specified in it. Some applications do not require all the subfield types - refer to the table for details.

Layout

On CPs with several interfaces, the structure of the parameter field described below applies only to the PROFINET interface.

- Type = 0
- ID = 0
- Number of subfields = n
- Subfield 1
- Subfield 2
- Subfield n
...

Usable subfields

<table>
<thead>
<tr>
<th>Subfield ID</th>
<th>Parameter</th>
<th>Special features / notes</th>
<th>Application ***</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SUB_IP_V4</td>
<td>Local IP address</td>
<td>++</td>
</tr>
<tr>
<td>2</td>
<td>SUBNET_MASK</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>8</td>
<td>SUB_DEF_ROUTER</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>SUB_DNS_SERV_ADDR</td>
<td>This subfield can occur from 0 to 4 times. The first entry is the primary DNS server.</td>
<td>+</td>
</tr>
<tr>
<td>14</td>
<td>SUB_DHCP_ENABLE</td>
<td>0: No DHCP 1: DHCP</td>
<td>+</td>
</tr>
</tbody>
</table>
2.5 Program blocks for programmed connections and IP configuration

| Subfield          | Parameter                          | Special features / notes | Application (***)
|-------------------|------------------------------------|--------------------------|-----------------
| ID 15             | SUB_CLIENT_ID                      | -                        | +               
|                   | Note: Only useful when SUB_DHCP_ENABLE = 1 |                          |                 
| ID 30**)          | SUBDEVICE_NAME                     | Device name complying with PROFINET IO convention Enter a device name to make the device individually recognizable for analysis and diagnostics in the network. | +               

*) The subfield type is used only for E-mail connections.
**) ID is supported only by certain CP types.
***) ++ = mandatory; + = optional

See also

Subfield types (Page 119)

2.5.6 Parameter fields for connection types

General

Below, you will see which values need to be entered in the parameter fields and which subfields are used for the various connection types.

Some applications do not require all the subfield types - refer once again to the table for details.

Connection ID

The ID parameter that precedes each connection parameter field beside the type ID is particularly important.

On programmed connections, you can assign this ID freely within the permitted range of values. You must then use this ID on the call interface of the FCs for the SEND/RECV interface to identify the connection.

Range of values for the connection ID:

- S7-400: 1,2...64
- S7-300: 1,2...16
2.5.6.1 Parameter field for TCP connection

Layout

Enter the parameters in the parameter field for TCP connections as follows:

- Type = 1
- ID = connection ID
- Number of subfields = n
- Subfield 1
- Subfield 2
- Subfield n

Legend:
1 Identifier for the connection type
2 Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.

Range of values for the connection ID:
- for S7-400: 1, 2...64
- for S7-300: 1, 2...16

Usable subfields

<table>
<thead>
<tr>
<th>Subfield</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 SUB_IP_V4</td>
<td>IP address of the partner ++ *)</td>
</tr>
<tr>
<td>9 SUB_LOC_PORT</td>
<td>- ++</td>
</tr>
<tr>
<td>10 SUB_REM_PORT</td>
<td>- ++ **)</td>
</tr>
<tr>
<td>18 SUB_CONNECT_NAME</td>
<td>- +</td>
</tr>
<tr>
<td>19 SUB_LOC_MODE</td>
<td>- +</td>
</tr>
<tr>
<td>21 SUB_KBUS_ADR</td>
<td>This value is always set to 2 for CPs for the S7300 and does not need to be specified. ++ (for S7-400)</td>
</tr>
<tr>
<td>22 SUB_CON_ESTABL</td>
<td>- ++</td>
</tr>
</tbody>
</table>

*) optional for a passive connection.
***) ++ = mandatory; + = optional

See also

Subfield types (Page 119)
2.5.6.2 Parameter field for UDP connection

Layout

Enter the parameters in the parameter field for UDP connections as follows:

- **Type = 2**
- **ID = connection ID**
- Number of subfields = \( n \)
- Subfield 1
- Subfield 2
- Subfield \( n \)

Legend:

1. Identifier for the connection type
2. Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.

Range of values for the connection ID:
- for S7-400: 1, 2...64
- for S7-300: 1,2...16

Usable subfields

<table>
<thead>
<tr>
<th>Subfield</th>
<th>Parameter</th>
<th>Special features / notes</th>
<th>Application ***)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SUB_IP_V4</td>
<td>IP address of the partner</td>
<td>++</td>
</tr>
<tr>
<td>9</td>
<td>SUB_LOC_PORT</td>
<td></td>
<td>++</td>
</tr>
<tr>
<td>10</td>
<td>SUB_REM_PORT</td>
<td></td>
<td>++</td>
</tr>
<tr>
<td>18</td>
<td>SUB_CONNECT_NAME</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>19</td>
<td>SUB_LOC_MODE</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>21</td>
<td>SUB_KBUS_ADR</td>
<td>This value is always set to 2 for CPs for the S7300 and does not need to be specified.</td>
<td>++ (for S7-400)</td>
</tr>
<tr>
<td>23</td>
<td>SUB_ADDR_IN_DATABLOCK</td>
<td>If the &quot;Free UDP connection&quot; is selected for this parameter, the parameters SUB_IP_V4 and SUB_REM_PORT are omitted.</td>
<td>+</td>
</tr>
</tbody>
</table>

***) ++ = mandatory; + = optional

See also

Subfield types (Page 119)
2.5 Program blocks for programmed connections and IP configuration

2.5.6.3 Parameter field for an ISOonTCP connection

Layout

Enter the parameters in the parameter field for ISO-on-TCP connections as follows:

- **Type** = 3 -> ①
- **ID** = connection ID -> ②
- Number of subfields = n
- Subfield 1
- Subfield 2
- Subfield n
...  

Legend:
① Identifier for the connection type
② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.

Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

<table>
<thead>
<tr>
<th>Subfield ID</th>
<th>Type</th>
<th>Parameter</th>
<th>Special features / notes</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SUB_IP_V4</td>
<td>IP address of the partner</td>
<td></td>
<td>++ *)</td>
</tr>
<tr>
<td>11</td>
<td>SUB_LOC_TSAP</td>
<td>-</td>
<td></td>
<td>++</td>
</tr>
<tr>
<td>12</td>
<td>SUB_REM_TSAP</td>
<td>-</td>
<td></td>
<td>++ *)</td>
</tr>
<tr>
<td>18</td>
<td>SUB_CONNECT_NAME</td>
<td>-</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>19</td>
<td>SUB_LOC_MODE</td>
<td>-</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>21</td>
<td>SUB_KBUS_ADR</td>
<td>This value is always set to 2 for CPs for the S7300 and does not need to be specified.</td>
<td></td>
<td>++ (for S7-400)</td>
</tr>
<tr>
<td>22</td>
<td>SUB_CON_ESTABL</td>
<td>-</td>
<td></td>
<td>++</td>
</tr>
</tbody>
</table>

*) optional on the passive connection (if the IP address is specified, the TSAP must also be specified)
***) ++ = mandatory; + = optional

See also

Subfield types (Page 119)
2.5.6.4 Parameter field for an Email connection

Meaning
To send Emails, one Email connection must be set up per Advanced CP. The Email connection specifies the mail server via which all the mails sent by the Advanced CP are delivered.

Layout
Enter the parameters in the parameter field for E-mail connections as follows:

- **Type = 4**
- **ID = connection ID**
- Number of subfields = \( n \)

<table>
<thead>
<tr>
<th>Subfield</th>
<th>Parameter</th>
<th>Special features / notes</th>
<th>Application ***)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SUB_IP_V4</td>
<td>IP address of the mail server, over which the Emails are sent.</td>
<td>++ / + ¹</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can specify an absolute or alias IP address.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The use of an alias assumes that the Advanced CP knows the address of the domain name server (DNS). This entry must be made when configuring the Advanced CP in STEP 7. For more detailed information refer to the online help.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SUB_DNS_NAME</td>
<td>DNS name of the Email server</td>
<td>++ / + ¹</td>
</tr>
<tr>
<td>13</td>
<td>SUB_EMAIL_SENDER</td>
<td>Email address of the sender</td>
<td>++</td>
</tr>
<tr>
<td>18</td>
<td>SUB_CONNECT_NAME</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

Legend:
① Identifier for the connection type
② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.

Range of values for the connection ID:
- for S7-400: 1, 2...64
- for S7-300: 1,2...16

Usable subfields
### Parameter field for FTP connection

#### Meaning

To run an FTP job sequence between the S7 station acting as the FTP client and an FTP server, the Advanced CP must establish a connection to the S7 CPU. This connection is known as an FTP connection.

FTP connections are TCP connections, with the parameter SUB_LOC_MODE set to the "FTP" mode.

#### Layout

Enter the parameters in the parameter field for FTP connections as follows:

- **Type** = 1
- **ID** = connection ID
- Number of subfields = n
- Subfield 1
- Subfield 2
- Subfield n

... 

**Legend:**

1. Identifier for the connection type
2. Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.

**Range of values for the connection ID:**

- for S7-400: 1, 2...64
- for S7-300: 1, 2...16

---

Note

Mail server ports are "well-known ports" and do not need to be specified.

---

See also

Subfield types (Page 119)

---

**2.5.6.5 Parameter field for FTP connection**
Usable subfields

<table>
<thead>
<tr>
<th>Subfield</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID Type</td>
</tr>
</tbody>
</table>
| ID | Type | Special features / notes | Application (***)
| 18 | SUB_CONNECT_NAME | - | +
| 19 | SUB_LOC_MODE | here: 0x01 = FTP protocol | ++
| 21 | SUB_KBUS_ADR | This value is always set to 0 for CPs for the S7300 and does not need to be specified. | ++ (for S7-400)

***) ++ = mandatory; + = optional

See also

Subfield types (Page 119)

2.5.7 Subfield types

Different parameters are required depending on the parameter field. Each parameter is described by a subfield. Which subfields are required is explained in the descriptions of the system data and the connection types in the previous sections.

Each subfield consists of the specific parameter section and the header (4 Byte).

Example

The following excerpt from a CONF_DB illustrates the structure of a subfield based on the example of the SUBNET_MASK subfield type.

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th>Initial value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>+14.0</td>
<td>Sub_field_2</td>
<td>STRUCT</td>
<td></td>
<td>Subfield 2 type SUBNET_MASK</td>
</tr>
<tr>
<td>+0.0</td>
<td>Sub_field_ID</td>
<td>INT</td>
<td>2</td>
<td>Subfield ID</td>
</tr>
<tr>
<td>+2.0</td>
<td>Sub_field_len</td>
<td>INT</td>
<td>8</td>
<td>Total length of the subfield in bytes</td>
</tr>
<tr>
<td>+4.0</td>
<td>Parameter</td>
<td>STRUCT</td>
<td></td>
<td>Parameter range of SUBNET_MASK</td>
</tr>
<tr>
<td>+0.0</td>
<td>Value_1</td>
<td>BYTE</td>
<td>B#16#FF</td>
<td></td>
</tr>
<tr>
<td>+1.0</td>
<td>Value_2</td>
<td>BYTE</td>
<td>B#16#FF</td>
<td></td>
</tr>
<tr>
<td>+2.0</td>
<td>Value_3</td>
<td>BYTE</td>
<td>B#16#FF</td>
<td></td>
</tr>
<tr>
<td>+3.0</td>
<td>Value_4</td>
<td>BYTE</td>
<td>B#16#0</td>
<td></td>
</tr>
<tr>
<td>=4.0</td>
<td></td>
<td>END_STRUCT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>=8.0</td>
<td></td>
<td>END_STRUCT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In total, the following subfield types are available:

<table>
<thead>
<tr>
<th>Subfield ID</th>
<th>Subfield type</th>
<th>Subfield length (in bytes)</th>
<th>Meaning of the Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SUB_IP_V4</td>
<td>4 + 4</td>
<td>IP address according to IPv4</td>
</tr>
<tr>
<td>2</td>
<td>SUBNET_MASK</td>
<td>4 + 4</td>
<td>Subnet mask</td>
</tr>
<tr>
<td>3</td>
<td>SUB_DNS_NAME</td>
<td>Length of DNS name + 4</td>
<td>DNS name</td>
</tr>
<tr>
<td>4</td>
<td>SUB_DNS_SERV_ADDR</td>
<td>4 + 4</td>
<td>DNS server address</td>
</tr>
<tr>
<td>8</td>
<td>SUB_DEF_ROUTER</td>
<td>4 + 4</td>
<td>IP address of default router</td>
</tr>
<tr>
<td>9</td>
<td>SUB_LOC_PORT</td>
<td>2 + 4</td>
<td>Local port</td>
</tr>
<tr>
<td>10</td>
<td>SUB_REM_PORT</td>
<td>2 + 4</td>
<td>Remote port, also for Email connections</td>
</tr>
<tr>
<td>11</td>
<td>SUB_LOC_TSAP</td>
<td>TSAP length + 4</td>
<td>Local TSAP *</td>
</tr>
<tr>
<td>12</td>
<td>SUB_REM_TSAP</td>
<td>TSAP length + 4</td>
<td>Remote TSAP *</td>
</tr>
<tr>
<td>13</td>
<td>SUB_EMAIL_SENDER</td>
<td>Length of the sender E-mail address + 4</td>
<td>Email address of the sender</td>
</tr>
<tr>
<td>14</td>
<td>SUB_DHCP_ENABLE</td>
<td>2 + 4</td>
<td>Obtain an IP address from a DHCP server</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Range of values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 = no DHCP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = DHCP (optional)</td>
</tr>
<tr>
<td>15</td>
<td>SUB_CLIENT_ID</td>
<td>Length of the client ID + 4</td>
<td>(optional)</td>
</tr>
<tr>
<td>18</td>
<td>SUB_CONNECT_NAME</td>
<td>Length of the name + 4</td>
<td>Name of the connection Possible characters are: a...z, A...Z, 0...9, -, _</td>
</tr>
<tr>
<td>19</td>
<td>SUB_LOC_MODE</td>
<td>1 + 4</td>
<td>Local mode of the connection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Range of values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00 = SEND/RECV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x01 = FTP protocol (TCP connection only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x10 = S5 addressing mode for FETCH/WRITE *)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x20 = SPEED SEND/RECV (permitted only for CP 443-1 Advanced)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x80 = FETCH *)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x40 = WRITE *)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If you do not set the parameter, the default setting is SEND/RECV.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: FETCH / WRITE require the passive connection establishment setting (see SUB_CON_ESTABL).</td>
</tr>
<tr>
<td>Subfield ID 1)</td>
<td>Subfield type</td>
<td>Subfield length (in bytes)</td>
<td>Meaning of the Parameter</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------</td>
<td>----------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>20</td>
<td>SUB_REM_MODE</td>
<td>1 + 4</td>
<td>Setting the mode on the communication partner. (not currently supported)</td>
</tr>
<tr>
<td>21</td>
<td>SUB_KBUS_ADR</td>
<td>5</td>
<td>• For S7-400</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• For S7-300</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• To be entered as a fixed value for the slot address: 2</td>
</tr>
<tr>
<td>22</td>
<td>SUB_CON_ESTABL</td>
<td>1 + 4</td>
<td>Type of connection establishment. With this option, you specify whether connection establishment from this S7 station is active or passive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Range of values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 = passive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = active</td>
</tr>
<tr>
<td>23</td>
<td>SUB_ADDR_IN_DATA-BLOCK</td>
<td>1 + 4</td>
<td>Select free UDP connection. The remote node is entered in the job header of the job buffer by the user program when it calls AG_SEND. This allows any node on Ethernet/LAN/WAN to be reached.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Range of values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = free UDP connection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 = other</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The parameter is practical only for a UDP connection.</td>
</tr>
<tr>
<td>24</td>
<td>SUB_NTP_SERVER</td>
<td>4 + 4</td>
<td>The subfield defines an NTP server from which the CP can obtain its time via the NTP protocol. For the situation when one or more NTP servers are defined, up to 4 subfields of ID 24 can be defined. The subfields of ID 24 may only be installed in the system parameter field type 0 / ID 0.</td>
</tr>
</tbody>
</table>
### Subfield (ID 1)

<table>
<thead>
<tr>
<th>Subfield type</th>
<th>Subfield length (in bytes)</th>
<th>Meaning of the Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB_DEVICE_NAME</td>
<td>Length of the name + 4</td>
<td>Device name complying with PROFINET IO convention</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The device name must comply with DNS conventions, in other words:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Restriction to a total of 127 characters (letters, numbers, hyphen or period)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Parts of the name within the device name; in other words, a string between two periods, must not exceed a maximum of 63 characters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- No special characters such as umlauts (ä, ö etc.), brackets, underscore, slash, blank etc. The dash (hyphen) is the only permitted special character.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The device name must not begin or end with the &quot;.&quot; or &quot;.&quot; character, nor may either of these be the last character.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The device name must not begin with numbers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The device name must not have the format n.n.n.n (n = 0...999).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The device name must not begin with the character string &quot;port-xyz-&quot; (x, y, z = 0...9).</td>
</tr>
</tbody>
</table>

1) Note: ID numbers not listed are not currently used.

* For subfield 11 and 12: If the subblock length is an uneven number of bytes, a padding byte that has no further use is inserted after the subblock so that there is an even byte address for the next subblock. The padding byte is not displayed in the subblock length, but must be taken into account in the total length of the data block.

**See also**

Configuration data block - example (Page 108)

### 2.5.8 IP_CONFIG - meaning and call

**Meaning of the block**

FB55 transfers the IP configuration specified in a data block (configuration DB) and the connection data to the CP. The configuration DB contains all the connection data to allow the connections for the SEND/RECEIVE interface of an Ethernet CP to be set up.

You can use this variant of programmed communication connections as an alternative to connection configuration with STEP 7.
Depending on the size of the configuration DB, the data may be transferred to the CP in several segments. This means that the FB must continue to be called until the FB signals complete transfer by setting the DONE bit to 1.

**Call**

Call interface in FBD representation

![FBD interface diagram]

**Example in STL representation**

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call fb 55 {</td>
<td>//IP_CONFIG block call</td>
</tr>
<tr>
<td>ACT := M 10.0,</td>
<td>//Job initiated by memory bit</td>
</tr>
<tr>
<td>LADDR := W#16#0100,</td>
<td>//LADDR 256 dec. in hardware configuration</td>
</tr>
<tr>
<td>CONF_DB := P#db99.dbx10.0 byte 240,</td>
<td>//Data block with connection data</td>
</tr>
<tr>
<td>LEN := MW 14,</td>
<td>//Length info for the connection data</td>
</tr>
<tr>
<td>DONE := M 10.1,</td>
<td>//Execution code</td>
</tr>
<tr>
<td>ERROR := M 10.2,</td>
<td>//Error code</td>
</tr>
<tr>
<td>STATUS := MW 16,</td>
<td>//Status code</td>
</tr>
<tr>
<td>EXT_STATUS := MW 18);</td>
<td>//Cause of error in connection data</td>
</tr>
</tbody>
</table>

**Note**

**No use of FB55 when operating the CP with fault-tolerant S7 connections**

If you configure fault-tolerant S7 connections via the CP, you cannot use FB55 for IP configuration of the CP.

**Note**

**Avoid possible double addressing**

If you use FB55, make sure that the assignment of IP addresses is unique. If an address is detected twice, it is possible that the CP will not become active in the network.
2.5 Program blocks for programmed connections and IP configuration

2.5.9 How IP_CONFIG works

Handling in the user program

When using FB55, the following use cases must be distinguished:

- Standard application
  call FB55 in the startup OB (OB100). When OB1 starts, the CP then already has its IP configuration and possibly also its connection configuration.

- Use in fault-tolerant systems (H systems)
  The procedure recommended below allows CPs in the redundant system to be configured with FB55 if there is a redundancy failover in the H system.

**Note**

**CPU in RUN mode**

In an H system, you can only configure a CP with FB55 if the assigned CPU is in RUN.

Initially, only one rack starts up in the H system. This means that the CPU can only reach its own peripherals (CP). As a result, you can only set parameters for the CPs in this rack with FB55 in the startup OB (OB100).

To be able to set parameters for the CPs in the redundant rack as well, the following procedure is recommended:

1. In OB100, program the FB55 calls for all CPs that are to be configured with FB55.

2. As the individual FB55 blocks execute, save the information as to whether or not configuration of the individual CPs was possible.

3. When the H system changes to the redundant state, OB72 (CPU redundancy error) is called automatically. Make sure that status information is saved while OB72 executes to indicate which configuration FBs (FB55) still need to be called.

4. Based on the previously saved status information, call the FB55 blocks in OB1 that you require for the IP configuration in the redundant system.

**Note:**

In principle, you can make those FB55 calls that were unsuccessful during startup in OB100 in OB72. Since, however, this is an FB that requires more than one call, this would extend the execution time of the OB. This is why the procedure in OB1 described above is recommended.

Operating principle

The following diagram illustrates the normal sequence of IP configuration and connection configuration triggered by IP_CONFIG in the user program.

The job executes as soon as the parameter ACT is transferred with value 1.

Due to the segmented transfer of the CONF_DB, you will need to repeat the job with ACT = 1 until completion of the job is indicated in the parameters DONE, ERROR, and STATUS.
If you want to transfer a connection configuration again later, the parameter ACT must first be transferred with value 0 in at least one further call.

**Note**

The data transferred with the configuration DB are stored in volatile memory on the CP and must be downloaded to the CP again following a power down!
### 2.5.10 Explanation of the formal parameters - IP_CONFIG

#### Explanation of the formal parameters

The following table explains the formal parameters for the call interface of the IP_CONFIG function block:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>INPUT</td>
<td>BOOL</td>
<td>0.1</td>
<td>When the FB is called with ACT = 1, the DBxx is sent to the CP. If the FB is called with ACT = 0, only the status codes DONE, ERROR and STATUS are updated.</td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.</td>
</tr>
<tr>
<td>CONF_DB</td>
<td>INPUT</td>
<td>ANY</td>
<td></td>
<td>The parameter points to the start address of the configuration data area in a data block (data type: byte).</td>
</tr>
<tr>
<td>LEN</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>Length information in bytes for the configuration data area.</td>
</tr>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Job completed with data transfer.</td>
<td>The parameter indicates whether the configuration data area was completely transferred. Remember that it may be necessary to call the FB several times depending on the size of the configuration data area (in several cycles) until the DONE parameter is set to 1 to signal completion of the transfer. For the meaning in conjunction with the parameters ERROR and STATUS, refer to IP_CONFIG status codes (Page 127)</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Error</td>
<td>Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to IP_CONFIG status codes (Page 127)</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td></td>
<td>Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to IP_CONFIG status codes (Page 127)</td>
</tr>
<tr>
<td>EXT_STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td></td>
<td>If an error occurs in the execution of a job, the parameter indicates which parameter was detected as the cause of the error in the configuration DB. High byte: Index of the parameter field Low byte: Index of the subfield within the parameter field</td>
</tr>
</tbody>
</table>
2.5.11 Reserved port numbers - IP_CONFIG

Reserved Port Numbers

The following local port numbers are reserved; do not use these in the connection project engineering.

Table 2-19 Reserved Port Numbers

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port number</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>20, 21</td>
<td>FTP</td>
</tr>
<tr>
<td>TCP</td>
<td>25</td>
<td>SMTP</td>
</tr>
<tr>
<td>TCP</td>
<td>80</td>
<td>HTTP</td>
</tr>
<tr>
<td>TCP</td>
<td>102</td>
<td>RFC1006</td>
</tr>
<tr>
<td>TCP</td>
<td>135</td>
<td>RPC-DCOM</td>
</tr>
<tr>
<td>HTTPS</td>
<td>443</td>
<td>Security With CPs with the Security function</td>
</tr>
<tr>
<td>TCP</td>
<td>502</td>
<td>ASA application protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>161</td>
<td>SNMP_REQUEST</td>
</tr>
<tr>
<td>UDP</td>
<td>34964</td>
<td>PN IO</td>
</tr>
<tr>
<td>UDP</td>
<td>65532</td>
<td>NTP</td>
</tr>
<tr>
<td>UDP</td>
<td>65533</td>
<td>NTP</td>
</tr>
<tr>
<td>UDP</td>
<td>65534</td>
<td>NTP</td>
</tr>
<tr>
<td>UDP</td>
<td>65535</td>
<td>NTP</td>
</tr>
</tbody>
</table>

2.5.12 IP_CONFIG status codes

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Table 2-20 Condition codes for FB55 IP_CONFIG

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0000H</td>
<td>Job completed without errors</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Job active</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80A4H</td>
<td>Communication error on the K-bus or Data error: Configuration by the user program is not set.</td>
</tr>
</tbody>
</table>
## Program blocks for Industrial Ethernet

### 2.5 Program blocks for programmed connections and IP configuration

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>80B1H</td>
<td>The number of data bytes to be sent exceeds the upper limit for this service. (upper limit = 16 Kbytes)</td>
</tr>
</tbody>
</table>
| 0    | 1     | 80C4H  | Communication error  
The error can occur temporarily; it is usually best to repeat the job in the user program. |
| 0    | 1     | 80D2H  | Configuration error  
The module you are using does not support this service. |

Errors detected in the evaluation of the FB in the CPU or on the interface between CPU and CP:

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>The CP rejects the requested data record number.</td>
</tr>
</tbody>
</table>
| 0    | 1     | 8184H  | System error or bad parameter type. (data type of the ANY pointer CONF_DB not OK)  
(Currently only the byte data type is accepted) |
| 0    | 1     | 8185H  | The value of the LEN parameter is larger than the CONF_DB less the reserved header (4 bytes) or the length information is incorrect. |
| 0    | 1     | 8186H  | Illegal parameter detected  
The ANY pointer CONF_DB does not point to a data block. |
| 0    | 1     | 8187H  | Illegal status of the FB  
Data in the header of CONF_DB was possibly overwritten. |

Further errors detected on the interface between the CPU and CP:

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8A01H</td>
<td>The status code in the data record is invalid (value is &gt;= 3).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A02H</td>
<td>There is no job running on the CP; the FB, however, expected an acknowledgment for a completed job.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A03H</td>
<td>There is no job running on the CP and the CP is not ready; the FB triggered the first job to read a data record.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A04H</td>
<td>There is no job running on the CP and the CP is not ready; the FB nevertheless expected an acknowledgment for a completed job.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A05H</td>
<td>There is a job running, but there was no acknowledgment; the FB nevertheless triggered the first job for a read data record job.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A06H</td>
<td>A job is complete but the FB nevertheless triggered the first job for a read data record job.</td>
</tr>
</tbody>
</table>

Errors detected when evaluating the FB on the CP:

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0    | 1     | 8B01H  | Communication error  
The DB could not be transferred. |
| 0    | 1     | 8B02H  | Parameter error  
Double parameter field |
| 0    | 1     | 8B03H  | Parameter error  
The subfield in the parameter field is not permitted. |
| 0    | 1     | 8B04H  | Parameter error  
The length specified in the FB does not match the length of the parameter fields / subfields. |
| 0    | 1     | 8B05H  | Parameter error  
The length of the parameter field is invalid. |
| 0    | 1     | 8B06H  | Parameter error  
The length of the subfield is invalid. |
| 0    | 1     | 8B07H  | Parameter error  
The ID of the parameter field is invalid |
<table>
<thead>
<tr>
<th>CODE</th>
<th>意义</th>
</tr>
</thead>
</table>
| 0 1 8B08H | 参数错误  
  该子字段的ID无效 |
| 0 1 8B09H | 系统错误  
  连接不存在 |
| 0 1 8B0AH | 数据错误  
  子字段的内容不正确 |
| 0 1 8B0BH | 结构错误  
  子字段存在两次 |
| 0 1 8B0CH | 数据错误  
  参数不包含所有必要的参数 |
| 0 1 8B0DH | 数据错误  
  CONF_DB不包含系统数据的参数字段 |
| 0 1 8B0EH | 数据错误/结构错误  
  CONF_DB类型无效 |
| 0 1 8B10H | 数据错误  
  用户程序的配置未设置 |
| 0 1 8B11H | 数据错误  
  指定的参数字段类型无效 |
| 0 1 8B12H | 数据错误  
  太多的连接被指定（要么是总和要么是特定类型的太多；例如，只能有一个电子邮件连接） |
| 0 1 8B13H | 内部错误  
  CP内部错误 |
| 0 1 8B14H | 数据错误  
  现行数据块的写保护参数 |
| 0 1 8B15H | 数据错误  
  写保护的第二当前数据块的参数 |
| 0 1 8B16H | 数据错误  
  参数包含的DB号太高 |
| 0 1 8B17H | DB编号错误  
  目标区域未加载 (DB) |
| 0 1 8F22H | 区域长度错误  
  读参数时（例如，DB太短） |
| 0 1 8F23H | 区域长度错误  
  写参数时（例如，DB太短） |
| 0 1 8F24H | 区域错误  
  读参数 |
| 0 1 8F25H | 区域错误  
  写参数 |
| 0 1 8F28H | 对齐错误  
  读参数 |
| 0 1 8F29H | 对齐错误  
  写参数 |
| 0 1 8F30H | 参数在写保护的第一个当前数据块中 |
| 0 1 8F31H | 参数在写保护的第二个当前数据块中 |
| 0 1 8F32H | 参数包含的DB号太高 |
| 0 1 8F33H | DB编号错误  
  参数在写保护的第一个当前数据块中 |
| 0 1 8F34H | DB编号错误  
  参数在写保护的第二个当前数据块中 |
| 0 1 8F35H | DB编号错误  
  参数包含的DB号太高 |
| 0 1 8F42H | 读参数时超时  
  从I/O区域读取参数 |
| 0 1 8F43H | 写参数时超时  
  向I/O区域写入参数 |
| 0 1 8F44H | 读参数时禁止访问  
  读参数期间访问参数被禁止 |
| 0 1 8F45H | 写参数时禁止访问  
  写参数期间访问参数被禁止 |
| 0 1 8F47H | 内部错误  
  例如，检测到非法的ANY引用。 |

进一步检测到的程序接口内的CPU（SFC错误）。

<table>
<thead>
<tr>
<th>CODE</th>
<th>意义</th>
</tr>
</thead>
</table>
| 0 1 8F31H | 内部错误  
  例如，检测到非法的ANY引用。 |
2.6 Program blocks for ERPC-CP

2.6.1 LOGICAL_TRIGGER for the logical trigger

Meaning of the function block

The FB56 function block LOGICAL_TRIGGER is available if you want to use a logical trigger for ERPC communication.

To start a logical trigger, call the LOGICAL_TRIGGER program block in the user program of the CPU in OB1.

Further blocks are required for the LOGICAL_TRIGGER call:

- An automatically generated instance DB
- A data block "CONF_DB"

This configuration DB contains the configuration data of the logical trigger. You create and configure the configuration DB available in the STEP 7 project.

If you want to call more than one logical trigger, you will also need to make more than one configuration DB available.

You can change the numbers of FB56 and the instance DB.

Validity

The LOGICAL_TRIGGER program block can be used with the following module types:

- CP 343-1 ERPC

Call

Call interface in FBD representation

```
LOGICAL_TRIGGER

BOOL ACT DONE
INT ID ERROR
WORD LADDR STATUS
INT CONF_DB
INT CnvlLevel

BOOL
BOOL
WORD
```
2.6 Program blocks for ERPC-CP

Table 2-21 Example in STL representation

```
call fb 56, DatabaseInstanceDB  //FB56 call with data area
ACT := TRUE,  //Trigger job when value = 1
ID := 1      //Trigger ID (possible values: 1...16)
LADDR := W#16#100,  //Module start address
CONF_DB:= 1,   //Configuration data block (here: DB 1)
CnfLevel:= 1,   //Acknowledgment mode (here: 1)
DONE := M 1.1,  //Execution code
ERROR := M 1.2,  //Error code
STATUS := MW 2);  //Status code
```

How LOGICAL_TRIGGER works

Operating principle

The following table shows the steps involved in a trigger call by the user program of the CPU.

<table>
<thead>
<tr>
<th>Step</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FB56 LOGICAL_TRIGGER is called at the intended point in the user program of the CPU with the corresponding instance DB and the selected configuration data block CONF_DB.</td>
</tr>
<tr>
<td></td>
<td>- If FB56 LOGICAL_TRIGGER is called with ACT = 1, the current trigger data is read and sent to the CP firmware.</td>
</tr>
<tr>
<td></td>
<td>- If FB56 LOGICAL_TRIGGER is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.</td>
</tr>
<tr>
<td>2</td>
<td>FB56 LOGICAL_TRIGGER reads the current trigger data.</td>
</tr>
<tr>
<td>3</td>
<td>FB56 LOGICAL_TRIGGER creates the PDU that will be sent to the CP firmware with the current data.</td>
</tr>
<tr>
<td>4</td>
<td>The CP firmware creates the data frame and transfers it to the ERPC application.</td>
</tr>
<tr>
<td>5</td>
<td>The ERPC application sends the data frame to the ERP subscriber (ERP system or MES).</td>
</tr>
</tbody>
</table>
2.6.3 Explanation of the formal parameters for LOGICAL_TRIGGER

Explanation of the formal parameters

The following table explains all the formal parameters for the FB56 LOGICAL_TRIGGER function block:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>INPUT</td>
<td>BOOL</td>
<td>0</td>
<td>If the FB is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>If the FB is called with ACT = 1, the current trigger data is read in and sent to the CP.</td>
</tr>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>Trigger ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This value identifies the logical trigger configured in the ILS Workbench.</td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.</td>
</tr>
<tr>
<td>CONF_DB</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>This data block contains the configuration data of the configured logical trigger.</td>
</tr>
<tr>
<td>CnfLevel</td>
<td>INPUT</td>
<td>INT</td>
<td>0</td>
<td>Acknowledgment mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>You can find the relevant acknowledgment based on the STATUS value in the codes of FB56.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 0 = transport acknowledgment (STATUS = 0000H)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The job is reported as successful, as soon as the data is transferred to the ERPC application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This does not necessarily mean that the data frame was sent to the ERP subscriber (ERP system or MES) and does not preclude the ERPC application detecting an error later.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1 = end-to-end acknowledgment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(STATUS = 0001H)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The job is only acknowledged after the ERPC application has checked the data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The &quot;TriggerResponse&quot; variable of the configuration DB (DB_CONF) is used to report whether or not the ERP subscriber could be reached and whether the ERPC application is in store-and-forward mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compared with the transport acknowledgment, the end-to-end acknowledgment means a longer job execution time.</td>
</tr>
</tbody>
</table>
2.6 Program blocks for ERPC-CP

### Parameter Declaration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: Job active 1: Job completed</td>
<td>The parameter indicates whether or not the job for transferring the configuration data area was handled free of errors. When the job is accepted, DONE is set to 0 by the CP. As long as DONE = 0, no further job can be triggered. For the meaning in the context of the ERROR and STATUS parameters, refer to the table &quot;FB56 LOGICAL_TRIGGER codes&quot;.</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Error situation</td>
<td>Error code For the meaning in the context of the DONE and STATUS parameters, refer to the table &quot;FB56 LOGICAL_TRIGGER codes&quot;.</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Refer to the table &quot;FB56 LOGICAL_TRIGGER codes&quot;.</td>
<td>Status code For the meaning in the context of the DONE and ERROR parameters, refer to the table &quot;FB56 LOGICAL_TRIGGER codes&quot;.</td>
</tr>
</tbody>
</table>

#### 2.6.4 LOGICAL_TRIGGER codes

**Condition codes**

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Codes relating to job execution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0000\text{H}</td>
<td>Job completed without error. The logical trigger was completed successfully.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0001\text{H}</td>
<td>Job completed without error. The database is unreachable (store-and-forward mode).</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181\text{H}</td>
<td>Job active.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>7000\text{H}</td>
<td>FB56 was called with ACT = 0. The job will, however, not be executed. Call the block at least once with ACT = 1</td>
</tr>
<tr>
<td>Codes relating to configuration and the sequence of the logical trigger</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80D2\text{H}</td>
<td>The CP in use does not support ERPC communication (wrong CP type).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183\text{H}</td>
<td>The CP in use does not support ERPC communication (wrong CP type).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8187\text{H}</td>
<td>Invalid FB56 status (unknown LOGICAL_TRIGGER_STATE). Call the block again.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A01\text{H}</td>
<td>The number of configured logical triggers equals 0.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A02\text{H}</td>
<td>The is no configuration in the configuration DB for this logical trigger. Check the ILS Workbench configuration.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A03\text{H}</td>
<td>The structure of the configuration DB is incorrect. The &quot;header identifier&quot; does not have the correct value. Correct the value of the &quot;ident&quot; variable in the configuration DB (see manual of the ERPC-CP).</td>
</tr>
<tr>
<td>DONE</td>
<td>ERROR</td>
<td>STATUS</td>
<td>Meaning</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A04H</td>
<td>The structure of the configuration DB is incorrect. Download the ILS Workbench configuration to the CP again, create and configure the configuration DB(s) again (see &quot;ERPC-CP&quot; manual).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A05H</td>
<td>The configured configuration DB does not exist on the CPU.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A06H</td>
<td>The next call called a trigger that is still running with a different ID. Check the &quot;ID&quot; in the called FB56 function blocks.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A08H</td>
<td>The configuration data in the configuration DB does not exist or is incomplete. If the error occurs only during startup of the S7 station, the cause may be that the configuration data of the logical trigger was not completely transferred to the configuration DB. If the error continues to occur, check the configuration of the ERPC symbols.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A09H</td>
<td>An unknown error was reported in the configuration DB.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A0AH</td>
<td>The logical trigger cannot be started because a new trigger configuration is currently being loaded.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A0BH</td>
<td>Error identifying the time stamp of the current data record (CPU data)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A0CH</td>
<td>The configuration DB was created with the &quot;Unlinked&quot; property. Correct the object properties of the block.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A0DH</td>
<td>Error in the input parameter CONF_DB of FB56. The parameter has the value &quot;0&quot; or higher than the maximum DB number for the CPU.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A0EH</td>
<td>The transferred trigger ID is not in the permitted range of 1...16. Correct the value in the FB56 call in the user program.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8A0FH</td>
<td>The set acknowledgment mode (CnfLevel) is invalid. Correct the value in the FB56 call in the user program.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8BxxH</td>
<td>Error copying the current variable values to the PDU of the logical trigger. The last two places (xx) are the variable number. Check the configuration of the symbol involved in the symbol table of the CPU and in the list of ERPC symbols in the properties dialog of the CP.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8C01H</td>
<td>The internal status code of FB56 is invalid. Download the ILS Workbench configuration to the CP again, create and configure the configuration DB(s) again (see &quot;ERPC-CP&quot; manual).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8C02H</td>
<td>The return value of the end-to-end acknowledgment is invalid. Download the ILS Workbench configuration to the CP again, create and configure the configuration DB(s) again (see &quot;ERPC-CP&quot; manual).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8C03H</td>
<td>The logical trigger contains more than 255 variables.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8C06H</td>
<td>Error reading the data record.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8D03H</td>
<td>The firmware is signaling a timeout during a database action.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8D04H</td>
<td>The database application is signaling a general error in the acknowledgment of the current action.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8E01H</td>
<td>The configured configuration DB on the CPU is not large enough. Change the size of the configuration DB.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8E06H</td>
<td>No connection has yet been established to the logical trigger.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8EXxH</td>
<td>Status codes with values in the range 8E02H ... 8EFFH are copies of an internal trigger response. If such values occur, they are relevant for service purposes.</td>
</tr>
</tbody>
</table>
2.6.5 The configuration data block

Preparing the configuration data block "CONF_DB"

If you use the "logical trigger" ERPC function, you will need to create a data block (DB) in STEP 7 for the configuration data of the logical trigger and specify it in the call parameters of FB56. FB56 accesses DB CONF_DB. CONF_DB has no further significance for the user program.

Programming the configuration data block

To identify the newly created DB, you will need to open the DB and specify the "header identifier" and the DB size in the first two free lines.

Open the DB in STEP 7 and configure the first two free lines with the variables "ident" and "data" as follows:

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th>Initial value</th>
<th>Comment (optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>*)</td>
<td>ident</td>
<td>DWORD</td>
<td>DW#16#45525043</td>
<td>header identifier</td>
</tr>
<tr>
<td>*)</td>
<td>data</td>
<td>array[1..2048]</td>
<td></td>
<td>DB size (see warning below)</td>
</tr>
<tr>
<td>*)</td>
<td>Byte</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*)</td>
<td>END_STRUCT *)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) Values are entered by the program

Note

DB size

2 048 bytes are recommended as the DB size. If it becomes apparent during commissioning that this value is not enough, increase it. If a value is too low, this is reported by FB56 LOGICAL_TRIGGER with an error and the STATUS "8A05H".
2.7 Configuration limits / resources required for the program blocks (Ethernet)

Required resources

Note
Note the version information of the blocks. The currently supplied block versions may differ from those shown here. Blocks with other versions have different resource requirements.

You will find information on the current block versions under entry ID:

Table 2-23 Information for FCs / FBs with S7400

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AG_SEND</td>
<td>1.2</td>
<td>FC5</td>
<td>732</td>
<td>576</td>
<td>540</td>
<td>20</td>
</tr>
<tr>
<td>AG_RECV</td>
<td>1.2</td>
<td>FC6</td>
<td>656</td>
<td>522</td>
<td>486</td>
<td>20</td>
</tr>
<tr>
<td>AG_LOCK</td>
<td>1.0</td>
<td>FC7</td>
<td>272</td>
<td>200</td>
<td>164</td>
<td>6</td>
</tr>
<tr>
<td>AG_UNLOCK</td>
<td>1.0</td>
<td>FC8</td>
<td>256</td>
<td>186</td>
<td>150</td>
<td>6</td>
</tr>
<tr>
<td>AG_CNTRL</td>
<td>1.0</td>
<td>FC10</td>
<td>2048</td>
<td>1610</td>
<td>1574</td>
<td>178</td>
</tr>
<tr>
<td>AG_CNTEX</td>
<td>1.0</td>
<td>FB10</td>
<td>7002</td>
<td>6036</td>
<td>6000</td>
<td>78</td>
</tr>
<tr>
<td>AG_LSEND</td>
<td>3.1</td>
<td>FC50</td>
<td>1044</td>
<td>846</td>
<td>810</td>
<td>52</td>
</tr>
<tr>
<td>AG_LRECV</td>
<td>3.1</td>
<td>FC60</td>
<td>1190</td>
<td>992</td>
<td>956</td>
<td>58</td>
</tr>
<tr>
<td>AG_SSEND</td>
<td>1.2</td>
<td>FC53</td>
<td>1928</td>
<td>1618</td>
<td>1582</td>
<td>154</td>
</tr>
<tr>
<td>AG_SRECV</td>
<td>1.2</td>
<td>FC63</td>
<td>1882</td>
<td>1584</td>
<td>1548</td>
<td>158</td>
</tr>
<tr>
<td>IP_CONFIG</td>
<td>1.3</td>
<td>FB55</td>
<td>1864</td>
<td>1576</td>
<td>1540</td>
<td>76</td>
</tr>
<tr>
<td>FTP_CMD</td>
<td>2.0</td>
<td>FB40</td>
<td>2400</td>
<td>2084</td>
<td>2048</td>
<td>154</td>
</tr>
<tr>
<td>FTP_CONNECT</td>
<td>1.0</td>
<td>FC40</td>
<td>1482</td>
<td>1236</td>
<td>1200</td>
<td>86</td>
</tr>
<tr>
<td>FTP_STORE</td>
<td>1.0</td>
<td>FC41</td>
<td>1794</td>
<td>1514</td>
<td>1478</td>
<td>102</td>
</tr>
<tr>
<td>FTP_RETRIEVE</td>
<td>1.0</td>
<td>FC42</td>
<td>1934</td>
<td>1642</td>
<td>1606</td>
<td>106</td>
</tr>
<tr>
<td>FTP_DELETE</td>
<td>1.0</td>
<td>FC43</td>
<td>1478</td>
<td>1232</td>
<td>1196</td>
<td>86</td>
</tr>
<tr>
<td>FTP.Quit</td>
<td>1.0</td>
<td>FC44</td>
<td>968</td>
<td>796</td>
<td>760</td>
<td>46</td>
</tr>
</tbody>
</table>
### Table 2- 24  Information for FCs / FBs with S7-300

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AG_SEND</td>
<td>4.2</td>
<td>FC5</td>
<td>1976</td>
<td>1664</td>
<td>1628</td>
<td>50</td>
</tr>
<tr>
<td>AG_RECV</td>
<td>4.7</td>
<td>FC6</td>
<td>1440</td>
<td>1206</td>
<td>1170</td>
<td>40</td>
</tr>
<tr>
<td>AG_LOCK</td>
<td>4.0</td>
<td>FC7</td>
<td>748</td>
<td>636</td>
<td>600</td>
<td>34</td>
</tr>
<tr>
<td>AG_UNLOCK</td>
<td>4.0</td>
<td>FC8</td>
<td>712</td>
<td>604</td>
<td>568</td>
<td>32</td>
</tr>
<tr>
<td>AG_CNTRL</td>
<td>1.4</td>
<td>FC10</td>
<td>1418</td>
<td>1152</td>
<td>1116</td>
<td>82</td>
</tr>
<tr>
<td>AG_CNTEX</td>
<td>1.0</td>
<td>FB10</td>
<td>4594</td>
<td>4006</td>
<td>3970</td>
<td>78</td>
</tr>
<tr>
<td>IP_CONFIG</td>
<td>1.3</td>
<td>FB55</td>
<td>2406</td>
<td>1984</td>
<td>1948</td>
<td>62</td>
</tr>
<tr>
<td>FTP_CMD</td>
<td>1.0</td>
<td>FB40</td>
<td>2590</td>
<td>2240</td>
<td>2204</td>
<td>70</td>
</tr>
<tr>
<td>FTP_CONNECT</td>
<td>1.1</td>
<td>FC40</td>
<td>928</td>
<td>774</td>
<td>738</td>
<td>68</td>
</tr>
<tr>
<td>FTP_STORE</td>
<td>1.1</td>
<td>FC41</td>
<td>1232</td>
<td>1046</td>
<td>1010</td>
<td>74</td>
</tr>
<tr>
<td>FTP_RETRIEVE</td>
<td>1.1</td>
<td>FC42</td>
<td>1306</td>
<td>1114</td>
<td>1078</td>
<td>84</td>
</tr>
<tr>
<td>FTP_DELETE</td>
<td>1.1</td>
<td>FC43</td>
<td>922</td>
<td>770</td>
<td>734</td>
<td>68</td>
</tr>
<tr>
<td>FTP_QUIT</td>
<td>1.1</td>
<td>FC44</td>
<td>452</td>
<td>370</td>
<td>334</td>
<td>28</td>
</tr>
<tr>
<td>LOGICAL_TRIGGER</td>
<td>1.0</td>
<td>FB56</td>
<td>4294</td>
<td>3648</td>
<td>3612</td>
<td>98</td>
</tr>
</tbody>
</table>
2.7 Configuration limits / resources required for the program blocks (Ethernet)
3.1 Overview of program blocks and their use

Program blocks for transferring user data

The program blocks listed below are available for transferring data cyclically on the PROFINET IO interface. The significance of the program blocks differs depending on how you use the CP (as a PROFINET IO controller or PROFINET IO device) in an S7 station.

<table>
<thead>
<tr>
<th>Program block</th>
<th>can be used with:</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S7-300</td>
<td>S7-400</td>
</tr>
<tr>
<td>PNIO_SEND (FC11)</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNIO_RECV (FC12)</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For CPs operating as PROFINET IO controller and IO device at the same time, the FCs as of version 2.0 are available.
### Program blocks for PROFINET IO (S7-300)

#### 3.1 Overview of program blocks and their use

**Program blocks for transferring data records and interrupt information**

The FBs listed below are available for transferring data (data records, alarm information) acyclically on the PROFINET IO interface. The two blocks can only be used in PROFINET IO controller mode.

<table>
<thead>
<tr>
<th>Program block</th>
<th>can be used with:</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNIO_RW_REC (FB52)</td>
<td>x</td>
<td>• Read data record (from a PROFINET IO device)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Write data record (to a PROFINET IO device)</td>
</tr>
<tr>
<td>PNIO_ALARM (FB54)</td>
<td>x</td>
<td>Receive alarm information from the PROFINET IO devices</td>
</tr>
</tbody>
</table>

**Program blocks for PROFIenergy**

The following FBs are available for the PROFIenergy functions.

<table>
<thead>
<tr>
<th>Program block</th>
<th>can be used with:</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE_START_END_CP (FB85)</td>
<td>x</td>
<td>Start / end of an energy-saving pause (on the PROFINET IO controller)</td>
</tr>
<tr>
<td>PE_CMD_CP (FB86)</td>
<td>x</td>
<td>Start / end of an energy-saving pause and reading out of energy data from the device (on the PROFINET IO controller)</td>
</tr>
<tr>
<td>PE_I_DEV_CP (FB87)</td>
<td>x</td>
<td>Execution of the PROFIenergy commands from the controller (on the PROFINET IO device) Requires supplementary functions FC 0... FC 8 (standard library).</td>
</tr>
<tr>
<td>DS3_WRITE_CP (FB53)</td>
<td>x</td>
<td>Transfer of PROFIenergy data to an ET 200S (in the PROFINET IO controller) No PROFIenergy block</td>
</tr>
</tbody>
</table>
3.2 PROFINET IO - data transfer and interrupt evaluation

3.2.1 PNIO_SEND

3.2.1.1 Meaning and call - PNIO_SEND

How It works

The PNIO_SEND program block is used to transfer data in the PROFINET IO controller or PROFINET IO device modes of the CP.

- Operating as a PROFINET IO controller
  The block transfers process data (outputs) of a specified output area to the CP for forwarding to PROFINET IO devices. As the status code, the block returns the IO Consumer Status (IOCS) of the outputs from the PROFINET IO devices.

- Operating as a PROFINET IO device
  The block reads the preprocessed process inputs of the CPU on the PROFINET IO device and transfers them to the PROFINET IO controller (configured I addresses); the block also returns the IO Consumer Status (IOCS) of the PROFINET IO controller as a status code.

The preprocessed process data is available in a DB or bit memory area.

Expansions

- As of block version V2.0
  PNIO_SEND supports the parallel operation of PROFINET IO controller and IO device on one CP. With the additional MODE parameter, you set the mode for which the FC will be called.

- As of block version V3.0
  Using the MODE parameter, you have the following options for the transfer of the IO consumer status:
  - Restriction to the group status information in the CHECK_IOCS parameter optimized for faster transfer;
  - Additional, detailed status information in the IO Consumer Status parameter
### Call interface (as of block version 2.0)

#### Call interface in FBD representation

![Call FBD diagram](image)

#### Example in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>call fc 11 (</code></td>
<td>//Call PNIO_SEND</td>
</tr>
<tr>
<td>CPLADDR:=W#16#0100,</td>
<td>//Module address from hardware configuration</td>
</tr>
<tr>
<td>MODE :=B#16#80,</td>
<td>//Controller mode or device mode;</td>
</tr>
<tr>
<td>LEN :=20,</td>
<td>//Length of the data area</td>
</tr>
<tr>
<td>IOCS :=P#DB10.DBX20.0 BYTE 3,</td>
<td>//One bit status per send data byte in DB10</td>
</tr>
<tr>
<td>DONE :=M 70.0,</td>
<td>//Address for return parameter DONE</td>
</tr>
<tr>
<td>ERROR :=M 70.1,</td>
<td>//Address for return parameter ERROR</td>
</tr>
<tr>
<td>STATUS :=MW 72,</td>
<td>//Address for return parameter STATUS</td>
</tr>
<tr>
<td>CHECK_IOCS :=M 70.2,</td>
<td>//Address for return parameter CHECK_IOCS</td>
</tr>
<tr>
<td>SEND :=P#DB10.DBX0.0 BYTE 20 );</td>
<td>//Data area to transfer from DB10</td>
</tr>
<tr>
<td></td>
<td>//(20 bytes)</td>
</tr>
</tbody>
</table>
### 3.2.1.2 Explanation of the formal parameters - PNIO_SEND

#### Explanation of the formal parameters

The following table explains all the formal parameters for FC11:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>-</td>
<td>Module start address</td>
</tr>
<tr>
<td>MODE (parameters version 2.0 or later)</td>
<td>INPUT</td>
<td>BYTE</td>
<td>The following can be specified for (XY_H):</td>
<td>Specification of the CP operating mode with: (Y) = selection of IO controller IO device mode; (X) = selection whether only group message is transferred in CHECK_IOCS or also status bits in IOCS. Notes on compatibility:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(X0_H):</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>(X0_H):</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>IO controller mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>IO device mode (without parallel operation)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>Not compatible with FC in version 1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(X1_H):</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>(X1_H):</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IO device mode (both modes at same time)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(0Y_H)</td>
<td>Status bits are transferred in IOCS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(8Y_H)</td>
<td>Restriction to group message in CHECK_IOCS; no status bits in IOCS.</td>
</tr>
</tbody>
</table>

Notes on compatibility:
- The version 1.0 FC can continue to be used as long as the CP is not being operated as an IO controller and IO device at the same time.
- When \(MODE=0\), the FC as of version 2.0 behaves like the FC version 1.0.
- When \(MODE=0\) and \(MODE = 1\), the FC as of version 3.0 behaves like the FC version 2.0.
## Parameter Declaration Data type Possible values Description

### SEND
- **Declaration**: IN_OUT
- **Data type**: ANY (as VARTYPE only BYTE is permitted)
- **Possible values**: The address of the data area points to one of the alternatives:
  - Memory bit area
  - Data block area
- **Description**: Specifies the address and length

**IO controller mode:**
The length should match the total length of the distributed IO configured, whereby address gaps are also transmitted.
The length can also be shorter than the total length of the distributed IO, for example when the block is called more than once in one OB. It must, however, have the total length in at least one call.

**IO device mode:**
The data structure results from the order of the slots of the input modules configured for this PROFINET IO device on the PROFINET IO controller line and their length without address gaps.

**Notes:**
- The block begins to transfer the data at address 0 regardless of how you configured the addresses (regardless of the lowest configured address).
- Specifying an I/O area is not permitted since you must first check the IOCS for GOOD before data can be accepted in the I/O.
## 3.2 PROFINET IO - data transfer and interrupt evaluation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEN</td>
<td>INPUT</td>
<td>INT</td>
<td>Value &gt; 0</td>
<td>Length of the data area to be transferred in bytes. The maximum total length of the data areas to be transferred can be found in the devicespecific Part B of this manual in the &quot;Performance data&quot; chapter. This may differ for controller or device mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The transfer of the data always begins with address 0 regardless of the configuration. Please note that the IO address &quot;0&quot; with a length of 1 is included.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IO controller mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• The highest configured address of the devices must be specified here. The individual areas are not grouped together. If the block is called more than once, LEN can also be shorter than the highest address. The highest address should be specified in at least one call (compare &quot;SEND&quot; parameter).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• The data is transferred in the order of the logical addresses (as with PROFIBUS DP).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IO device mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• The data is transferred in the order of the slots as the configured input modules on the PROFINET IO controller chain for this PROFINET IO device.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Make sure that the length programmed here and the configuration of the PROFINET IO controller are consistent. The entire data area length including any gaps is transferred for the device.</td>
</tr>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: new data accepted</td>
<td>This parameter indicates whether or not the job was completed without errors.</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: -1: Error</td>
<td>Error code</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>-</td>
<td>Status code</td>
</tr>
</tbody>
</table>
### Parameter Declaration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK_IOCS</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: All IOCS set to GOOD 1: At least one IOCS set to BAD</td>
<td>Group message that indicates whether or not it is necessary to evaluate the IOCS status area. CHECK_IOCS is always returned regardless of the MODE parameter.</td>
</tr>
<tr>
<td>IOCS</td>
<td>OUTPUT</td>
<td>ANY</td>
<td>The address of the data area points to one of the alternatives: Memory bit area Data block area</td>
<td>IO Consumer Status A status bit is transferred per byte of user data. Requirement: Transfer is requested in the MODE parameter (MODE=0 or MODE=1). The parameter is relevant only in this mode. The length information depends on the length in the LEN parameter (one bit per byte) = (Length LEN + 7/8) Controller mode: Address gaps are also transferred according to the SEND parameter. Address gaps are transferred with the status GOOD. Device mode: Address gaps are not transferred. The block begins the transfer of the status for address 0. Note: The minimum length of the ANY pointer is (length LEN + 7/8)</td>
</tr>
</tbody>
</table>

---

**Note**

**Wait for confirmation of execution**

Execute the following actions only after the block has signaled either DONE = 1 or ERROR = 1:
- Evaluate output parameters;
- Change the MODE parameter.

---

**Note**

You must assume that the returned IOCS status does not arrive timesynchronized with the data (SEND parameter) but delayed by one user program cycle. This means: User data and IOCS are not consistent.
3.2.1.3 Condition codes of PNIO_SEND

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8180H</td>
<td>• Data transfer active; or • The CP is in STOP mode.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Module does not support block version 2.0. Remedy: Use block version 1.0.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>New data transferred without error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>• PROFINET IO configuration missing; or • CPLADDR parameter is bad; or • The CP is in STOP mode; or • The interconnection of MODE does not match the module configuration or there is an incorrect interconnection of the MODE parameter. Extra in device mode: • The connection between PROFINET IO controller and PROFINET IO device is down, or • PROFINET IO controller not reachable or • Total lengths (configuration and LEN parameter) are not consistent.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8184H</td>
<td>System error or bad parameter type.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8185H</td>
<td>Parameter LEN is greater than source area SEND or target buffer (IOCS) is too small.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F22H</td>
<td>Area length error reading a parameter (e.g. DB too short).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F23H</td>
<td>Area length error writing a parameter (e.g. DB too short).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F24H</td>
<td>Range error when reading a parameter.</td>
</tr>
</tbody>
</table>

Note

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.
### Program blocks for PROFINET IO (S7-300)

#### 3.2 PROFINET IO - data transfer and interrupt evaluation

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8F25H</td>
<td>Range error when writing a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F28H</td>
<td>Alignment error when reading a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F29H</td>
<td>Alignment error when writing a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F30H</td>
<td>Parameter is in the write-protected 1st current data block.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F31H</td>
<td>Parameter is in the write-protected 2nd current data block.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F32H</td>
<td>Parameter contains a DB number that is too high.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F3AH</td>
<td>Destination area is not loaded (DB).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F42H</td>
<td>Timeout reading a parameter from the I/O area.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F43H</td>
<td>Timeout writing a parameter to the I/O area.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F44H</td>
<td>Access to a parameter to be read during block execution is prevented.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F45H</td>
<td>Access to a parameter to be written during block execution is prevented.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F7FH</td>
<td>Internal error, e.g. illegal ANY reference.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8090H</td>
<td>Module with this address does not exist.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80A0H</td>
<td>Negative acknowledgment writing to the module.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80A1H</td>
<td>Negative acknowledgment writing to the module.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B0H</td>
<td>The module does not recognize the data record.</td>
</tr>
</tbody>
</table>
| 0    | 1     | 80B1H  | • The specified data record length is incorrect.  
|      |       |        | or  
|      |       |        | • The CP changes to STOP. |
| 0    | 1     | 80C0H  | The data record cannot be read. |
| 0    | 1     | 80C1H  | The specified data record is currently being processed. |
| 0    | 1     | 80C2H  | There are too many jobs pending. |
| 0    | 1     | 80C3H  | Resources occupied (memory). |
| 0    | 1     | 80C4H  | Communication error (occurs temporarily, it is usually best to repeat the job in the user program). |
3.2.2 PNIO_RECV

3.2.2.1 Meaning and call - PNIO_RECV

How It works

The PNIO_RECV program block is used to receive data in the PROFINET IO controller or PROFINET IO device modes of the CP.

- Operating as a PROFINET IO controller
  The block received the process data from PROFINET IO devices (inputs of the controller) and the IO provider status (IOPS) from the PROFINET IO devices in the specified input areas.

- Operating as a PROFINET IO device
  The block receives the data transferred by the PROFINET IO controller (configured O addresses) and the IO Provider Status (IOPS) of the PROFINET IO controller and writes it to the data areas on the CPU of the PROFINET IO device reserved for the process outputs.

Expansions

- As of block version V2.0
  PNIO_RECV supports the parallel operation of PROFINET IO controller and IO device on one CP. With the additional MODE parameter, you set the mode for which the FC will be called.

- As of block version V3.0
  Using the MODE parameter, you have the following options for the transfer of the IO provider status:
  - Restriction to the group status information in the CHECK_IOPS parameter optimized for faster transfer;
  - Additional, detailed status information in the IO Provider Status parameter
Program blocks for PROFINET IO (S7-300)

3.2 PROFINET IO - data transfer and interrupt evaluation

Call interface (as of block version 2.0)

Call interface in FBD representation

```
Call PNIO_RECV
```

Example in STL representation

```
call fc 12 
  CPLADDR :=W#16#0100,
  MODE :=B#16#80,
  LEN :=7,
  IOPS :=P#DB11.DBX7.0 BYTE 1,
  NDR :=M 74.0,
  ERROR :=M 74.1,
  STATUS :=MW76,
  CHECK_IOPS :=M74.2,
  ADD_INFO :=MW 26,
  RECV :=P#DB11.DBX0.0 BYTE 7 );
```

//Call PNIO_RECV
//Module address from hardware configuration
//Controller mode or device mode;
//IOCS status bits are not transferred.
//Length of the data area
//One bit status per received data byte in DB11
//Address for return parameter NDR
//Address for return parameter ERROR
//Address for return parameter STATUS
//Address for return parameter CHECK_IOPS
//Diagnostic information
//Received data in DB11 (7 bytes)

See also

Data consistency (Page 158)
Substitute values (Page 159)
### 3.2.2.2 Explanation of the formal parameters - PNIO_RECV

**Explanation of the formal parameters**

The following table explains all the formal parameters for FC12:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>-</td>
<td>Module start address</td>
</tr>
<tr>
<td>MODE (parameters version 2.0 or later)</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Values with the following meaning can be specified in the MODE = XYH parameter:</td>
<td>Specification of the CP operating mode with: Y = selection of IO controller IO device mode; X = selection whether only group message is transferred in CHECK_IOPS or also status bits in IOPS. Notes on compatibility:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• X0H:</td>
<td>• The version 1.0 FC can continue to be used as long as the CP is not being operated as an IO controller IO device at the same time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- IO controller mode</td>
<td>• When MODE=0, the FC as of version 2.0 behaves like the FC version 1.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- IO device mode (without parallel operation)</td>
<td>• When MODE=0 and MODE = 1, the FC as of version 3.0 behaves like the FC version 2.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Not compatible with FC in version 1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• X1H:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- IO device mode (both modes at same time)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 0YH</td>
<td>Status bits are transferred in IOPS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 8YH</td>
<td>Restriction to group message in CHECK_IOPS; no status bits in IOPS.</td>
</tr>
</tbody>
</table>
### 3.2 PROFINET IO - data transfer and interrupt evaluation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
</table>
| RECV      | IN_OUT      | ANY (as VARTYPE only BYTE is permitted) | The address of the data area points to one of the alternatives:  
  - Memory bit area  
  - Data block area | Specifies the address and length  
  IO controller mode:  
  The length should match the total length of the distributed IO configured, whereby address gaps are also transmitted.  
  The length can also be shorter than the total length of the distributed IO, for example when the block is called more than once in one OB. It must, however, have the total length in at least one call.  
  IO device mode:  
  The data structure results from the order of the slots of the output modules configured for this PROFINET IO device on the PROFINET IO controller line and their length without address gaps.  
  Notes:  
  - The block begins to transfer the data at address 0 regardless of how you configured the addresses (regardless of the lowest configured address).  
  - Specifying an I/O area is not permitted since you must first change the IOPS for GOOD before data can be accepted in the I/O. |
### Parameter Declaration Data type Possible values Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
</table>
| LEN       | INPUT       | INT       | Value > 0       | Length of the data area to be transferred in bytes. The transfer of the data always begins with address 0 regardless of the configuration. Please note that the IO address "0" with a length of 1 is included.  
  - **IO controller mode:**  
    - The highest configured address of the devices must be specified here. The individual areas are not grouped together. If the block is called more than once, LEN can also be shorter than the highest address. The highest address should be specified in at least one call (compare "RECV" parameter).  
    - The data is transferred in the order of the logical addresses (as with PROFIBUS DP).  
  - **IO device mode:**  
    - The data is transferred in the order of the slots corresponding to the configuration of the input modules on the PROFINET IO controller line for this PROFINET IO device.  
    - Note: Make sure that the length programmed here and the configuration of the PROFINET IO controller are consistent. The entire data area length including any gaps is transferred for the device. |

<table>
<thead>
<tr>
<th>NDR</th>
<th>OUTPUT</th>
<th>BOOL</th>
<th>0: - 1: Data accepted</th>
<th>This parameter indicates whether or not the job was completed without errors.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Error</td>
<td>Error code</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>-</td>
<td>Status code</td>
</tr>
<tr>
<td>CHECK_</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: All IOPS set to GOOD 1: At least one IOPS set to BAD</td>
<td>Group message that indicates whether or not it is necessary to evaluate the IOPS status area. CHECK_IOPS is always returned regardless of the MODE parameter.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Possible values</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>-----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| IOPS       | OUTPUT      | ANY (as VARTYPE only BYTE is permitted) | The address of the data area points to one of the alternatives:  
- Memory bit area  
- Data block area  
Length:  
For the maximum value, refer to the devicespecific Part B of this manual in the section "Performance data". This may differ for controller or device mode. | IO Provider Status  
A status bit is transferred per byte of user data.  
Requirement: Transfer is requested in the MODE parameter (MODE=0 or MODE=1). The parameter is relevant only in this mode.  
The length information depends on the length in the RECV parameter (one bit per byte)  
\(= (\text{Length LEN} + 7/8)\)  
Controller mode:  
Address gaps are also transferred according to the RECV parameter.  
Address gaps are transferred with the status GOOD.  
Device mode:  
Address gaps are not transferred.  
The block begins the transfer of the status for address 0.  
Note:  
- The minimum length of the ANY pointer is \((\text{length LEN} + 7/8)\) |
| ADD_INFO   | OUTPUT      | WORD      | Additional Diagnostic Information  
In controller mode:  
- 0: No alarm  
- \(>0\): Number of pending alarms  
In device mode, the parameter is always = 0. | Parameter expansion  
Note:  
The ADD_INFO parameter is also updated when there are no INPUT addresses configured on the PROFINET IO controller. In this case, the PNIO_RECV block is called with a length LEN > 0 (for example LEN = 1 byte). It then transfers an address gap of 1 byte.  
The parameter expansion can be used for CPs as of the following firmware version:  
- CP 343−1 (EX30) as of firmware V2.0  
- CP 343−1 Lean (CX10) as of firmware V2.0  
- CP 343−1 Advanced (GX30) as of firmware V1.0  
In older firmware versions, the parameter is reserved. |
Wait for confirmation of execution

 Execute the following actions only after the block has signaled either DONE = 1 or ERROR = 1:

• Evaluate output parameters;
• Change the MODE parameter.

### 3.2.2.3 Condition codes of PNIO_RECV

#### Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

**Note**

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8180H</td>
<td>• Data acceptance active; or • The CP is in STOP mode.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Module does not support block version 2.0. Remedy: Use block version 1.0.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>New data accepted without error.</td>
</tr>
</tbody>
</table>
### Program blocks for PROFINET IO (S7-300)

**3.2 PROFINET IO - data transfer and interrupt evaluation**

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0   | 1     | 8183H  | • PROFINET IO configuration missing;  
|     |       |        | or  
|     |       |        | • CPLADDR parameter is bad;  
|     |       |        | or  
|     |       |        | • The CP is in STOP mode.  
|     |       |        | or  
|     |       |        | • The interconnection of MODE does not match the module configuration or there is an incorrect interconnection of the MODE parameter.  
| Extra in device mode: |       |        | • The connection between PROFINET IO controller and PROFINET IO device is down,  
|     |       |        | or  
|     |       |        | • PROFINET IO controller not reachable  
|     |       |        | or  
|     |       |        | • Total lengths (configuration and LEN parameter) are not consistent |
| 0   | 1     | 8184H  | System error or bad parameter type.  
| 0   | 1     | 8185H  | Destination buffer (RECV of IOCS) is too small.  
| 0   | 1     | 8F22H  | Area length error reading a parameter (e.g. DB too short).  
| 0   | 1     | 8F23H  | Area length error writing a parameter (e.g. DB too short).  
| 0   | 1     | 8F24H  | Range error when reading a parameter.  
| 0   | 1     | 8F25H  | Range error when writing a parameter.  
| 0   | 1     | 8F28H  | Alignment error when reading a parameter.  
| 0   | 1     | 8F29H  | Alignment error when writing a parameter.  
| 0   | 1     | 8F30H  | Parameter is in the write-protected 1st current data block.  
| 0   | 1     | 8F31H  | Parameter is in the write-protected 2nd current data block.  
| 0   | 1     | 8F32H  | Parameter contains a DB number that is too high.  
| 0   | 1     | 8F3A_H | Destination area is not loaded (DB).  
| 0   | 1     | 8F42H  | Timeout reading a parameter from the I/O area.  
| 0   | 1     | 8F43H  | Timeout writing a parameter to the I/O area.  
| 0   | 1     | 8F44H  | Access to a parameter to be read during block execution is prevented.  
| 0   | 1     | 8F45H  | Access to a parameter to be written during block execution is prevented.  
| 0   | 1     | 8F7F_H | Internal error, e.g. illegal ANY reference.  
| 0   | 1     | 8090H  | Module with this address does not exist.  
| 0   | 1     | 80A0H  | Negative acknowledgment writing to the module.  
| 0   | 1     | 80A1H  | Negative acknowledgment writing to the module.  
| 0   | 1     | 80B0H  | The module does not recognize the data record.  
| 0   | 1     | 80B1H  | • The specified data record length is incorrect.  
|     |       |        | or  
|     |       |        | • The CP changes to STOP.  
| 0   | 1     | 80C0H  | The data record cannot be read.  
| 0   | 1     | 80C1H  | The specified data record is currently being processed.  
| 0   | 1     | 80C2H  | There are too many jobs pending.  

---

Program blocks for SIMATIC NET S7 CPs  
Programming Manual, 11/2015, C79000-G8976-C229-08
### 3.2.3 General characteristics of the FCs for PROFINET IO

**IO Consumer Status (IOCS) and IO Provider Status (IOPS)**

For both communication partners - CPU/CP on the one hand and IO device on the other - there is the status information GOOD or BAD for the data. This status information is transferred parallel to the data. The status of the partner that sends the data is called IOPS (IO Provider Status), the status of the receiving partner is called IOCS (IO Consumer Status).

The IOPS and IOCS status are not necessarily identical. It is, for example, possible that the S7-300 CPU is in STOP mode (output disable or no PROFINET IO blocks active). In this case, the CP as PROFINET IO controller transfers the BAD status to the IO devices.

You receive a group message informing you whether an evaluation of the status information is necessary. With the MODE parameter, you also decide whether you want detailed status information returned on the call interface. The jobs are handled faster if you restrict the information to the group message.

**Relationship between block call and IO data**

- **Operation as PROFINET IO controller**
  As a PROFINET IO controller, the CP does not monitor the cyclic calls of the PNIO_SEND/RECV blocks. If the blocks are not called, the last transferred IO data and IOCS/IOPS data are taken as valid.

- **Operation as PROFINET IO device**
  FC11 and FC12 each have their own watchdog. Depending on the CPU cycle time, the connection to the PROFINET IO controller is terminated if one of the two blocks is no longer called following the initialization phase.

**Optimizing data transfer (only when operating as PROFINET IO controller)**

It is possible to call the blocks with a length (LEN parameter) that is shorter than the configured total length of the IO data on the PNIO chain.

You can use this so that timecritical data is transferred in every CPU cycle whereas non critical data is not transferred in every cycle.

**Example:**
You could, for example, transfer only the first data area (timecritical data) in every cycle and the total length of the configured IO data in every second cycle. To do this, place the timecritical data in the lower area (starting at IO address 0) during configuration.
3.2.4 Data consistency

The entire input or output data area of the PROFINET IO controller is always transferred in its entirety and is therefore consistent.

- Operating as PROFINET IO controller
  Regardless of this, using the length information in the block call, you can also read or output an input or output area smaller than the configured area consistently.

Note: You should, however, bear in mind that in terms of the "IO user data" within a PROFINET IO system, data consistency can only be guaranteed within the individual IO slots. This applies regardless of the fact that consistent data transfer between CPU and IO controller is guaranteed for the blocks described here.

Block call

To guarantee data consistency, you may, however, only access the IO data when the block has completed free of errors (output parameter NDR = TRUE). You must also check that the IOCS or IOPS status for the data is GOOD.

Example

In a normal situation (depending on the total length of the IO data), the block will run over several user program cycles until the condition code DONE/NDR = 1 is signaled.

Note: The user program cycle and the cycle of the IO data exchange between the PROFINET IO controller and PROFINET IO devices are independent of each other.
3.2.5 Substitute values

Operational situations

The setting of substitute values is supported for the two following operational situations:

- Substitute values during startup (mode change on the CPU from STOP to RUN)
- Substitute values if problems are detected (remove/insert or station failure/return)

Substitute values during startup

You can initialize the outputs with substitute values by setting a memory bit ("startup" memory bit) in the startup OB. In cyclic mode (OB1), evaluate this "startup" memory bit to call PNIO_SEND with the initialization values when appropriate.

Substitute values if a problem occurs (only when operating as PROFINET IO controller)

If there is a fault (device/submodule failed), you can find out which submodules have failed by querying the status information IOCS / IOPS status. You then have the option of setting substitute values.

3.2.6 PNIO_RW_REC

3.2.6.1 Meaning and call - PNIO_RW_REC

Significance and how it works

FB52 is used both for the "read data record" and the "write data record" function in PROFINET IO controller mode. FB52 can only execute one of the functions at any one time. The "read data record" or "write data record" function is controlled by the WRITE_REC parameter.

CPs support only I&M0 and I&M1.

Example: The CP can be informed of the location ID and plant designation using the "write data record" function (if this parameter was not already set in the properties dialog of the CP in STEP 7). This is done using the maintenance data record "IM1" with index AFF1H.

Data record I&M0 with index AFF0H (order number, serial number, version) can only be read out (write protected).

You will find details of the supported data records and their structure at the following Internet address:

3.2 PROFINET IO - data transfer and interrupt evaluation

Call interface

Call interface in FBD representation:

Example in STL representation:

```
CALL FB 52, DB 52 (  
CPLADDR := W#16#0110,  //Module address from hardware configuration  
WRITE_REC := M 1.1,  //Job type  
ID := W#16#86A,  //Logical address of the module to be addressed  
INDEX := W#16#8000,  //Data record number  
DONE := M 1.3,  //Address for return parameter DONE  
ERROR := M 1.1,  //Address for return parameter ERROR  
STATUS := MW 12,  //Address for return parameter STATUS  
LEN := MW 16,  //Length of the data record in bytes  
RECORD := P#DB3.DBX0.0 BYTE 80 );  //Target or source of the data record  // (here max. 80 bytes)
```

3.2.6.2 Explanation of the formal parameters - PNIO_RW_REC

Explanation of the formal parameters

The following table explains all the formal parameters for FB52:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>-</td>
<td>Module start address</td>
</tr>
<tr>
<td>WRITE_REC</td>
<td>INPUT</td>
<td>BOOL</td>
<td>0: Read data record 1: Write data record</td>
<td>Job type; The parameter must not be changed while the block is executing.</td>
</tr>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Logical address of the PROFINET IO component (module or submodule). For an output module, bit 15 is set (example of output address 5: ID:=DW#16#8005). For a mixed module, the lower of the two addresses must be specified.</td>
</tr>
</tbody>
</table>
3.2.6.3 Condition codes of PNIO_RW_REC

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.
Program blocks for PROFINET IO (S7-300)

3.2 PROFINET IO - data transfer and interrupt evaluation

Table 3- 3 PNIO_RW_REC condition codes

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8180H</td>
<td>Data transfer active</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>Data record transferred successfully</td>
</tr>
</tbody>
</table>
| 0    | 1     | 8183H  | • No PROFINET IO controller configuration,  
|      |       |        | • wrong CPLADDR  
|      |       |        | or  
|      |       |        | • CP in STOP mode |
| 0    | 1     | 8184H  | System error or illegal parameter type |
| 0    | 1     | 8185H  | Destination buffer (RECORD) is too short |
| 0    | 1     | 8F22H  | Area length error reading a parameter (e.g. DB too short) |
| 0    | 1     | 8F23H  | Area length error writing a parameter (e.g. DB too short) |
| 0    | 1     | 8F24H  | Area error reading a parameter |
| 0    | 1     | 8F25H  | Area error writing a parameter |
| 0    | 1     | 8F28H  | Orientation error when reading a parameter |
| 0    | 1     | 8F29H  | Alignment error writing a parameter |
| 0    | 1     | 8F30H  | Parameter is in the write-protected 1st active data block. |
| 0    | 1     | 8F31H  | Parameter is in the write-protected 2nd active data block. |
| 0    | 1     | 8F32H  | Parameter contains a DB number that is too high. |
| 0    | 1     | 8F3AH  | Destination area not loaded (DB). |
| 0    | 1     | 8F42H  | Timeout reading a parameter from the I/O area |
| 0    | 1     | 8F43H  | Timeout writing a parameter to the I/O area |
| 0    | 1     | 8F44H  | Access to a parameter to be read during block execution is prevented. |
| 0    | 1     | 8F45H  | Access to a parameter to be written when executing the block is disabled. |
| 0    | 1     | 8F7FH  | Internal error, e.g. illegal ANY reference |
| 0    | 1     | 8090H  | Module with this address does not exist. |
| 0    | 1     | 80A0H  | Negative acknowledgment reading from the module |
| 0    | 1     | 80A1H  | Negative acknowledgment writing to the module |
| 0    | 1     | 80A3H  | General PROFINET IO context management error |
| 0    | 1     | 80A9H  | PROFINET IO device or module reports an illegal type. |
| 0    | 1     | 80B0H  | The module does not recognize the data record. |
| 0    | 1     | 80B1H  | • The specified data record length is incorrect;  
|      |       |        | or  
|      |       |        | • The CP changes to STOP. |
| 0    | 1     | 80B2H  | The logical address or the configured slot is not in use. |
| 0    | 1     | 80B4H  | PROFINET IO device or module reports access to an illegal area. |
| 0    | 1     | 80B6H  | PROFINET IO device or module denies access. |
| 0    | 1     | 80B8H  | The module reports an illegal parameter. |
| 0    | 1     | 80B9H  | The block type and / or version is not permitted. |
| 0    | 1     | 80C0H  | The data record cannot be read. |
3.2.7  **PNIO_ALARM**

### 3.2.7.1  **Meaning and call - PNIO_ALARM**

**How It works**

FB54 is used for alarm evaluation by a CP 3431 operating as PROFINET IO controller and should be called in its user program when the ADD_INFO parameter in FC12 is not equal to 0. After complete and errorfree transfer of all OUTPUT parameters of FB54, the received alarms are acknowledged automatically.

The alarms are forwarded to the user program in the chronological order in which they were signaled. Older alarms that have not yet been signaled to the user program and that become invalid due to more recent alarms are not deleted by the newer alarms.

**Note**

As long as the block has not yet been called, the alarms are acknowledged automatically in the CP.

If FB54 has been called (at least) once in the user program, it must continue to be called to acknowledge pending alarms. This is the situation when FC12 signals a value not equal to "0" in the ADD_INFO parameter.

If FB54 is no longer called after it has been called once or more in the user program, alarms are not acknowledged and there is no guarantee that the IO image will be updated correctly. The can occur, for example, following a station return alarm. The need to call FB54 can only be reset by restarting the CP (power cycle).
Program blocks for PROFINET IO (S7-300)

3.2 PROFINET IO - data transfer and interrupt evaluation

Call interface

Call interface in FBD representation

Example in STL representation:

```
CALL FB 54, DB 54 //Call PNIO_ALARM
CPLADDR:= W#16#0110, //Module address from hardware configuration
DONE := M 1.1, //Address for return parameter DONE
ERROR := M 1.2, //Address for return parameter ERROR
NEW := M 1.3, //TRUE: A new alarm was received
STATUS := MW 12, //Error code
ID := MW14, //Logical start address of the reporting component
LEN := MW 16, //Length of the received alarm information (AINFO)
MODE := MD 18, //RESERVED (value always = 0)
TINFO := P#DB4.DBX0.0 BYTE 32, //task information
AINFO := P#DB4.DBX32.0 BYTE 532 ); //alarm information
```

3.2.7.2 Explanation of the formal parameters - PNIO_ALARM

Explanation of the formal parameters

The following table explains all the formal parameters for FB54:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>-</td>
<td>Start address of the module that caused the error</td>
</tr>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Alarm information transferred successfully</td>
<td>This parameter indicates whether or not the job was completed without errors. If DONE = 1, the NEW parameter must also be checked.</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Error</td>
<td>Error code</td>
</tr>
</tbody>
</table>
### Program blocks for PROFINET IO (S7-300)

#### 3.2 PROFINET IO - data transfer and interrupt evaluation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: Data transfer active or no new alarm 1: New alarm received and acknowledged</td>
<td>If DONE = 1 and NEW = 1, a new received alarm is signaled.</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>-</td>
<td>Status code</td>
</tr>
<tr>
<td>ID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Logical start address of the PNIO component that triggers the alarm (module or submodule). For an output module, bit 15 is set (example of output address 5: ID:=DW#16#8005). For a mixed module, the lower of the two addresses is specified.</td>
<td></td>
</tr>
<tr>
<td>LEN</td>
<td>OUTPUT</td>
<td>INT</td>
<td>Length of the received alarm information (AINFO)</td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>IN_OUT</td>
<td>DWORD</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>TINFO</td>
<td>IN_OUT</td>
<td>ANY</td>
<td>The address of the data area points to one of the alternatives: • Memory bit area • Data block area The length of the ANY pointer must be &gt;= 32 bytes.</td>
<td>(task information) Destination area for the alarm management information. The error OB start information (OB header = byte 0...19 of TINFO) is reproduced as far as possible by the CP firmware. See also 1)</td>
</tr>
<tr>
<td>AINFO</td>
<td>IN_OUT</td>
<td>ANY</td>
<td>The address of the data area points to one of the alternatives: • Memory bit area • Data block area The length of the ANY pointer must be greater than or equal to the maximum additional alarm information that can be expected, maximum 1432 bytes (see LEN parameter)</td>
<td>(alarm information) Destination area for header information and additional alarm information. If the ANY pointer AINFO is too low, the information will be truncated. See also 1)</td>
</tr>
</tbody>
</table>

1) Reference Manual "STEP 7 - System and Standard Functions for S7-300 and S7-400", receiving an alarm with SFB54 "RALRM" /5/ (Page 282)
### Condition codes of PNIO_ALARM

#### Condition codes

The following table shows the condition codes formed by the DONE, NEW, ERROR and STATUS parameters that must be evaluated by the user program.

**Note**

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

<table>
<thead>
<tr>
<th>DONE</th>
<th>NEW</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8180H</td>
<td>Data transfer active</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>Alarm data successfully transferred and alarm acknowledged</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0000H</td>
<td>No alarm data exist</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>• No PROFINET IO controller configuration,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• wrong CPLADDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• CP in STOP mode</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8184H</td>
<td>System error or illegal parameter type</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8185H</td>
<td>Destination buffer (TINFO or AINFO) is too short</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F22H</td>
<td>Area length error reading a parameter (e.g. DB too short)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F23H</td>
<td>Area length error writing a parameter (e.g. DB too short)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F24H</td>
<td>Area error reading a parameter</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F25H</td>
<td>Area error writing a parameter</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F28H</td>
<td>Orientation error when reading a parameter</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F29H</td>
<td>Alignment error writing a parameter</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F30H</td>
<td>Parameter is in the write-protected first active data block</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F31H</td>
<td>Parameter is in the write-protected second active data block</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F32H</td>
<td>The DB number in the parameter is too high</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F3AH</td>
<td>Destination area not loaded (DB)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F42H</td>
<td>Timeout reading a parameter from the I/O area</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F43H</td>
<td>Timeout writing a parameter to the I/O area</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F44H</td>
<td>Access to a parameter to be read during block execution is prevented.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F45H</td>
<td>Address of the parameter to be written is disabled in the accessed rack</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8F7FH</td>
<td>Internal error, e.g. illegal ANY reference</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8090H</td>
<td>Module with this address does not exist</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>80A0H</td>
<td>Negative acknowledgment reading from the module</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>80A1H</td>
<td>Negative acknowledgment writing to the module</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>80B0H</td>
<td>Module does not recognize the data record</td>
</tr>
</tbody>
</table>
### 3.3 PROFIlenergy

**PROFIlenergy**

The PROFIlenergy functions in PROFINET are used for energy management of plants. These include the planned or spontaneous shutdown of individual field devices, units or plant sections to save energy. The shutdown takes place during times without production or during breaks in production. Energy and diagnostics data can be read from devices included in the energy concept and that support these functions.

**PROFIlenergy controller**

The commands for shutting down are output by the higher-level controller, in PROFINET IO, the IO controller.

With the SIMATIC S7-300, an S7-300 CPU with PROFIlenergy functionality can be the PROFIlenergy controller.

**PROFIlenergy devices**

The commands of the PROFIlenergy controller are processed by the IO devices with PROFIlenergy functionality to shut down connected devices in the field.

In the context of the PROFIlenergy program blocks, an IO device with PROFIlenergy functionality is known as a PROFIlenergy device. With the SIMATIC S7-300, an S7-300 CPU with PROFIlenergy functionality can be the PROFIlenergy device.

**I-devices**

In SIMATIC S7, an intelligent device (I-device) itself can have subordinate PROFIlenergy devices. In this case, the I-device also has the function of a PROFIlenergy controller.

---

<table>
<thead>
<tr>
<th>DONE</th>
<th>NEW</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>80B1H</td>
<td>- The specified data record length is incorrect or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- The CP changes to STOP</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>80C0H</td>
<td>The data record cannot be read</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>80C1H</td>
<td>The specified data record is being processed</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>80C2H</td>
<td>Too many jobs pending</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>80C3H</td>
<td>Resources (memory) occupied</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>80C4H</td>
<td>Communication error (occurs temporarily, it is usually best to repeat the job in the user program).</td>
</tr>
</tbody>
</table>
Energy saving modes and PE_MODE_ID

Many devices support only the operating states "ready to operate" (power ON) and "pause" (power OFF). Scaled energy-saving states with different energy-saving modes can be specified for devices that support this or for groups of units in the controller of the PROFIenergy device. With PROFIenergy, these various states of energy consumption can be assigned to the devices that will be shut down in the field.

The various states of energy consumption are known as "energy-saving modes". For each individual energy-saving mode, a defined "PE_MODE_ID" is specified.

Programming of the features of the energy saving modes

The details of the energy-saving modes (addressed field device, pause duration etc) are programmed in the user program of the CPU of the PROFIenergy device.

3.3.1 PROFIenergy program blocks for the CP 300

Implementation of the PROFIenergy functions in S7-300

With a SIMATIC S7-300, the PROFIenergy functions are provided by program blocks for the IO controller and the IO device.

Note that an S7-300 CPU and a CP 300 use different PROFIenergy program blocks.

PROFIenergy specification

The functions of the PROFIenergy program blocks for the CP 300 are based on the following specification of the PROFIBUS Users Organization (PNO):

Common Application Profile PROFIenergy, Technical Specification for PROFINET, Version 1.0, January 2010, Order No. 3.802

PROFIenergy program blocks for the CP 300

The PROFIenergy program blocks are called by the user program of the CPU. The following PROFIenergy program blocks are available for the PROFIenergy functions of the CP 300:

- CP 300 as IO controller:
  - PE_START_END_CP
    Program block for initiating and ending pauses for power supply and setting defined energy-saving modes for the PROFIenergy device.
  - PE_CMD_CP
    Program block for initiating and ending pauses for power supply and setting defined energy-saving modes and for querying measured energy values from the PROFIenergy device.
The two program blocks can be used as alternatives. Compared with PE_START_END_CP, PE_CMD_CP has an expanded range of functions for the integration of measured energy values.

For each PROFenergy device, the program block must be called separately.

- **DS3_WRITE_CP**
  
  Does not belong to the PROFenergy function blocks, but expands the PROFenergy functions for an ET 200S.
  
  With DS3_WRITE_CP, the settings for the switching behavior of up to 8 slots (in this case: power modules) of the ET 200S are specified.

- **CP 300 as IO device:**
  
  - **PE_I_DEV_CP**
    
    Receives all PROFenergy commands and allows the user program to execute the PROFenergy functions.
    
    Makes the response frames of the IO device available to the IO controller.
    
    PE_I_DEV_CP is called cyclically by the user program of the IO device.
    
    - Supplementary program blocks (FC 0...FC 8) for PE_I_DEV_CP:
      
      These FCs make the response data available for PE_I_DEV_CP. The FCs must be called in the user program and linked with PE_I_DEV_CP.

If the PROFenergy device is an I-device and itself has subordinate PROFenergy devices, PE_START_END_CP or PE_CMD_CP is called in the CPU of the I-device for the subordinate PROFenergy devices.

### System and program blocks for transferring data records

The PROFenergy commands and status information between IO controller and IO device are exchanged by reading and writing data records. This is implemented using the program blocks RDREC and RWREC.

The PROFenergy data records are described below along with the response data of the individual program blocks.

#### Note

**Block calls**

PE_START_END_CP, PE_CMD_CP, PE_I_DEV_CP and DS3_WRITE_CP must not be called at the same time. The next program block can only be called after one of these program blocks as signaled "no error" (VALID = 1) or "error" (ERROR = 1).

The program block PNIO_RW_REC must also not be called at the same time as PE_START_END_CP, PE_CMD_CP, PE_I_DEV_CP or DS3_WRITE_CP.
3.3.2 PE_START_END_CP

3.3.2.1 Meaning and call - PE_START_END_CP

Significance and how it works

PE_START_END_CP can be used as an alternative to PE_CMD_CP.

PE_START_END_CP is used on the IO controller. It triggers an energy saving pause or ends a pause on the assigned PROFIenergy device.

The program block can be used ideally on IO controllers with IO devices that have only field devices connected to them and no energy data needs to be or can be read out from them.

The energy-saving modes are configured in the user program of the IO device. The energy-saving mode actually adopted is reported back by the IO device after execution of PE_START_END_CP and output at the PE_MODE_ID parameter.

The Pause_Time parameter specifies the length of the energy-saving pause for the IO device. On the IO device, the PE_I_DEV_CP program block checks whether or not the specified duration of the pause is adequately long and can be implemented.

Sequence

Image 3-1 Flow chart of the write/read jobs of PE_START_END_CP and PE_CMD_CP
Using WRREC, PE_START_END_CP sends a PROFIenergy command as a write job to the IO device. Following this, PE_START_END_CP waits for the acknowledgment from the IO device. To achieve this, the acknowledgment data record is read every 100 milliseconds using the program block RDREC.

As long as no acknowledgment has arrived from the IO device, the read job is repeated for 10 seconds at intervals of 100 ms.

The response data of the IO device is read with RDREC.

**Call interface in FBD representation**

```plaintext
PE_START_END_CP

WORD CPLADDR PE_MODE_ID BYTE
BOOL START VALID BOOL
BOOL END BUSY BOOL
WORD ID ERROR BOOL
TIME PAUSE_TIME STATUS WORD
```

**Call interface in STL representation**

```plaintext
STL Explanation

call fb 85 ( //Call program block PE_START_END_CP;
CPLADDR :=W#16#0100, //Module address from the hardware configuration;
START :=M100.0, //Address for "Start of pause" signal;
END :=M100.1, //Address for "End of pause" signal;
ID :=W#16#110, //Address of the destination device;
PAUSE_TIME :=T#10S //Specification of the pause time as IEC time;
VALID :=M100.2 //Address for VALID return parameter;
BUSY :=M110.0, //Address for BUSY return parameter;
ERROR :=M110.1, //Address for ERROR return parameter;
STATUS :=MW128, //Address for STATUS return parameter;
PE_MODE_ID :=MB111 ); //Address for the ID of the energy-saving mode
```

**3.3.2.2 Explanation of the formal parameters of PE_START_END_CP**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT WORD</td>
<td>I, Q, M, D, L, const.</td>
<td>Module start address of the CP</td>
<td></td>
</tr>
<tr>
<td>START</td>
<td>INPUT BOOL</td>
<td>1 = command active, 0 = command not active</td>
<td>A rising edge enables the &quot;Start_Pause&quot; command</td>
<td></td>
</tr>
</tbody>
</table>
### Program blocks for PROFINET IO (S7-300)

#### 3.3 PROFElenergy

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
</table>
| END       | INPUT       | BOOL      | • 1 = command active  
             • 0 = command not active | A rising edge enables the "End_Pause" command |
| ID        | INPUT       | WORD      |                 | Logical address of the destination PROFElenergy device |
| PAUSE_TIME| INPUT       | TIME      | T#24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS | IEC time in steps of 1 ms, integer with sign |
| PE_MODE_ID| OUTPUT      | BYTE      | • 00h: Power OFF (pause)  
             • 01h...FEh: Configurable  
             • FFh: Ready for operation | ID of the energy-saving mode adopted by the IO device after execution of the command. |
| VALID     | OUTPUT      | BOOL      | 0: -  
             1: Execution completed successfully | This parameter indicates whether or not the job was completed without errors. |
| BUSY      | OUTPUT      | BOOL      | 0: Execution completed, aborted or not yet started  
             1: Execution active | Condition code of the processing status of the program block |
| ERROR     | OUTPUT      | BOOL      | 0: -  
             1: Errors | Error code |
|           |             |           | For the meaning in conjunction with the STATUS parameter, refer to Condition codes of PE_START_END_CP (Page 172). |
| STATUS    | OUTPUT      | WORD      |                 | Status code |
|           |             |           | For the meaning in conjunction with the ERROR parameter, refer to Condition codes of PE_START_END_CP (Page 172). |

#### 3.3.2.3 Condition codes of PE_START_END_CP

**Condition codes of PE_START_END_CP**

PE_START_END_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

The following additional PROFElenergy-specific condition codes are output. The error codes of STATUS are valid only in conjunction with ERROR = 1.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Block-specific errors</strong></td>
<td></td>
</tr>
<tr>
<td>8080h</td>
<td>Rising edge at START and END at the same time</td>
</tr>
<tr>
<td>8081h</td>
<td>Length conflict between CMD_PARAM and CMD_PARAM_LEN</td>
</tr>
<tr>
<td><strong>PROFIenergy-specific errors</strong></td>
<td></td>
</tr>
<tr>
<td>FE01h</td>
<td>Invalid Service_Request_ID</td>
</tr>
</tbody>
</table>

---

Program blocks for SIMATIC NET S7 CPs
Programming Manual, 11/2015, C79000-G8976-C229-08
### 3.3.3 PE_CMD_CP

#### 3.3.3.1 Meaning and call - PE_CMD_CP

**Significance and how it works**

PE_CMD_CP can be used as an alternative to PE_START_END_CP.

PE_CMD_CP is used on the IO controller and initiates an energy-saving pause or ends a pause on the assigned PROFIenergy device. PE_CMD_CP can also read out further information and energy measured values from an IO device.

The program block can be used ideally on IO controllers with IO devices that have field devices connected to them and energy data needs to be read out from them.

You will find a flowchart of the write/read jobs of PE_CMD_CP in section Meaning and call - PE_START_END_CP (Page 170).

The individual commands that can be transferred to the IO device with the program block are assigned defined “Service_Request_IDs”. The Service_Request_IDs 01...05 and 16 are assigned in the CMD parameter.

The CMD_MODIFIER parameter specifies the two commands 04 (Query_Modes) and 16 (Query_Measurement) in greater detail.

---

**STATUS**

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE02h</td>
<td>Invalid Request_Reference</td>
</tr>
<tr>
<td>FE03h</td>
<td>Invalid CMD_MODIFIER</td>
</tr>
<tr>
<td>FE04h</td>
<td>Invalid information about the data structure of a command (Data_Structure_Identifier_RQ) in the frame for writing the PROFIenergy data record</td>
</tr>
<tr>
<td>FE05h</td>
<td>Invalid information about the data structure of a command (Data_Structure_Identifier_RS) in the frame for reading the PROFIenergy data record</td>
</tr>
<tr>
<td>FE06h</td>
<td>Energy saving mode (PE_Mode_ID) not supported</td>
</tr>
<tr>
<td>FE07h</td>
<td>Response longer than max transfer length</td>
</tr>
<tr>
<td>FE08h</td>
<td>Invalid number of commands</td>
</tr>
<tr>
<td>FE09h</td>
<td>Invalid block type (see frame header)</td>
</tr>
<tr>
<td>FE0Ah</td>
<td>Invalid block length (see frame header)</td>
</tr>
<tr>
<td>FE0Bh</td>
<td>Invalid block version (see frame header)</td>
</tr>
<tr>
<td>FE50h</td>
<td>Not a suitable energy saving mode (PE_Mode_ID)</td>
</tr>
<tr>
<td>FE51h</td>
<td>Value for PAUSE_TIME not supported</td>
</tr>
<tr>
<td>FE52h</td>
<td>PE_Mode_ID not supported</td>
</tr>
</tbody>
</table>

Details on the parameters of the PROFIenergy-specific errors can be found in the section Response data (Page 179).

**See also**

Condition codes of PNIO_RW_REC (Page 161)
The CMD_PARA parameter assigns the values for certain parameters to some commands using an Any pointer. The CMD_PARA_LEN parameter specifies the length of this parameter.

The RESPONSE_DATA parameter points to the data area of the response data of the IO device.

Call interface in FBD representation

```
PE_CMD_CP

WORD   CPLADDR   VALID   BOOL
BOOL    REQ      BUSY    BOOL
WORD    ID       ERROR   BOOL
BYTE    CMD      STATUS  WORD
BYTE    CMD_MODIFIER
ANY     CMD_PARA
INT     CMD_PARA_LEN
ANY     RESPONSE_DATA
```

Call interface in STL representation

```
STL                     Explanation
call fb 86 (             //Call program block PE_CMD_CP;
CPLADDR :=W#16#0100,    //Module address from the hardware configuration;
REQ :=M220.0,          //Address for edge signal for block execution;
ID :=W#16#110,         //Address of the destination device;
CMD :=MB222,           //Service_Request_ID of the PROFIenergy command;
CMD_MODIFIER :=MB224,   //Modifier of the PROFIenergy command;
CMD_PARA :=MD240,       //Pointer to parameter of the modifier;
CMD_PARA_LEN :=MW226,   //Length of the parameter of CMD_PARA;
VALID :=M220.2          //Address for VALID return parameter;
BUSY :=M220.1,          //Address for BUSY return parameter;
ERROR :=M220.3,         //Address for ERROR return parameter;
STATUS :=MW228,         //Address for STATUS return parameter;
RESPONSE_DATA           //Address for the response data of the IO device
:=P#DB400.DBX0.0 BYTE 244 );
```

See also

Explanation of the formal parameters of PE_CMD_CP (Page 175)
3.3.3.2 **Explanation of the formal parameters of PE_CMD_CP**

### Explanation of the formal parameters of PE_CMD_CP

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>I, Q, M, D, L, const.</td>
<td>Module start address of the CP</td>
</tr>
<tr>
<td>REQ</td>
<td>INPUT</td>
<td>BOOL</td>
<td></td>
<td>Starts the transfer of the PROFlenergy commands on a rising edge.</td>
</tr>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Logical address of the destination PROFlenergy device</td>
</tr>
<tr>
<td>CMD</td>
<td>INPUT</td>
<td>BYTE</td>
<td>• 01: Start_Pause</td>
<td>Service_Request_ID of the PROFlenergy command. You will find the meaning of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 02: End_Pause</td>
<td>the commands below this table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 03: Query_Modes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 04: PEM_Status</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 05: PE_Identity</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 16: Query_Measurement</td>
<td></td>
</tr>
<tr>
<td>CMD.MODIFIER</td>
<td>INPUT</td>
<td>BYTE</td>
<td>For &quot;Start_Pause&quot;: 00</td>
<td>Modifier of the PROFlenergy command, meaning:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For &quot;End_Pause&quot;: 00</td>
<td>• &quot;Query_Modes&quot; command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For &quot;Query_Modes&quot;:</td>
<td>– Modifier 01: Reads all supported energy-saving modes (PE_Mode_ID)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 01: List_Energy_Saving_Modes</td>
<td>– Modifier 02 reads the parameters of the selected PE_Mode_ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 02: Get_Mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For &quot;PE_Identity&quot;: 00</td>
<td>• &quot;Query_Measurement&quot; command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For &quot;Query_Measurement&quot;:</td>
<td>– Modifier 01: Reads the configured Measurement_IDS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 01: Get_Measurement_List</td>
<td>– Modifier 02: Reads the measured values of the selected Measurement_IDS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 02: Get_Measurement_Values</td>
<td></td>
</tr>
</tbody>
</table>

Modifier 00 means "no options".
### 3.3 PROFenergy

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMD_PARA</td>
<td>INPUT</td>
<td>ANY</td>
<td></td>
<td>Any pointer to parameters for commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• For command 01 Start_Pause: &quot;Pause_Time&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• For command 02 End_Pause: Irrelevant</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• For command 03 Query_Modes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• For modifier 01: Irrelevant</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• For modifier 02 Get_Mode: &quot;PE_Mode_ID&quot;</td>
</tr>
<tr>
<td>CMD_PARA</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>Actual length of the parameters in CMD_PARA. Max. length: 234 bytes</td>
</tr>
<tr>
<td>RESPONSE</td>
<td>INOUT</td>
<td>ANY</td>
<td></td>
<td>Pointer to the address of the response data of the IO device (complete frame including block header)</td>
</tr>
<tr>
<td>DATA</td>
<td></td>
<td></td>
<td></td>
<td>Note: If the area selected is not large enough, only the configured number of bytes is saved.</td>
</tr>
<tr>
<td>VALID</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Execution completed successfully</td>
<td>The status parameter of the program block indicates whether or not the job was completed without errors.</td>
</tr>
<tr>
<td>BUSY</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: Execution not yet started, completed or aborted 1: Execution active</td>
<td>Condition code of the processing status of the program block</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Errors</td>
<td>Error code For the meaning in conjunction with the STATUS parameter, refer to Condition codes of PE_CMD_CP (Page 178).</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td></td>
<td>Status code For the meaning in conjunction with the ERROR parameter, refer to Condition codes of PE_CMD_CP (Page 178).</td>
</tr>
</tbody>
</table>
Service_Request_IDs and meaning of the PROFIenergy commands

The PROFIenergy commands with Service_Request_ID 01...05 and 16 have the following significance:

- **01 = Start_Pause**
  Command for starting an energy-saving pause.
  The IO device selects the configured energy-saving mode. The energy-saving mode is reported back to the controller in the response data.

- **02 = End_Pause**
  Command for ending an energy saving pause

- **03 = Query_Modes**
  Queries the configured energy-saving modes with all corresponding time and energy information on the IO device.
  The queried information is detailed using the CMD_MODIFIER parameter:
  - **List_Energy_Saving_Modes**
    Reads all supported PROFIenergy modes of the IO device.
  - **Get_Mode**
    Reads the data of the selected PROFIenergy mode.

- **04 = PEM_Status**
  Query of the energy-saving mode actually adopted by the field device or the unit group.

- **05 = PE_Identity**
  Queries the PROFIenergy services supported by the IO device.

- **16 = Query_Measurement**
  Queries the energy data of the IO device.
  The queried information is detailed using the CMD_MODIFIER parameter:
  - **Get_Measurement_List**
    Reads all the configured Measurement_IDs on the device.
  - **Get_Measurement_Values**
    Reads the measured energy values of the selected Measurement_IDs.
### Commands for various device classes

The devices that can be included in PROFIenergy concepts can be divided into three classes that are addressed by the IO controller with different commands:

- **IO modules, actuators, motor starters**
  - Supported commands:
    - Start_Pause, End_Pause
    - Query_Modes, PEM_Status, PE_Identify
- **Measuring devices for electrical variables**
  - Supported commands:
    - Query_Measurement
- **Frequency converters**
  - Supported commands:
    - Start_Pause, End_Pause
    - Query_Modes, PEM_Status, PE_Identify
    - Query_Measurement
  
  Data of electrical variables data acquired by frequency converters can also be queried.

### Condition codes of PE_CMD_CP

**Condition codes of PE_CMD_CP**

PE_CMD_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

The following additional PROFIenergy-specific condition codes are output. The error codes of STATUS are valid only in conjunction with ERROR = 1.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block-specific errors</td>
<td></td>
</tr>
<tr>
<td>8081h</td>
<td>Length conflict between CMD_PARAM and CMD_PARAM_LEN</td>
</tr>
<tr>
<td>PROFIenergy-specific errors</td>
<td></td>
</tr>
<tr>
<td>FE01h</td>
<td>Invalid Service_Request_ID</td>
</tr>
<tr>
<td>FE02h</td>
<td>Invalid Request_Reference</td>
</tr>
<tr>
<td>FE03h</td>
<td>Invalid CMD_MODIFIER</td>
</tr>
<tr>
<td>FE04h</td>
<td>Invalid information on the data structure of a command (Data_Structure_Identifier_RQ) in the frame for the PROFIenergy data record to be written</td>
</tr>
<tr>
<td>FE05h</td>
<td>Invalid information on the data structure of a command (Data_Structure_Identifier_RS) in the frame for the PROFIenergy data record to be read</td>
</tr>
<tr>
<td>FE06h</td>
<td>Energy saving mode (PE_Mode_ID) not supported</td>
</tr>
</tbody>
</table>
Details on the parameters of the PROFIenergy-specific errors can be found in the section Response data (Page 179).

See also

Condition codes of PNIO_RW_REC (Page 161)

3.3.4 Response data

Structure of the response data

The following tables show the structure of the data record (80A0h) of the response data of PE_START_END_CP and PE_CMD_CP.

The following table shows an overview of the structure of the data record of the returned response data according to the PROFIenergy specification. The composition of the "Service Data Response" area is described below for the individual PROFIenergy commands.

Table 3-6 Structure of the response data

<table>
<thead>
<tr>
<th>Block definitions</th>
<th>Attributes</th>
<th>Value</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block header</td>
<td>BlockType</td>
<td>0801h</td>
<td>Unsigned16</td>
<td>Frame length (without the &quot;BlockType&quot; and &quot;BlockLength&quot; fields)</td>
</tr>
<tr>
<td></td>
<td>BlockLength</td>
<td></td>
<td>Unsigned16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BlockVersionHigh</td>
<td>01h</td>
<td>Unsigned8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BlockVersionLow</td>
<td>00h</td>
<td>Unsigned8</td>
<td></td>
</tr>
</tbody>
</table>
### Block definitions

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Response header</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service_Request_ID</td>
<td>01h...FFh</td>
<td>Unsigned8</td>
<td>01h: Start_Pause 02h: End_Pause 03h: Query_Modes 04h: PEM_Status 05h: PE_Identify 06h...09h: Reserved 10h: Query_Measurement 11h...CF: Reserved D0h...FFh: Vendor-specific</td>
</tr>
<tr>
<td>Request_Reference</td>
<td>01h...FFh</td>
<td>Unsigned8</td>
<td>Identification number of the query (mirrored in the response of the IO device)</td>
</tr>
<tr>
<td><strong>Service header response</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>01h...FFh</td>
<td>Unsigned8</td>
<td>00h: Reserved 01h: Done 02h: Done with error(s) 03h: Data incomplete 04h...CFh: Reserved D0h...FFh: Depends on the Service_Request_ID</td>
</tr>
<tr>
<td>Data_Structure_Identifier_RS</td>
<td>01h...FFh</td>
<td>Unsigned8</td>
<td>00h: Reserved 01h...FFh: Data structure dependent on the Service_Request_ID FFh: error</td>
</tr>
<tr>
<td><strong>Service data response</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Response data of the IO device Depending on the particular PROFIenergy command (described below)</td>
</tr>
</tbody>
</table>

### Meaning of "Service data request" and "Service data response"

The following sections explain the parameter values for the queries of the IO controller to the IO device (Service Data Request) for each PROFIenergy command and the structure of the response data of the IO device (Service Data Response).

- **Service data request**
  - Parameter values for IO controller queries

- **Service data response**
  - Structure of the response data of the IO device
PROFIenergy command "Start_Pause"

- **Service data request**
  - CMD = 01
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 04
  - CMD_PARA = Any pointer to the value for "Pause_Time" (data type "TIME")
    - IEC time in steps of 1 ms, integer with sign
    - Value: T#-24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS

- **Service data response**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE_Mode_ID *</td>
<td>01h...FFh</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>- Reserved -</td>
<td>00h</td>
<td>Unsigned8</td>
</tr>
</tbody>
</table>

* Identification number of the energy-saving mode

PROFIenergy command "End_Pause"

- **Service data request**
  - CMD = 02
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant

- **Service data response**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time_to_operate *</td>
<td></td>
<td>Unsigned32</td>
</tr>
</tbody>
</table>

* Expected time for switching over the PROFIenergy device to "ready to operate"

PROFIenergy command "Query_Modes" – List_Energy_Saving_Modes

- **Service data request**
  - CMD = 03
  - CMD_MODIFIER = 01
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant

- **Service data response**
### PROFInergy

#### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number_of_PE_Mode_IDs *</td>
<td>01&lt;sub&gt;h&lt;/sub&gt;</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>PE_Mode_IDs</td>
<td></td>
<td>Unsigned8 array of Number_of_PE_Mode_IDs (unique ID for mode)</td>
</tr>
</tbody>
</table>

* Number of energy-saving modes

#### PROFInergy command "Query_Modes" – Get_Mode

- **Service data request**
  - CMD = 03
  - CMD_MODIFIER = 02
  - CMD_PARA_LEN = 01
  - CMD_PARA = Any pointer to value for PE_MODE_ID (unsigned8)

- **Service data response**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE_Mode_ID</td>
<td>01&lt;sub&gt;h&lt;/sub&gt;...FF&lt;sub&gt;h&lt;/sub&gt;</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>PE_Mode_Attributes *</td>
<td>00&lt;sub&gt;h&lt;/sub&gt;...01&lt;sub&gt;h&lt;/sub&gt;</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Time_min_Pause</td>
<td></td>
<td>Unsigned32</td>
</tr>
<tr>
<td>Time_to_Pause</td>
<td></td>
<td>Unsigned32</td>
</tr>
<tr>
<td>Time_to_operate</td>
<td></td>
<td>Unsigned32</td>
</tr>
<tr>
<td>Time_min_length_of_stay</td>
<td></td>
<td>Unsigned32</td>
</tr>
<tr>
<td>Time_max_length_of_stay</td>
<td></td>
<td>Unsigned32</td>
</tr>
<tr>
<td>Mode_Power_Consumption</td>
<td></td>
<td>Float32</td>
</tr>
<tr>
<td>Energy_Consumption_to_pause</td>
<td></td>
<td>Float32</td>
</tr>
<tr>
<td>Energy_Consumption_to_operate</td>
<td></td>
<td>Float32</td>
</tr>
</tbody>
</table>

* Coding of bit 0:
  0 = Only static time and energy measured values available.
  1 = Dynamic time and energy measured values available.

 Bits 1...7: Reserved

#### PROFInergy command "PEM_Status"

- **Service data request**
  - CMD = 04
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant

- **Service data response**
### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE_Mode_ID_Source</td>
<td>00h, 01h...FEh, FFh</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>PE_Mode_ID_Destination</td>
<td>00h, 01h...FEh, FFh</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Time_to_operate</td>
<td></td>
<td>Unsigned32</td>
</tr>
<tr>
<td>Remaining_time_to_destination</td>
<td></td>
<td>Unsigned32</td>
</tr>
<tr>
<td>Mode_Power_Consumption</td>
<td></td>
<td>Float32</td>
</tr>
<tr>
<td>Energy_Consumption_to_Destination</td>
<td></td>
<td>Float32</td>
</tr>
<tr>
<td>Energy_Consumption_to_operate</td>
<td></td>
<td>Float32</td>
</tr>
</tbody>
</table>

* Possible values for "PE_Mode_ID_Source" and "PE_Mode_ID_Destination":
  - 00h: PE_Power_off
  - 01h...FEh: Freely configurable
  - FFh: PE_Ready_to_operate

### PROFIdenergy command "PE_Identify"

- **Service data request**
  - CMD = 05
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant

- **Service data response**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count *</td>
<td>6</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Start_Pause **</td>
<td>01h</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>End_Pause</td>
<td>02h</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Query_Modes</td>
<td>03h</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>PEM_Status</td>
<td>04h</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>PE_Identify</td>
<td>05h</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Query_Measurement ***</td>
<td>10h</td>
<td>Unsigned8</td>
</tr>
</tbody>
</table>

* Number of supported PROFIdenergy commands

** Service_Request_ID of the first supported PROFIdenergy command

*** Service_Request_ID of the last supported PROFIdenergy command
3.3 PROFInergy

PROFInergy command "Query_Measurement" – Get_Measurement_List

- **Service data request**
  - CMD = 16
  - CMD_MODIFIER = 01
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant

- **Service data response**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count *</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>- Reserved -</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Measurement_ID **</td>
<td></td>
<td>Unsigned16</td>
</tr>
<tr>
<td>Accuracy_Domain 1</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Accuracy_Class 2</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Range 3</td>
<td></td>
<td>Float32</td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measurement_ID ***</td>
<td></td>
<td>Unsigned16</td>
</tr>
<tr>
<td>Accuracy_Domain 1</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Accuracy_Class 2</td>
<td></td>
<td>Unsigned8</td>
</tr>
</tbody>
</table>
| Range 3                     |        | Float32     |}

* Number of Measurement_IDs
** First supported Measurement_ID
*** Last supported Measurement_ID

1 Accuracy domain (range 1...4):
  - 0 = Reserved
  - 1 = Percentage of the measuring range
  - 2 = Percentage of the current measured values
  - 3 = Accuracy according to IEC 61557-12
  - 4 = Accuracy according to EN 50470-3 section 8

2 Accuracy class (range 1...15):
  - 0 = Reserved
  - 1 (0.01%) ... 15 (>20%)

3 Measuring range if Accuracy_Domain = 1; otherwise undefined

PROFInergy command "Query_Measurement" – Get_Measurement_Values

- **Service data request**
  - CMD = 16
  - CMD_MODIFIER = 02
  - CMD_PARA_LEN = length of the data structure in bytes
  - CMD_PARA = Any pointer to data structure with the following structure:
### 3.3 PROFienergy

#### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count *</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>- Reserved -</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Measurement_ID **</td>
<td></td>
<td>Unsigned16</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measurement_ID ***</td>
<td></td>
<td>Unsigned16</td>
</tr>
</tbody>
</table>

* Number of Measurement IDs  
** First queried measured value  
*** Last queried measured value

---

### Service data response

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count *</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>- Reserved -</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Length_of_Structure 0002h...FFFFh</td>
<td></td>
<td>Unsigned16</td>
</tr>
<tr>
<td>Measurement_Data_Structure_ID 1 = simple value</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Measurement_ID ** 00h...FFh</td>
<td></td>
<td>Unsigned16</td>
</tr>
<tr>
<td>Status_of_Measurement_Value 1 = valid 2 = not available 3 = not available at times</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Transmission_Data_Type</td>
<td></td>
<td>Float32</td>
</tr>
<tr>
<td>End_of_demand</td>
<td></td>
<td>Unsigned32 or Unsigned16</td>
</tr>
<tr>
<td>Length_of_Structure</td>
<td></td>
<td>Unsigned16</td>
</tr>
<tr>
<td>Measurement_Data_Structure_ID</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Measurement_ID ***</td>
<td></td>
<td>Unsigned16</td>
</tr>
<tr>
<td>Status_of_Measurement_Value</td>
<td></td>
<td>Unsigned8</td>
</tr>
<tr>
<td>Transmission_Data_Type</td>
<td></td>
<td>Float32</td>
</tr>
<tr>
<td>End_of_demand</td>
<td></td>
<td>Unsigned32 or Unsigned16</td>
</tr>
</tbody>
</table>

* Number of Measurement IDs  
** First queried measured value  
*** Last queried measured value
3.3.5 PE_I_DEV_CP

3.3.5.1 Meaning and call - PE_I_DEV_CP

Significance and how it works

The program block PE_I_DEV_CP is used on the PROFIenergy device where it handles the PROFIenergy commands of the IO controller. The PROFIenergy data records (80A0h) sent by the IO controller are forwarded by the CP firmware to PE_I_DEV_CP. The PROFIenergy data of the IO device is made available to the IO controller as the response by PE_I_DEV_CP using the PROFIenergy data record (80A0h).

The response data of PE_I_DEV_CP is generated by the supplementary functions FC 0 to FC 8, see section Supplementary program blocks for PE_I_DEV_CP (Page 189).

Call interface in FBD representation

```
PE_I_DEV_CP

WORD

CPLADDR    CMD

BOOL

RESET      CMD_MODIFIER

BOOL

VALID      CMD_PARA

INDEX

INT

NEW        BOOL

ERROR      BOOL

STATUS     WORD

RESPONSE_DATA
```

Call interface in STL representation

```
call fb 87 (  //Call program block PE_I_DEV_CP;
CPLADDR :=W#16#0100,
RESET :=M1.0,  //Signal for resetting the block;
VALID :=M1.3,  //Signal for data transfer to the controller;
CMD :=MW222,   //Service_Request_ID of the PROFIenergy command;
CMD_MODIFIER :=MW224,  //Modifier of the PROFIenergy command;
CMD_PARA :=MD230,   //Pointer to parameters of the modifier;
INDEX :=MW228,    //Number of the PROFIenergy data record;
NEW :=M1.1,       //Processing status of the block;
ERROR :=M1.2,     //Address for the ERROR return parameter;
STATUS :=MW2,     //Address for the STATUS return parameter;
RESPONSE_DATA :=P#DB400.DBX0.0 BYTE 244 );  //Address for the response data of the IO device
```

### 3.3.5.2 Explanation of the formal parameters of PE_I_DEV_CP

#### Explanation of the formal parameters of PE_I_DEV_CP

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>I, Q, M, D, L, const.</td>
<td>Module start address of the CP</td>
</tr>
<tr>
<td>RESET</td>
<td>INPUT</td>
<td>BOOL</td>
<td></td>
<td>Resets processing of the program block. NEW is set to 0.</td>
</tr>
<tr>
<td>VALID</td>
<td>INPUT</td>
<td>BOOL</td>
<td></td>
<td>If the response data was written to the relevant memory area of the IO device, VALID = 1 must be set by the user program. Following this, the program block makes the data available to the IO controller. NEW is set to 0.</td>
</tr>
<tr>
<td>CMD</td>
<td>OUTPUT</td>
<td>INT</td>
<td>• 01: Start_Pause</td>
<td>Service ID of the PROFInergy command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 02: End_Pause</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 03: Query_Modes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 04: PEM_Status</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 05: PE_Identity</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 16: Query_Measurement</td>
<td></td>
</tr>
<tr>
<td>CMD_MODIFIER</td>
<td>OUTPUT</td>
<td>INT</td>
<td>• Modifier for Start_Pause: 00</td>
<td>Modifier of the PROFInergy commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Modifier for End_Pause: 00</td>
<td>Meaning of the modifiers for commands:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Query_Modes, Modifier:</td>
<td>• &quot;Query_Modes&quot; command, Modifier:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 01 (List_Energy_Saving_Modes)</td>
<td>– 01 (List_Energy_Saving_Modes): Reads all supported PROFInergy modes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 02 (Get_Mode)</td>
<td>– 02 (Get_Mode): Reads the data of the selected PROFInergy mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Modifier for PEM_Status: 00</td>
<td>• &quot;Query_Measurement&quot; command, Modifier:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Modifier for PE_Identity: 00</td>
<td>– 01 (Get_Measurement_List): Reads all configured Measurement_IDs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Query_Measurement, Modifier:</td>
<td>– 02 (Get_Measurement_Values): Reads the measured values of the selected Measurement_IDs.</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMD_PARA</td>
<td>OUTPUT</td>
<td>ANY</td>
<td></td>
<td>Any pointer to parameters for the following command modifiers (compare CMD_MODIFIER parameter):</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• For “Get_Mode”:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE_Mode_ID (ID of the energy-saving mode) length = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• For “Get_Measurement_Values”: measured values of the Measurement_IDs length = max. 236 bytes (complete frame of the controller command without header)</td>
</tr>
<tr>
<td>INDEX</td>
<td>OUTPUT</td>
<td>INT</td>
<td></td>
<td>Number of the PROFIenergy data record (80A0h)</td>
</tr>
<tr>
<td>NEW</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: Execution not yet started, completed or aborted</td>
<td>Condition codes of the processing status of the program block</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Errors</td>
<td>Error code</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td></td>
<td>Status code</td>
</tr>
<tr>
<td>RESPONSE _DATA</td>
<td>INOUT</td>
<td>ANY</td>
<td>See &quot;Response data&quot; of the program block</td>
<td>Pointer to the data area of the response of the IO device (complete response frame including header).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The data area must match the data area of the supplementary program blocks FC 0 - FC 8 (parameter &quot;DATA_ERRORRSP&quot;).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Recommended size: At least 244 bytes. If the data area is too small, only the data of the configured bytes are transferred.</td>
</tr>
</tbody>
</table>

### 3.3.5.3 Condition codes of PE_I_DEV_CP

#### Condition codes of PE_I_DEV_CP

PE_I_DEV_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

### See also

Condition codes of PNIO_RW_REC (Page 161)
3.3.6 Supplementary program blocks for PE_I_DEV_CP

3.3.6.1 Overview of the FCs

Function

The supplementary program blocks FC 0 to FC 8 support the preparation of the response data made available to the controller by PE_I_DEV_CP:

- For the response data of each PROFenergy command, there is a separate program block (FC 1 - FC 8).
- FC 0 generates a common negative response for all PROFenergy commands.

The FCs are called in the user program. In STEP 7 V5.5, they are available in the standard library in the "PROFenergy" folder.

The FCs have several common parameters as well as individual parameters. Some of the common parameters of the FCs are interconnected with parameters of PE_I_DEV_CP. With some of the individual input parameters of the FCs, the response data is entered as plain language for the user or stored in the memory area of the IO device.

Overview of the FCs

The following supplementary program blocks are made available:

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC 0</td>
<td>PE_ERROR_RSP</td>
</tr>
<tr>
<td>FC 1</td>
<td>PE_START_RSP</td>
</tr>
<tr>
<td>FC 2</td>
<td>PE_END_RSP</td>
</tr>
<tr>
<td>FC 3</td>
<td>PE_LIST_MODES_RSP</td>
</tr>
<tr>
<td>FC 4</td>
<td>PE_GET_MODE_RSP</td>
</tr>
<tr>
<td>FC 5</td>
<td>PE_PEM_STATUS_RSP</td>
</tr>
<tr>
<td>FC 6</td>
<td>PE_IDENTIFY_RSP</td>
</tr>
<tr>
<td>FC 7</td>
<td>PE_MEASUREMENT_LIST_RSP</td>
</tr>
<tr>
<td>FC 8</td>
<td>PE_MEASUREMENT_VALUE_RSP</td>
</tr>
</tbody>
</table>

See also

Individual parameters of the FCs (Page 191)
3.3.6.2 Interconnection of the FCs with PE_I_DEV_CP

Interconnection of the FCs with the program block PE_I_DEV_CP

![Diagram of PE_I_DEV_CP and FC 0...FC 8 interconnection](image3-2)

* Evaluation and reaction by user
** Entry by user

**Note**

Interconnection of the program blocks is an absolute necessity

PE_I_DEV_CP must be interconnected with FC 0...FC 8 at the parameters shown on a light blue background that are assigned to the corresponding parameters of the FCs indicated by red arrows.
### 3.3.6.3 Common parameters of the FCs

#### Common parameters of the supplementary program blocks FC 0 - FC 8

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Input parameters</strong></td>
</tr>
<tr>
<td>ACTIVATE</td>
<td>INPUT</td>
<td>BOOL</td>
<td></td>
<td>Instructs the block to copy the input parameters to the &quot;DATA_ERRORRSP&quot; data area are on a rising edge. Is then reset by the block. Must be set by the user within 10 seconds after a positive edge was detected at PE_I_DEV_NEW.</td>
</tr>
<tr>
<td>PE_I_DEV_NEW</td>
<td>INPUT</td>
<td>BOOL</td>
<td></td>
<td>Must be interconnected with the NEW output parameter of PE_I_DEV_CP. The block is processed only when 1 is set.</td>
</tr>
<tr>
<td>CMD</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>Must be interconnected with the CMD output parameter of PE_I_DEV_CP.</td>
</tr>
<tr>
<td>CMD_MODIFIER</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>Must be interconnected with the CMD_MODIFIER output parameter of PE_I_DEV_CP.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Output parameters</strong></td>
</tr>
<tr>
<td>DATA_ERRORRSP</td>
<td>OUTPUT</td>
<td>ANY</td>
<td></td>
<td>Pointer to the data area in which the response data will be stored (complete response frame including header). Must be interconnected with the RESPONSE_DATA output parameter of PE_I_DEV_CP. Recommended size: At least 244 bytes.</td>
</tr>
<tr>
<td>VALID</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: No error</td>
<td>Is set by the block. Must be interconnected with the VALID input parameter of PE_I_DEV_CP.</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: No error 1: Errors</td>
<td>Error code</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>0: No error</td>
<td>Status code 80B1h: Error in ANY information (for example wrong area)</td>
</tr>
</tbody>
</table>

### 3.3.6.4 Individual parameters of the FCs

#### Individual parameters of FC 0 to FC 8

Below you will find a description of the individual parameters of the FCs.
3.3 PROFIenergy

**PE_ERROR_RSP**

Generates a negative response if the required PROFIenergy command is generally or temporarily not supported. The negative response is not dependent on the requesting command.

Table 3-8 Individual parameters of FC 0 PE_ERROR_RSP

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_CODE</td>
<td>INPUT</td>
<td>BYTE</td>
<td>E</td>
<td>Error number</td>
</tr>
</tbody>
</table>

**PE_START_RSP**

Initiates an energy saving pause. Generates the response to the "Start_Pause" command. Returns the energy-saving mode adopted by the device.

Table 3-9 Individual parameters of FC 1 PE_START_RSP

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE_Mode_ID</td>
<td>INPUT</td>
<td>BYTE</td>
<td>ID</td>
<td>ID of the energy-saving mode that the device or the unit group adopts.</td>
</tr>
</tbody>
</table>

Return message with the PE_Mode_ID of the energy-saving mode that the field devices or the unit group have adopted.

**PE_END_RSP**

Generates the response to the "End_Pause" command.

Table 3-10 Individual parameters of FC 2 PE_END_RSP

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time_to_Operate</td>
<td>INPUT</td>
<td>DWORD</td>
<td></td>
<td>Time required to change from the current energy-saving mode after &quot;ready to operate&quot;.</td>
</tr>
</tbody>
</table>

**PE_LISTModes_RSP**

Generates the response to the "Query_Modes" > modifier "List_Modes" command (list of the supported energy-saving modes).
The IDs of the energy-saving modes must be specified in the user program.

Table 3-11 Individual parameters of FC 3 PE_LIST_MODES_RSP

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number_of_PE_Mode_IDs</td>
<td>INPUT</td>
<td>BYTE</td>
<td></td>
<td>Number of supported energy-saving modes</td>
</tr>
<tr>
<td>PE_Mode_ID</td>
<td>INPUT</td>
<td>ANY</td>
<td>00h, 01h...FEh, FFh</td>
<td>Pointer to the area in which the energy-saving modes are stored. As the user, you will need to store the IDs of the energy-saving modes here. An energy-saving mode ID is configured in the Unsigned8 format. Permitted range: 1 to 254 bytes.</td>
</tr>
</tbody>
</table>

If the devices or a group need to react differently to different lengths of pause you can set up different energy-saving modes (PE_Mode) to achieve this. You assign a different PE_Mode_ID to the various energy-saving modes.

Possible values for "PE_Mode_ID":
- 00h: PE_Power_off
- 01h...FEh: Freely configurable or vendor-specific
- FFh: PE_Ready_to_operate

**PE_GET_MODE_RSP**

Generates the response to the "QueryModes" > Modifier "Get_Mode". command

Table 3-12 Individual parameters of FC 4 PE_GET_MODE_RSP

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE_Mode_ID</td>
<td>INPUT</td>
<td>BYTE</td>
<td></td>
<td>Currently used energy-saving mode ID</td>
</tr>
<tr>
<td>Time_Min_Pause *</td>
<td>INPUT</td>
<td>Unsigned32</td>
<td></td>
<td>Minimum pause duration for this PE energy-saving mode. It is the sum of the three parameters: Time_to_Pause, Time_to_operate, Time_min_length_of_stay</td>
</tr>
<tr>
<td>Time_to_Pause *</td>
<td>INPUT</td>
<td>Unsigned32</td>
<td></td>
<td>Time from the START edge until the requested energy-saving mode is reached</td>
</tr>
<tr>
<td>Time_to_operate *</td>
<td>INPUT</td>
<td>Unsigned32</td>
<td></td>
<td>Max. time after turn on before PE_ready_to_operate. Time_to_operate can be used directly for the relevant calculations. The value can either be a static MAX value or can be calculated dynamically by the PE device.</td>
</tr>
<tr>
<td>Time_min_length_of_stay</td>
<td>INPUT</td>
<td>Unsigned32</td>
<td></td>
<td>Minimum time that the PE device must remain in this PE_Mode.</td>
</tr>
</tbody>
</table>
Parameters | Declaration | Data type | Range of values | Description
---|---|---|---|---
Time_max_length_of_stay * | INPUT | Unsigned32 | | Maximum time that the PE device can remain in this PE_Mode.
Mode_Power_Consumption ** | INPUT | Float32 | | Energy consumption in current PE_Mode [kW]
Energy_Consumption_to_pause ** | INPUT | Float32 | | Energy consumption of PE_ready_to_operate until the current PE_Mode [kWh]
Energy_Consumption_to_operate ** | INPUT | Float32 | | Energy consumption from current PE_Mode until PE_ready_to_operate [kWh]

* The PROFIenergy profile does not specify an invalid time format.
  If the time is unlimited, the maximum value FFFFFFFFh can be specified.
  If the time is zero, 00h can be used.
** If an energy consumption value is not defined, 0.0 (Float32) can be specified.

PE_PEM_STATUS_RSP

Generates the response to the "PEM_STATUS" command.

Table 3-13 Individual parameters of FC 5 PE_PEM_STATUS_RSP

Parameters | Declaration | Data type | Range of values | Description
---|---|---|---|---
PE_Mode_ID_Source | INPUT | BYTE | 00h, 01h...FEh, FFh | ID of the energy-saving mode actually adopted
PE_Mode_ID_Destination | INPUT | BYTE | 00h, 01h...FEh, FFh | ID of the energy-saving mode set by the controller
Time_to_operate * | INPUT | Unsigned32 | | Max. time after turn on before PE_ready_to_operate
  Time_to_operate can be used directly for the relevant calculations. The value can either be a static MAX value or can be calculated dynamically by the PE device.
Remaining_time_to_destination * | INPUT | Unsigned32 | | Optional: Time remaining until the requested PE_Mode. Dynamic value or static MAX value
Mode_Power_Consumption ** | INPUT | Float32 | | Energy consumption in current PE_Mode [kW]
Parameters | Declaration | Data type | Range of values | Description
---|---|---|---|---
Energy_Consumption_to_Destination ** | INPUT | Float32 | | Energy consumption until the requested PE_Mode [kWh]
Energy_Consumption_to_operate ** | INPUT | Float32 | | Energy consumption from current PE_Mode until PE_ready_to_operate [kWh]

* The PROFIenergy profile does not specify an invalid time format.
  If the time is unlimited, the maximum value FFFFFFFFh can be specified.
  If the time is zero, 00h can be used.

** If an energy consumption value is not defined, 0.0 (Float32) can be specified.

Possible values for "PE_Mode_ID_Source" and "PE_Mode_ID_Destination":
- 00h: PE_Power_off
- 01h...FEh: Freely configurable or vendor-specific
- FFh: PE_Ready_to_operate

PE_IDENTIFY_RSP

Generates the response to the "PE_Identify" command
As the user you need to specify which PROFIenergy commands are supported.

Table 3-14 Individual parameters of FC 6 PE_IDENTIFY_RSP

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>INPUT</td>
<td>BYTE</td>
<td>0...6</td>
<td>Meaning of supported PROFIenergy commands</td>
</tr>
</tbody>
</table>
| Start_Pause | INPUT | BOOL | 0...1 | 1: Command is supported
0: Command is not supported |
| End_Pause | INPUT | BOOL | 0...1 | 1: Command is supported
0: Command is not supported |
| Query_Modes | INPUT | BOOL | 0...1 | 1: Command is supported
0: Command is not supported |
| PEM_Status | INPUT | BOOL | 0...1 | 1: Command is supported
0: Command is not supported |
| PEM_Identify | INPUT | BOOL | 0...1 | 1: Command is supported
0: Command is not supported |
| Query_Measurement | INPUT | BOOL | 0...1 | 1: Command is supported
0: Command is not supported |
**PE_MEASUREMENT_LIST_RSP**

Generates the response to the "Query_Measurement" > Modifier "Get_Measurement_List". command

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>INPUT</td>
<td>BYTE</td>
<td></td>
<td>Number of supported measured value IDs (Measurement_ID).</td>
</tr>
<tr>
<td>Measurement_List</td>
<td>INPUT</td>
<td>ANY</td>
<td></td>
<td>Pointer to the data area with the supported measured value IDs. As the user, you store the measured value IDs in this data area. Per frame, a maximum of 29 measured value IDs can be transferred. For information on the structure of the array, refer to section Response data (Page 179) &gt; &quot;Query_Measurement&quot; - Get_Measurement_List.</td>
</tr>
</tbody>
</table>

**PE_MEASUREMENT_VALUE_RSP**

Generates the response to the "Query_Measurement" > Modifier "Get_Measurement_Values". command

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>INPUT</td>
<td>BYTE</td>
<td></td>
<td>Number of supported Measurement_Values</td>
</tr>
<tr>
<td>Measurement_Values</td>
<td>INPUT</td>
<td>ANY</td>
<td></td>
<td>Pointer to the data area of the measured values (Measurement_Values). As the user, you store the measured values in this data area. Per frame, a maximum of 116 measured values can be transferred. For information on the structure of the array, refer to section Response data (Page 179) &gt; &quot;Query_Measurement&quot; - Get_Measurement_List.</td>
</tr>
</tbody>
</table>

**3.3.7 DS3_WRITE_CP / PE_DS3_Write_ET200S_CP**

The following description of the program block DS3_WRITE_CP also applies to PE_DS3_Write_ET200S_CP for STEP 7 Professional that has the same functionality.
3.3.7.1  Meaning and call - DS3_WRITE_CP

Significance and how it works

DS3_WRITE_CP is used in the CPU of the CP 300 as a PROFIenergy controller to transfer the settings for the switching behavior of power modules of an ET 200S. The switching behavior for up to 8 slots (in this case: power modules) can be transferred.

DS3_WRITE_CP is not a PROFIenergy program block.

Call interface in FBD representation

```
DS3_WRITE_CP
```

Call interface in STL representation

```
STL                          Explanation
CALL FB S3, DB 53 (          //Call program block DS3_WRITE_CP;
CPLADDR := W#16#0110,      //Module address from hardware configuration;
ENABLE := M 1.1,           //Address for edge signal for block execution;
ID := W#16#86A,            //Address of the header slot of the IO device;
SLOT_NO_1 := W#16#1000,    //Logical address of power module 1;
FUNC_1 := M 2.1,           //Address of the value of the switching behavior of power module 1;
...                         //...
...                         //...
BUSY := M 1.2,              //Address for the BUSY return parameter;
DONE := M 1.3,              //Address for the DONE return parameter;
ERROR := M 1.4,             //Address for the ERROR return parameter;
STATUS := MW 12 );         //Address for the STATUS return parameter
```
### Explanation of the formal parameters of DS3_WRITE_CP

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>I, Q, M, D, L, const.</td>
<td>Module start address of the CP</td>
</tr>
<tr>
<td>ENABLE</td>
<td>INPUT</td>
<td>BOOL</td>
<td></td>
<td>Starts the processing of the program block on a rising edge.</td>
</tr>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Logical address of the header slot of the IO device</td>
</tr>
<tr>
<td>SLOT_NO_1</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>Slot number of the first power module</td>
</tr>
<tr>
<td>FUNC_1</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>Specifies the switching behavior for the power module in terms of starting or ending energy-saving pauses</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 (FALSE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- PAUSESTART:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No influence (power module remains turned on)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- PAUSESTOP:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Turns the power module on again.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 (TRUE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- PAUSESTART:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Turns the power module off.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- PAUSESTOP:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Turns the power module on again.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Slot number of the eighth power module</td>
</tr>
<tr>
<td>FUNC_8</td>
<td>INPUT</td>
<td>INT</td>
<td></td>
<td>See &quot;FUNC_1&quot;</td>
</tr>
<tr>
<td>BUSY</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: Execution not yet started, completed or aborted 1: Execution active</td>
<td>Condition code of the processing status of the program block</td>
</tr>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Data record transferred successfully</td>
<td>This parameter indicates whether or not the job was completed without errors.</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Errors</td>
<td>Error code</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td></td>
<td>Status code</td>
</tr>
</tbody>
</table>

For the meaning of DONE, ERROR and STATUS, see Condition codes of DS3_WRITE_CP (Page 199).
3.3.7.3 Condition codes of DS3_WRITE_CP

Condition codes of DS3_WRITE_CP

DS3_WRITE_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

See also

Condition codes of PNIO_RW_REC (Page 161)

3.4 Configuration limits / resources required for the program blocks (PROFINET)

Required resources

Note

Note the version information of the blocks. The currently supplied block versions may differ from those shown here. Blocks with other versions have different resource requirements.

You will find information on the current block versions under entry ID:


Table 3-17 Information for FCs / FBs with S7400

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PN_InOut</td>
<td>1.3</td>
<td>FB88</td>
<td>2678</td>
<td>2234</td>
<td>2198</td>
<td>48</td>
</tr>
<tr>
<td>PN_InOut_Fast</td>
<td>1.0</td>
<td>FB90</td>
<td>2906</td>
<td>2266</td>
<td>2230</td>
<td>48</td>
</tr>
</tbody>
</table>
### Table 3-18 Information for FCs / FBs with S7-300

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PN_InOut</td>
<td>1.5</td>
<td>FB88</td>
<td>2470</td>
<td>2066</td>
<td>2030</td>
<td>54</td>
</tr>
<tr>
<td>PNIO_SEND</td>
<td>3.0</td>
<td>FC11</td>
<td>1420</td>
<td>1182</td>
<td>1146</td>
<td>46</td>
</tr>
<tr>
<td>PNIO_RECV</td>
<td>3.0</td>
<td>FC12</td>
<td>1270</td>
<td>1052</td>
<td>1016</td>
<td>46</td>
</tr>
<tr>
<td>PNIO_RW_REC</td>
<td>1.1</td>
<td>FB52</td>
<td>1636</td>
<td>1378</td>
<td>1342</td>
<td>62</td>
</tr>
<tr>
<td>PNIO_ALARM</td>
<td>1.1</td>
<td>FB54</td>
<td>1168</td>
<td>960</td>
<td>924</td>
<td>62</td>
</tr>
<tr>
<td>PE_START_END_CP</td>
<td>1.0</td>
<td>FB85</td>
<td>3286</td>
<td>2808</td>
<td>2772</td>
<td>92</td>
</tr>
<tr>
<td>PE_CMD_CP</td>
<td>1.0</td>
<td>FB86</td>
<td>3750</td>
<td>3264</td>
<td>3228</td>
<td>358</td>
</tr>
<tr>
<td>PE_I_DEV_CP</td>
<td>1.0</td>
<td>FB87</td>
<td>3192</td>
<td>2902</td>
<td>2866</td>
<td>114</td>
</tr>
<tr>
<td>DS3_WRITE_CP</td>
<td>1.0</td>
<td>FB53</td>
<td>1716</td>
<td>1408</td>
<td>1372</td>
<td>84</td>
</tr>
</tbody>
</table>
4.1 PN_InOut / PN_InOut_Fast - meaning and call

Significance and how it works

The task of function block FB88 / FB90 is to transfer data from the interface DB to the CP and from the CP to the interface DB. The interface DB itself is the interface to the user program.

FB88 / FB90 is called cyclically. It is also possible to call FB88 / FB90 more than once in a cycle.

At its interface, FB88 / FB90 only needs to be supplied with the module address of the CP.

To ensure data consistency, you can only modify the data to be transferred or start to read the received data when the job is completed (DONE=1 or ERROR=1).

As soon as DONE=1 or ERROR=1 is set, the transfer is complete or has been terminated with an error message. Data can now be evaluated or set again. Data will only be transferred with the next call.

In your user program, make sure that on completion of data transfer FB88 / FB90 is called again only after all the input data has been read and all output data has been written to the interface DB.

Calling the FB88 / FB90 blocks time-driven is permitted in principle. Refer to the notes on this mode later in this chapter.

Differences between FB88 and FB90

The activities of the function blocks FB90 and FB88 on the interface to the user program are largely identical. You can use FB90 with certain CP/CPU types with an S7-400; refer to the information in the manual of the CP.

If FB90 is approved for the CP type being used, we recommend that you use it. As a result, you can achieve shorter reaction times than with FB88. You should, however, remember the constraints regarding its use.

The following points apply:

- The interface parameters are identical;
- For FB90, there are several additional codes in the STATUS parameter;
• With some errors, there are different codes in the STATUS parameter of FB88 and FB90;
• There are differences in the configuration limits of the interface DB (see relevant manual).

---

**Note**
For more detailed information on handling the interface DB, refer to the SIMATIC iMap documentation.

---

**Note**
When you reload user program blocks, data consistency is only guaranteed if the CPU is first changed to STOP.

---

**How supplied - block library**

FB88 and FB90 are supplied with SIMATIC iMap. There are different block types for S7-300 and S7-400.

After SIMATIC iMap has been installed, the program blocks are available in the STEP 7 library "PROFINET System Library/CP300/Blocks" or "PROFINET System Library/CP400/Blocks".

---

**Call interface**

Call interface in FBD representation

---

![Call interface diagram](image)

---

**Example of a call in STL representation**

```
<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call FB 88 , DB88 (</td>
<td>(//)Block call with instance DB88</td>
</tr>
<tr>
<td>LADDR := W#16#0120,</td>
<td></td>
</tr>
<tr>
<td>DONE := M 99.1,</td>
<td></td>
</tr>
<tr>
<td>ERROR := M 99.0,</td>
<td></td>
</tr>
<tr>
<td>STATUS := MW 104);</td>
<td></td>
</tr>
</tbody>
</table>
```
4.2 Explanation of the formal parameters - PN_InOut / PN_InOut_Fast

Explanation of the formal parameters

The following table explains all the formal parameters for FB88 / FB90:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td>Module start address. When you configure the CP, the module start address is displayed in the configuration table. Specify this address here. Since only one CP in the rack supports operation with PROFINET CBA, this parameter must not be changed.</td>
</tr>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Reports the (positive) completion of a job.</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>This indicates that the job could not be executed errorfree.</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>This parameter supplies detailed information about the execution of the job. Status codes can be returned during execution of the job (DONE=0 and ERROR=0).</td>
</tr>
</tbody>
</table>

4.3 Condition codes of the PN_InOut and PN_InOut_Fast blocks

Evaluating status codes

Remember that the status codes DONE, ERROR, STATUS are updated at each block call.

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Table 4-1 Codes for PN_InOut (FB88) and PN_InOut_Fast (FB90)

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0000H</td>
<td>No job being processed; the block can be called.</td>
</tr>
</tbody>
</table>
| 0    | 0     | 8181H     | • Job active.  
          or  
          • (only with FB90): Connection establishment to addressed module active (see also information under 8090H). |
| 0    | 1     | 8183H     | (S7-300 only)  
The service has not yet started; data acceptance is not yet possible. |
| 0    | 1     | 8184H     | • Bad instance DB, generally triggered by illegal writing of the instance DB by the user program.  
          or  
          • (only with FB90)  
          Bad send or receive job. |
## Condition Codes of the PN_InOut and PN_InOut_Fast Blocks

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8085H</td>
<td>(only with FB90)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bad interface DB.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8090H</td>
<td>(S7-400 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Parameter assignment error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>An incorrect module address was specified; the address points to an</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>empty slot.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note (only with FB90):</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In the following cases, the value 8181H is shown in STATUS (job</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>active); In actual fact, there is no communication:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The address points to a slot that contains a different module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The addressed module is configured for PROFINET CBA operation.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80A1H</td>
<td>(only with FB90)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Possible communications errors:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Station internal connection to addressed module being terminated;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The configuration limits for connections of the CPU has been</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exceeded;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The interface is being reinitialized.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B0H</td>
<td>(S7-300 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Block error: The data record number is wrong.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This status can also occur after the following actions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cold or warm restart after power DOWN/UP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cold or warm restart on the CPU</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B1H</td>
<td>(S7-300 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Block error: Data record length or offset wrong.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B3H</td>
<td>(S7-300 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Parameter error: Wrong CP address.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C1H</td>
<td>(S7-300 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Temporary error: The specified data record is currently being</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C2H</td>
<td>(S7-300 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Temporary error: There is a job bottleneck; the data record cannot</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>be read yet.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C3H</td>
<td>(S7-300 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Temporary error: Resources occupied (memory).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C4H</td>
<td>(S7-300 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Communication error: Occurs temporarily and a repetition in the user</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>program will often remedy the problem.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80D0H</td>
<td>(S7-300 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Configuration error: The maximum number of blocks of input and output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>data has been exceeded; the interface DB is too large.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80D1H</td>
<td>(S7-300 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Configuration error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Possible causes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The interface of the configured component does not match the one</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>used in the program (outputs).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The wrong module was inserted; The PROFINET service is not</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>supported.</td>
</tr>
</tbody>
</table>

Program blocks for PROFINET CBA

4.3 Condition codes of the PN_InOut and PN_InOut_Fast blocks
### 4.3 Condition codes of the PN_InOut and PN_InOut_Fast blocks

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>80D2H</td>
<td>(S7-300 only) Configuration error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Possible causes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The interface of the configured component does not match the one used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>in the program (inputs).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The wrong module was inserted; The PROFINET service is not supported.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8322H</td>
<td>(only with FB90) Bad interface DB.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8332H</td>
<td>(only with FB90) The number of the interface DB is too high.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>833AH</td>
<td>(only with FB90) Access to the interface DB is not possible (possibly</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>because the interface DB was deleted).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8623H</td>
<td>(only with FB90) Bad interface DB.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>863AH</td>
<td>(only with FB90) Access to the interface DB is not possible (possibly</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>because the interface DB was deleted).</td>
</tr>
</tbody>
</table>

To find out which SFCs are used that are relevant for error evaluation, display the properties dialog of the FB described here in the "Calls" tab.

**Note**

For entries with the coding 8FxxH (for S7-300) or 8xxxH (for S7-400) under STATUS, note the information in the Reference Manual STEP 7 Standard and System Functions. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

**Status codes during CP startup**

With a complete restart or restart of the PROFINET CP (after activating a switch on the module), the output parameters of the block are reset as follows:

- DONE = 0
- ERROR = 0
- STATUS = 8181H
4.4 Timedriven PN_InOut / PN_InOut_Fast call - recommendation on application

Timedriven call - recommendation on application

If your application requires timedriven transfer of the CBA data instead of cyclic or event-driven processing, we recommend the following procedure to call the FB88 / FB90 blocks.

If you use a timedriven call, remember that the block must be called repeatedly after it has been started until the DONE flag is set. To allow the CBA data to be copied between the CPU and CP without any longer interruptions, the followon calls should be as fast as possible and independent of the timing.

Note the following recommendations for programming:

- The timing is provided by a timer OB; the timer OB should only cause the first call for the PROFINET CBA blocks FB88 or FB90 not by calling the block directly but, for example, by setting a start flag.
  
  The cycle for calling the time OB should not exceed 30 s.

- The call of the PROFINET CBA blocks FB88 and FB90 should then always be made in OB1; OB1 starts the call as soon as the start flag is set by the timer OB.

- After the first block call, this is repeated in OB1 until the DONE bit is set (or until an error occurs); the start flag must then be reset.

Result:
The CBA user data can be copied between the CPU and CP without any significant interruption thanks to separating the timer OB from the actual block calls in OB1. You can select the interval between the first calls depending on the requirements of your application.
5.1 Program blocks for open communications services (SEND/RECEIVE interface)

5.1.1 Overview of uses

Overview

The following program blocks are available for the SEND/RECEIVE interface for transferring data on configured FDL connections:

<table>
<thead>
<tr>
<th>Program block</th>
<th>Can be used with 1)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG_SEND (FC5)</td>
<td>x</td>
<td>for sending data</td>
</tr>
<tr>
<td>AG_RECV (FC6)</td>
<td>x</td>
<td>for receiving data</td>
</tr>
<tr>
<td>AG_LSEND (FC50)</td>
<td>x</td>
<td>for sending data</td>
</tr>
<tr>
<td>AG_LRECV (FC60)</td>
<td>x</td>
<td>for receiving data</td>
</tr>
</tbody>
</table>

1) Notes on the FCs for S7-300 and S7-400

To ensure the compatibility of PROFIBUS and Ind. Ethernet on the interface in the user program, you can use the FCs AG_LSEND and AG_LRECV on PROFIBUS as alternatives to AG_SEND and AG_RECV. There is no difference in the interface or the way they function. On PROFIBUS, however, you can only transfer data up to a maximum of 240 bytes even with these FCs although they are intended for longer data records on Industrial Ethernet. This is only possible if the block type and block version are permitted for the CP type you are using.

With the S7-CPs for S7-300 only the FCs AG_SEND and AG_RECV are used; on Industrial Ethernet even for transferring longer data records

The manuals contain information on the compatibility of the S7-CPs and the corresponding blocks (FCs / FBs). You will find an overview of the versions of the FCs/FBs in the documentation and block history.
Application

The following diagram illustrates the use of the FCs AG_SEND / AG_LSEND and AG_RECV / AG_LRECV for bi-directional data transfer on one configured FDL connection. With certain connection types, a job header should be included in the user data area.

![Diagram showing bi-directional data transfer with AG_SEND and AG_RECV on both communications partners](image5-1)

Image 5-1 Using AG_SEND and AG_RECV on both communications partners

Application without job header

With a specified FDL connection, the address and job parameters are specified by the configuration of the connection. The user program only provides the user data in the FDL data area when sending with AG_SEND / AG_LSEND or receives the data with AG_RECV / AG_LRECV.

Up to 240 bytes of user data can be transferred. This applies to PROFIBUS for both the AG_SEND or AG_LSEND functions.

Working with the job header

The following connection types require a job header in the FDL (user) data area:

- Unspecified FDL connection with free layer 2 access
- FDL connection with broadcast
- FDL connection with multicast

The following schematic illustrates the structure of the job buffer and the meaning and location of the parameters in the job header.
5.1 Program blocks for open communications services (SEND/RECEIVE interface)

### 5.1.2 AG_SEND / AG_LSEND

#### 5.1.2.1 Meaning and call - AG_SEND / AG_LSEND

**Meaning of the block**

The AG_SEND / AG_LSEND program block transfers data to the PROFIBUS CP for transmission on a configured FDL connection.

The selected data area can be a process image area, a memory bit area or a data block area.

Error free execution of the function is indicated when the entire FDL data area could be sent on PROFIBUS.

*Note:* Unless otherwise stated, all the following information applies equally to the FCs AG_SEND and AG_LSEND.
### Call

**Call interface in FBD representation**

![Call interface in FBD representation](image)

**Example in STL representation**

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call fc 5 (</td>
<td>//Function call</td>
</tr>
<tr>
<td>ACT := M 20.0,</td>
<td>//Job triggered by memory bit</td>
</tr>
<tr>
<td>ID := MW 22,</td>
<td>//Connection ID acc. to configuration</td>
</tr>
<tr>
<td>LADDR := W#16#01000,</td>
<td>//=LADDR 256 dec. in HW Config</td>
</tr>
<tr>
<td>SEND := P#db99.dbx10.0 byte 240,</td>
<td>//Buffer with send data</td>
</tr>
<tr>
<td>LEN := MW 24,</td>
<td>//Length for send data</td>
</tr>
<tr>
<td>DONE := M 20.1,</td>
<td>//Execution code</td>
</tr>
<tr>
<td>ERROR := M 20.2,</td>
<td>//Error code</td>
</tr>
<tr>
<td>STATUS := MW 26 );</td>
<td>//Status code</td>
</tr>
</tbody>
</table>

**Calls with job header**

The following table shows the connection types and job types for which parameters must be supplied in the job header.

The job header is located in the FDL (user) data area. It occupies the first 4 bytes and must be added to the length specified in the LEN parameter. The maximum user data length is therefore reduced for jobs with a job header to 236 bytes.

**Table 5-1 Supplying the job header in the user data area**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>FDL connection type</th>
<th>Broadcast</th>
<th>Multicast</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB address</td>
<td>Address of the destination station Range of values: 0..126 depending on node / 127 for broadcast/multicast</td>
<td>For AG_SEND no relevance; but area must be reserved.</td>
<td>For AG_SEND no relevance; but area must be reserved.</td>
</tr>
</tbody>
</table>
### Parameter | FDL connection type
---|---
**LSAP** | LSAP of the destination station<br.Range of values: 0..62 depending on node / 63 for broadcast<br.No significance but area must be reserved. / No significance but area must be reserved.
**Service 1)** | SDA (Send Data with Acknowledge):<br.Value: 00H<br>SDN (Send Data with No Acknowledge):<br.Value: 01H<br>No significance but area must be reserved. / No significance but area must be reserved.

1) for broadcast and multicast, only the SDN service is possible.
2) The information on broadcast and multicast in this column is relevant only when an unspecified FDL connection is used for broadcast or multicast. On a configured FDL connection (recommended application) with broadcast or multicast as the connection partner, the address parameters are assigned automatically according to the configuration.

#### 5.1.2.2 How AG_SEND / AG_LSEND work

**Operating principle**

The following diagram illustrates the normal sequence of data transmission triggered in the user program using AG_SEND.

The send job is executed as soon as the parameter ACT = 1 is passed.

Following this, the parameter ACT = 0 must be passed in at least one further call.

The status code in the output parameters DONE, ERROR and STATUS is updated in each block call and can be evaluated. To update the status code without starting a new send job, start a new block call with the parameter ACT = 0.

Refer to the sample program at the end of this section.
5.1.2.3 Explanation of the formal parameters - AG_SEND / AG_LSEND

**Explanation of the formal parameters**

The following table explains all the formal parameters for the AG_SEND / AG_LSEND functions:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>INPUT</td>
<td>BOOL</td>
<td>0, 1</td>
<td>If an FC is called with ACT=1, LEN bytes are sent from the ISO transport data area specified with the SEND parameter. If an FC is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.</td>
</tr>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>INT</td>
<td>1,2...64 (S7-400) 1,2...16 (S7-300)</td>
<td>The connection number of the FDL connection is specified in the parameter ID.</td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.</td>
</tr>
</tbody>
</table>
### 5.1 Program blocks for open communications services (SEND/RECEIVE interface)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
</table>
| SEND      | INPUT       | ANY       | (only the following are permitted as VARTYPE: WORD and DWORD are permitted) | Specifies the address and length  
The address of the data area points to one of the alternatives:  
• PI area  
• Memory bit area  
• Data block area  
With a call with job header, the FDL data area contains the job header and the user data. |
| LEN       | INPUT       | INT       | 1,2,...240 (or up to "length specified for SEND parameter") | Number of bytes to be sent from the FDL data area with this job. The possible values range from 1 to length specified for the SEND parameter.  
In a call, with job header, the length information is made up of the job header (4 bytes) + user data (1 to 236 bytes). Therefore LEN >= 4 ! |
| DONE      | OUTPUT      | BOOL      | 0: - 1: new data | The status parameter indicates whether or not the job was completed without errors.  
For the meaning in conjunction with the ERROR and STATUS parameters, refer to AG_SEND and AG_LSEND condition codes (Page 213). |
| ERROR     | OUTPUT      | BOOL      | 0: - 1: Error | Error code  
For the meaning in conjunction with the parameters DONE and STATUS, refer to AG_SEND and AG_LSEND condition codes (Page 213) |
| STATUS    | OUTPUT      | WORD      |                  | Status code  
For the meaning in conjunction with the parameters DONE and ERROR, refer to AG_SEND and AG_LSEND condition codes (Page 213) |

### 5.1.2.4 AG_SEND and AG_LSEND condition codes

#### Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

#### Note

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.
### Table 5-2  AG_SEND condition codes

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0000H</td>
<td>No job being executed.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Job active.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>7000H</td>
<td>The condition code is possible only with S7-400: The FC was called with ACT=0; the job has not yet been processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>No configuration or the FDL service has not yet started on the PROFIBUS CP.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8184H</td>
<td>Illegal data type specified for the SEND parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• FDL connection without job buffer: System error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• FDL connection with job buffer: Parameter LEN&lt;4 or illegal parameter in job header (with free layer 2 access).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8185H</td>
<td>LEN parameter longer than SEND source area.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8186H</td>
<td>ID parameter invalid. ID != 1, 2...16.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8301H</td>
<td>SAP not activated on destination station.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8302H</td>
<td>No receive resources on the destination station; the receiving station cannot process received data quickly enough or has not prepared any receive resources.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8303H</td>
<td>The PROFIBUS service (SDA Send Data with Acknowledge) is not supported on this SAP by the destination station.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This condition code can also occur temporarily when connections or gateways are downloaded &quot;in RUN&quot;.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8304H</td>
<td>The FDL connection is not established.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8311H</td>
<td>The destination station is not obtainable at the specified PROFIBUS address or the service is not possible for the specified PROFIBUS address.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8312H</td>
<td>PROFIBUS error on the CP: for example, bus short-circuit, own station not in ring.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8315H</td>
<td>Possible causes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Internal parameter error on an FDL connection with job header: Parameter LEN&lt;4 or illegal parameter in job header (with free layer 2 access).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Bus disruption</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Possible additional meaning:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• This error code can also occur with bus problems (for example physical disturbances due to bad cable connections or different settings for the transmission speed on the nodes).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F22H</td>
<td>Source area invalid, e.g.:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Area does not exist in the DB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LEN parameter &lt; 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F24H</td>
<td>Area error reading a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F28H</td>
<td>Alignment error reading a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F32H</td>
<td>Parameter contains a DB number that is too high.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F33H</td>
<td>DB number error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F3AH</td>
<td>Area not loaded (DB).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F42H</td>
<td>Timeout reading a parameter from the I/O area.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F44H</td>
<td>Address of the parameter to be read is disabled in the access track.</td>
</tr>
</tbody>
</table>
## Program blocks for PROFIBUS

### 5.1 Program blocks for open communications services (SEND/RECEIVE interface)

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8F7FH</td>
<td>Internal error, e.g. illegal ANY reference</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>e.g. parameter LEN=0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8090H</td>
<td>• No module with this module start address exists.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The FC being used does not match the system family being used (remember to use different FCs for S7300 and S7400).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8091H</td>
<td>Module start address not at a doubleword boundary.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8092H</td>
<td>In the ANY reference, a type other than BYTE is specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(S7-400 only)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80A4H</td>
<td>The communication bus connection between the CPU and CP is not established. (with newer CPU versions).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This can, for example, be caused by the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• No connection configuration;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The maximum number of CPs that can be operated at one time has been exceeded (for further information, refer to the CP manual).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B0H</td>
<td>The module does not recognize the data record.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B1H</td>
<td>The destination area is invalid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The amount of data to be sent exceeds the upper limit permitted for this service (e.g. destination area &gt; 240 bytes).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B2H</td>
<td>The communication bus connection between the CPU and CP is not established (with older CPU versions; otherwise 80A4H; for further information, refer to this code)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C0H</td>
<td>The data record cannot be read.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C1H</td>
<td>The specified data record is currently being processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C2H</td>
<td>There are too many jobs pending.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C3H</td>
<td>Resources occupied (memory).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C4H</td>
<td>Communication error (occurs temporarily, it is usually best to repeat the job in the user program).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80D2H</td>
<td>Module start address incorrect.</td>
</tr>
</tbody>
</table>

### 5.1.3 AG_RECV / AG_LRECV

#### 5.1.3.1 Meaning and call - AG_RECV / AG_LRECV

**Meaning of the block**

The AG_RECV program block receives the data transferred on a configured FDL connection from the PROFIBUS CP.

The data area specified for the receive data can be a process image area, a bit address area or a data block area.

Errorfree execution is indicated when the data could be received from the PROFIBUS CP.

**Note:**

All the following information applies equally to both FCs AG_RECV and AG_LRECV unless stated otherwise.
Call interface

Call interface in FBD representation

Example in STL representation

```
call fc 6 ( //Function call
ID := MW 30, //Connection ID acc. to configuration
LADDR := W#16#0100, //=LADDR 256 dec.in HW Config
RECV := F#M 10.0 BYTE 100, //Buffer for received data
NDR := DB 100.DBX 0.6, //Receive code
ERROR := DB 100.DBX 0.7, //Execution code
STATUS := DB 100.DBW 2, //Error code
LEN := DB 100.DBW 4 ); //Status code
```

Calls with job header

Table 5-3: Return parameters in the job header in the FDL (user) data area

<table>
<thead>
<tr>
<th>Parameter</th>
<th>FDL connection type</th>
<th>Broadcast</th>
<th>Multicast</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB address</td>
<td>Address of the sender</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Range of values: 0 to 126 depending on node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSAP</td>
<td>LSAP of the sender</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Range of values: 0 to 63 depending on node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td>SDN indication</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Send Data with No Acknowledge - Indication): Value: 01H</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>or</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDA indication</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Send Data with Acknowledge - Indication): Value: 00H</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDN indication</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Send Data with No Acknowledge - Indication): Value: 7FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDN indication</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Send Data with Acknowledge - Indication): Value: 7FH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.1.3.2 How AG_RECV / AG_LRECV work

Operating principle

The following diagram illustrates the normal sequence of data acceptance triggered by an AG_RECV in the user program.

Each AG_RECV job in the user program is acknowledged by the Ethernet CP with an entry in the output parameters NDR, ERROR and STATUS.

Key:
1) Parameter transfer DONE, ERROR, STATUS
### 5.1.3.3 Explanation of the formal parameters - AG_RECV / AG_LRECV

#### Explanation of the formal parameters

The following table explains all the formal parameters for the AG_RECV / AG_LRECV function:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INPUT</td>
<td>INT</td>
<td>1,2...16 (S7-300) 1,2...32 (S7-400)</td>
<td>The connection number of the FDL connection is specified in the parameter ID.</td>
</tr>
<tr>
<td>LADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.</td>
</tr>
<tr>
<td>RECV</td>
<td>INPUT</td>
<td>ANY (only the following are permitted as VARTYPE: WORD and DWord are permitted)</td>
<td>Specifies the address and length The address of the FDL data area points to one of the alternatives: • PI area • Memory bit area • Data block area With a call with job header, the FDL data area contains the job header and the user data.</td>
<td></td>
</tr>
<tr>
<td>LEN</td>
<td>OUTPUT</td>
<td>INT</td>
<td>1,2,...240</td>
<td>Specifies the number of bytes to be received in the FDL data area from the PROFIBUS CP. In a call, with job header, the length information is made up of the job header (4 bytes) + user data (1 to 236 bytes). Therefore LEN &gt;= 4 !</td>
</tr>
<tr>
<td>NDR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: new data</td>
<td>This parameter indicates whether new data were received. For the meaning in conjunction with the parameters ERROR and STATUS, refer to AG_RECV and AG_LRECV condition codes (Page 219).</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Error</td>
<td>Error code For the meaning in conjunction with the parameters NDR and STATUS, refer to AG_RECV and AG_LRECV condition codes (Page 219).</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td></td>
<td>Status code For the meaning in conjunction with the parameters NDR and ERROR, refer to AG_RECV and AG_LRECV condition codes (Page 219).</td>
</tr>
</tbody>
</table>
5.1.3.4 AG_RECV and AG_LRECV condition codes

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>New data accepted.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8180H</td>
<td>• There is no data available yet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The configuration is missing or the FDL service has not started on the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PROFIBUS CP (occurs here instead of the code 0,1,8183H).</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>Job active.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>No configuration or the FDL service has not yet started on the PROFIBUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CP.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8184H</td>
<td>• Illegal data type specified for the RECV parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• System error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8185H</td>
<td>Destination buffer (RECV) is too short.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8186H</td>
<td>ID parameter invalid. ID != 1, 2...16.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8303H</td>
<td>The PROFIBUS service (SDA - Send Data with Acknowledge) is not supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>on this SAP.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This condition code can also occur temporarily when connections or gate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ways are downloaded &quot;in RUN&quot;</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8304H</td>
<td>The FDL connection is not established.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F23H</td>
<td>Source area invalid, e.g. Area does not exist in the DB.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F25H</td>
<td>Area error writing a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F29H</td>
<td>Alignment error writing a parameter</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F30H</td>
<td>Parameter is in the writeprotected 1st current data block.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F31H</td>
<td>Parameter is in the writeprotected 2nd current data block.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F32H</td>
<td>Parameter contains a DB number that is too high.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F33H</td>
<td>DB number error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F3Ah</td>
<td>Destination area not loaded (DB).</td>
</tr>
</tbody>
</table>
| 0   | 1     | 8F43H    | Timeout writing a parameter to the I/O area.}

Program blocks for SIMATIC NET S7 CPs
Programming Manual, 11/2015, C79000-G8976-C229-08

219
### Program blocks for PROFIBUS

#### 5.1 Program blocks for open communications services (SEND/RECEIVE interface)

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8F45H</td>
<td>Address of the parameter to be written is disabled in the access track.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F7FH</td>
<td>Internal error, e.g. illegal ANY reference.</td>
</tr>
</tbody>
</table>
| 0   | 1     | 8090H  | • No module with this module start address exists.  
• The FC being used does not match the system family being used (remember to use different FCs for S7300 and S7400). |
| 0   | 1     | 8091H  | Module start address not at a doubleword boundary. |
| 0   | 1     | 8092H  | In the ANY reference, a type other than BYTE is specified.  
(S7-400 only) |
| 0   | 1     | 80A0H  | Negative acknowledgment reading from the module. |
| 0   | 1     | 80A4H  | The communication bus connection between the CPU and CP is not established. (with newer CPU versions).  
This can, for example, be caused by the following:  
• No connection configuration;  
• The maximum number of CPs that can be operated at one time has been exceeded  
(for further information, refer to the CP manual). |
| 0   | 1     | 80B0H  | The module does not recognize the data record. |
| 0   | 1     | 80B1H  | Possible causes:  
• The destination area is invalid.  
• The destination area is too short.  
The destination area for the received data was adequately dimensioned.  
Remedy: Run another receive call with maximum receive buffer size. This applies regardless of the connection type (unicast / multicast / broadcast) and the device family (S7-300 / S7-400). |
| 0   | 1     | 80B2H  | The communication bus connection between the CPU and CP is not established. |
| 0   | 1     | 80C0H  | The data record cannot be read. |
| 0   | 1     | 80C1H  | The specified data record is currently being processed. |
| 0   | 1     | 80C2H  | There are too many jobs pending. |
| 0   | 1     | 80C3H  | Resources occupied (memory). |
| 0   | 1     | 80C4H  | Communication error (occurs temporarily, it is usually best to repeat the job in the user program). |
| 0   | 1     | 80D2H  | Module start address incorrect. |
5.2 Program blocks for DP (distributed I/O) with S7-300

5.2.1 Overview of uses

Overview

The following program blocks are available for the DP master and DP slave modes with an S7-300:

<table>
<thead>
<tr>
<th>Program block</th>
<th>can be used with:</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DP master</td>
<td>DP slave</td>
</tr>
<tr>
<td>DP_SEND (FC1)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DP_RECV (FC2)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DP_DIAG (FC3)</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>DP_CTRL (FC4)</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

Application

The following diagram illustrates the use of the DP_SEND and DP_RECV program blocks on the DP master and DP slave.
5.2 Program blocks for DP (distributed I/O) with S7-300

5.2.2 DP_SEND

5.2.2.1 Meaning and call - DP_SEND

Meaning

The DP_SEND program block transfers data to the PROFIBUS CP. Depending on the mode of the PROFIBUS CP, DP_SEND has the following significance:

- When used in the DP master
  The block transfers the data of a specified DP output area to the PROFIBUS CP for output to the distributed I/O system.

- When used in the DP slave
  The block transfers the input data of the DP slave to the PROFIBUS CP for transfer to the DP master

The selected data area can be a process image area, a memory bit area or a data block area.

Correct execution is signaled when the entire DP data area could be accepted by the PROFIBUS CP.

To start the DP master exactly one DP-SEND or DP-RECV call must precede the call sequence. The following applies to this first call:

If DP-SEND is used for initialization, the transferred data area is not accepted and "0" is sent to the slaves. The user data to be transferred is accepted only with the second block call.

Call interface

Example of a call in STL representation

```
STL Explanation

call fc 1 ( //DP_SEND function call
CPLADDR:= W#16#0120,
SEND := P#db17.dbx0.0 byte 103,
DONE := M 99.1,
ERROR := M 99.0,
STATUS := MW 104 );
```
5.2.2.2 How DP_SEND works

Operating principle

The following flow diagram illustrates the normal sequence of data transfer triggered with DP_SEND in the user program.

In the diagram, it is assumed that the DP master has already been initialized by a previous DP_SEND or DP-RECV call.

Each DP_SEND job in the user program is acknowledged by the PROFIBUS CP setting values in the DONE, ERROR and STATUS output parameters.
Guarantee of data transfer

The diagram also shows that with the confirmation DONE=1, ERROR=0 and STATUS=0000, data transfer to the communications partner is functioning correctly.

The latest send data transferred to the PROFIBUS CP is always passed on to the communications partner. For this reason, new user data must only be entered in the send buffer following a positive acknowledgment (DONE=1, ERROR=0, STATUS=0000).

5.2.2.3 Explanation of the formal parameters - DP_SEND

Explanation of the formal parameters

The following table explains all the formal parameters for the DP_SEND function:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
</table>
| CPLADDR   | INPUT       | WORD      |                 | Module start address  
|= When you configure the CP, the module start address is displayed in the configuration table. Specify this address here. |
| SEND      | INPUT       | ANY       |                 | Specifies the address and length  
|= The address of the DP data area points to one of the alternatives:  
|= • PI area  
|= • Memory bit area  
|= • Data block area  
|= The length must be set for  
|= • DP master: 1...2160  
|= • DP slave: 1...240 |
| DONE      | OUTPUT      | BOOL      | 0: - 1: new data | The status parameter indicates whether or not the job was completed without errors.  
|= For the meaning in conjunction with the ERROR and STATUS parameters, refer to DP_SEND condition codes (Page 225). |
| ERROR     | OUTPUT      | BOOL      | 0: - 1: Error   | Error code  
|= For the meaning in conjunction with the DONE and STATUS parameters, refer to DP_SEND condition codes (Page 225) |
| STATUS    | OUTPUT      | WORD      |                 | Status code  
|= For the meaning in conjunction with the DONE and ERROR parameters, refer to DP_SEND condition codes (Page 225) |
5.2.2.4 DP_SEND condition codes

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 5-5 DP_SEND condition codes

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8180H</td>
<td>• Startup:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The DP service was started but data acceptance is not yet possible.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Normal operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Data transfer active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• DP has not started due to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– CP STOP or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– &quot;No parameter assignment&quot; (occurs here instead of the code 0,1,8183H)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>New data transferred without error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>No configuration or the DP service has not yet started on the PROFIBUS CP.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8184H</td>
<td>System error or bad parameter type.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F22H</td>
<td>Area length error reading a parameter (e.g. DB too short).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F23H</td>
<td>Area length error writing a parameter (e.g. DB too short).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F24H</td>
<td>Area error reading a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F25H</td>
<td>Area error writing a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F28H</td>
<td>Alignment error reading a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F29H</td>
<td>Alignment error writing a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F30H</td>
<td>Parameter is in the writeprotected 1st current data block.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F31H</td>
<td>Parameter is in the writeprotected 2nd current data block.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F32H</td>
<td>Parameter contains a DB number that is too high.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F33H</td>
<td>DB number error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F3AH</td>
<td>Destination area not loaded (DB).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F42H</td>
<td>Timeout reading a parameter from the I/O area.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F43H</td>
<td>Timeout writing a parameter to the I/O area.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F44H</td>
<td>Address of the parameter to be read is disabled in the access track.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F45H</td>
<td>Address of the parameter to be written is disabled in the access track.</td>
</tr>
</tbody>
</table>
### 5.2.3 DP_RECV

#### 5.2.3.1 Meaning and call - DP_RECV

**Meaning**

The DP_RECV program block receives data over PROFIBUS. DP_RECV has the following significance depending on the mode of the PROFIBUS CP:

- **When used in the DP master**
  
  DP_RECV receives the process data from the distributed I/O along with status information and enters this in a specified DP input area.

- **When used on the DP slave**
  
  DP_RECV accepts the output data transferred by the DP master in the DP data area specified in the block.

The data area specified for the receive data can be a process image area, a bit address area or a data block area.

Errorfree execution of the function is signaled when the entire DP data input area could be transferred by the PROFIBUS CP.

Note that FC DP_RECV must be called successfully at least once on the DP slave in the user program if output data was configured for this DP slave. Please read the information in the manual.

To start the DP master exactly one DP-SEND or DP-RECV call must precede the call sequence. The following applies to this first call:

- If DP-RECV is used for initialization, the received data is not adopted. The user data to be received is accepted only with the second block call.

#### Error Codes

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8F7FH</td>
<td>Internal error, e.g. illegal ANY reference.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8090H</td>
<td>No module with this address exists.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8091H</td>
<td>Logical base address not at a double word boundary.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80A1H</td>
<td>Negative acknowledgment writing to the module.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B0H</td>
<td>The module does not recognize the data record.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B1H</td>
<td>The number of data bytes to be sent exceeds the upper limit for this service (applies to DP master and DP slave mode).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C0H</td>
<td>The data record cannot be read.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C1H</td>
<td>The specified data record is currently being processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C2H</td>
<td>There are too many jobs pending.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C3H</td>
<td>Resources occupied (memory).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C4H</td>
<td>Communication error (occurs temporarily, it is usually best to repeat the job in the user program).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80D2H</td>
<td>Logical base address incorrect.</td>
</tr>
</tbody>
</table>
Additional task: Entering the status byte

The DP_RECV function has the following additional task:

- Updating the DP status byte DPSTATUS. This means that DP_RECV handles tasks for DP diagnostics. If no receive data is configured, DP_RECV must be called with a length of 1 to update the DPSTATUS status byte (applies only to DP masters). Please read the information in the manual as well.
- Enabling the station list (see DP_DIAG (Page 233)).

Call interface

Example in STL representation

| STL |
|-----|-----|
| STL | Explanation |
| call fc 2 { |
| CPLADDR:= W#16#0120, |
| RECV := P#db17.dbx240.0 byte 103, |
| NDR := M 99.1, |
| ERROR := M 99.0, |
| STATUS := MW 104, |
| DPSTATUS:= MB 0 ); |

5.2.3.2 How DP_RECV works

Operating principle

The following flow diagram illustrates the normal sequence of data transfer triggered with DP_RECV in the user program.

Each DP_RECV job in the user program is acknowledged by the PROFIBUS CP setting values in the NDR, ERROR and STATUS output parameters.
Guarantee of data acceptance

The diagram also shows that the confirmation NDR=1, ERROR=0 and STATUS=0000 indicates reliable data reception. Requirement: The DP master and the DP slaves are in the data transfer phase.

Note the following:

- In DP master mode:
  If a DB slave is not in the data transfer phase, the corresponding received data is set to 0.
  If the DP master is neither in the RUN nor CLEAR state (bits 4 and 5 in DPSTATUS), all...
the received data is set to 0.
If data has been received from the DP slave several times since the last DP_RECV function call, only the last received data is fetched with the next DP_RECV.

- In DP slave mode:
  If the DP slave is not in the data transfer phase (bit 1 in DPSTATUS) or the DP master is in the CLEAR state (bit 2 in DPSTATUS), the received data is set to 0.
  If data has been received from the DP master several times since the last DP_RECV function call, only the last received data is fetched with the next DP_RECV.

5.2.3.3 Explanation of the formal parameters - DP_RECV

Explanation of the formal parameters
The following table explains all the formal parameters for the function DP_RECV:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.</td>
</tr>
</tbody>
</table>
| RECV      | INPUT       | ANY       |                 | Specifies the address and length The address of the DP data area points to one of the alternatives:  
- PI area  
- Memory bit area  
- Data block area  
The length must be set for:  
- DP master: 1...2160  
- DP slave: 1...240  
- DP master; only read status byte: 1 (see also CP manual) |
| NDR       | OUTPUT      | BOOL      | 0: - 1: New data accepted | The status parameter indicates whether or not new data was accepted. For the meaning in conjunction with the ERROR and STATUS parameters, refer to DP_RECV condition codes (Page 230). |
| ERROR     | OUTPUT      | BOOL      | 0: - 1: Error | Error code For the meaning in conjunction with the NDR and STATUS parameters, refer to DP_RECV condition codes (Page 230). |
5.2 Program blocks for DP (distributed I/O) with S7-300

### Parameter Declaration Data type Possible values Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Status code</td>
<td>Status code For the meaning in conjunction with the parameters NDR and ERROR, refer to DP_RECV condition codes (Page 230).</td>
</tr>
<tr>
<td>DPSTATUS</td>
<td>OUTPUT</td>
<td>Byte</td>
<td>For coding, see below under DPSTATUS</td>
<td>DP status code</td>
</tr>
</tbody>
</table>

#### 5.2.3.4 DP_RECV condition codes

**Condition codes**

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

**Note**

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0   | 0     | 8180H   | • Startup:  
  The DP service was started but data acceptance is not yet possible.  
• Normal operation  
  Data transfer active.  
• DP has not started due to:  
  − CP STOP or  
  − "No parameter assignment" (occurs here instead of the code 0,1,8183H). |
| 1   | 0     | 0000H   | New data accepted without error. |
| 0   | 1     | 8183H   | No configuration or the DP service has not yet started on the PROFIBUS CP. |
| 0   | 1     | 8184H   | System error or bad parameter type. |
| 0   | 1     | 8F22H   | Area length error reading a parameter (e.g. DB too short). |
| 0   | 1     | 8F23H   | Area length error writing a parameter (e.g. DB too short). |
| 0   | 1     | 8F24H   | Area error reading a parameter. |
| 0   | 1     | 8F25H   | Area error writing a parameter. |
| 0   | 1     | 8F28H   | Alignment error reading a parameter. |
| 0   | 1     | 8F29H   | Alignment error writing a parameter. |
### 5.2.3.5 DPSTATUS - DP_RECV

**DPSTATUS**

The coding of the DPSTATUS output parameter is different for the DP master mode and DP slave mode.

**DP master mode**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>not used</td>
</tr>
<tr>
<td>6</td>
<td>This bit is not set. Please read the information in the manual as well.</td>
</tr>
</tbody>
</table>
### Program blocks for PROFIBUS

#### 5.2 Program blocks for DP (distributed I/O) with S7-300

#### Bit Meaning

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,4</td>
<td>Values for DPSTATUS of the DP master:</td>
</tr>
<tr>
<td></td>
<td>00 RUN</td>
</tr>
<tr>
<td></td>
<td>01 CLEAR</td>
</tr>
<tr>
<td></td>
<td>10 STOP (this is now the OFFLINE mode)</td>
</tr>
<tr>
<td></td>
<td>11 OFFLINE</td>
</tr>
<tr>
<td></td>
<td>Please read the information in the manual as well.</td>
</tr>
<tr>
<td>3</td>
<td>Value 1: Cyclic synchronization is active</td>
</tr>
<tr>
<td>2</td>
<td>Value 0: No new diagnostic data exists</td>
</tr>
<tr>
<td></td>
<td>Value 1: Evaluation of diagnostic list useful; at least one station has new diagnostic data</td>
</tr>
<tr>
<td>1</td>
<td>Value 0: All DP slaves are in the data transfer phase</td>
</tr>
<tr>
<td></td>
<td>Value 1: Evaluating the station list is useful</td>
</tr>
<tr>
<td>0</td>
<td>DP mode</td>
</tr>
<tr>
<td></td>
<td>Value 0: DP master mode</td>
</tr>
<tr>
<td></td>
<td>The other bits only have the specified meaning when this bit is not set.</td>
</tr>
</tbody>
</table>

#### DP slave mode

![Bit diagram]

Table 5- 7   Meaning of the bits in DPSTATUS in DP slave mode

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-5</td>
<td>not used</td>
</tr>
<tr>
<td>4</td>
<td>This bit is not set.</td>
</tr>
<tr>
<td></td>
<td>Please read the information in the manual as well.</td>
</tr>
<tr>
<td>3</td>
<td>This bit is not set.</td>
</tr>
<tr>
<td></td>
<td>Please read the information in the manual as well.</td>
</tr>
<tr>
<td>2</td>
<td>Value 1: DP master 1 is in the CLEAR mode. The DP slave receives the value 0 in the DP data intended for the outputs. This has no effect on the send data.</td>
</tr>
<tr>
<td>1</td>
<td>Value 1: The configuration/parameter assignment is not yet completed.</td>
</tr>
<tr>
<td>0</td>
<td>Value 1: DP slave mode.</td>
</tr>
<tr>
<td></td>
<td>The other bits only have the specified meaning when this bit is set.</td>
</tr>
</tbody>
</table>

#### Note

Please note, that DPSTATUS must not be evaluated until the return parameter NDR=1 is set.
5.2.4 DP_DIAG

5.2.4.1 Meaning and call - DP_DIAG

Meaning of the block

The DP_DIAG program block is used to request diagnostic information. The following types of job are possible:

- Request DP station list
- Request DP diagnostics list;
- Request DP single status;
- Read input/output data of a DP slave acyclically
- Read older DP single diagnostic information
- Read DP status.
- Read DP mode for PLC/CP stop
- Read current status of the DP slave.

Diagnostics data can also be requested for a specific slave by specifying a station address. To transfer the diagnostic data to the CPU, you should reserve a memory area in the CPU and specify this area in the call. This memory area can be a data block area or a bit memory area. The maximum length of the available memory area must also be specified in the job.

Note

FC DP_DIAG is only of practical use in the DP master mode.

Exclusion

As long as this block is running, it must not be supplied with new job data.

Exception: Requesting the DP station list or DP diagnostics list.

Call interface

![Call interface diagram]
Example in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| call fc 3 ( //DP_DIAG function call  
CPLADDR:= W#16#0120,  
DTYPE := B#16#00,  
STATION:= B#16#03,  
DIAG := P#db18.dbx0.0 byte 16,  
NDR := M 70.0,  
ERROR := M 70.1,  
STATUS := MW 72,  
DIAGLNG:= MB 20 ); |

5.2.4.2 How DP_DIAG works

Sequence / handling on the call interface

The DP_DIAG function call is processed during cyclic execution of the user program as follows:

The job is triggered with the first call. Diagnostic data is only returned in the acknowledgment of one of the subsequent calls.
**Note**

Please note the following special feature of the job types read_DP_station_list and read_DP_diagnostic_list:

- The diagnostic job supplies the diagnostic data available at the time of the last DP_RECV call. Reading a list prevents the data from being read out again (return value 0x8182).
- The lists are released again after a new diagnostic event followed by a DP_RECV call.

After calling DP_DIAG, you obtain information indicating one of the situations below:

- **NDR=0, ERROR=0, STATUS=8181**  
  As long as the code combination NDR=0, ERROR=0 and STATUS=8181 is set, the job parameters must not be modified.

- **NDR=1**  
  The parameter value NDR=1 indicates that valid diagnostic data is available. Additional information is possible in the STATUS parameter.

- **NDR=0, ERROR=1**  
  An error has occurred. The diagnostic data is invalid. The error message is located in STATUS.
### 5.2.4.3 Explanation of the formal parameters - DP_DIAG

The following table explains all the formal parameters for the function DP_DIAG:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.</td>
</tr>
<tr>
<td>DTYPE</td>
<td>INPUT</td>
<td>BYTE</td>
<td>0: Station list 1: Diagnostic list 2: Current diagnostic info 3: Older diagnostics info 4: Read status 5: Read status for CPU STOP 6: Read status for CP STOP 7: Read input data (acyclically) 8: Read output data (acyclically) 10: Read current status of the DP slave</td>
<td>Diagnostics type</td>
</tr>
<tr>
<td>STATION</td>
<td>INPUT</td>
<td>BYTE</td>
<td></td>
<td>Station address of the DP slave</td>
</tr>
</tbody>
</table>
| DIAG      | INPUT       | ANY       | The length must be set from 1 to 240 | Specifies the address and length Address of the data area. References the following alternatives:  
  - PI area  
  - Memory bit area  
  - Data block area |
| NDR       | OUTPUT      | BOOL      | 0: - 1: new data | This parameter indicates whether or not new data were accepted. For the meaning in conjunction with the parameters ERROR and STATUS, refer to DP_DIAG codes (Page 239) |
| ERROR     | OUTPUT      | BOOL      | 0: - 1: Error | Error code For the meaning in conjunction with the parameters NDR and STATUS, refer to DP_DIAG codes (Page 239). |
### 5.2 Program blocks for DP (distributed I/O) with S7-300

#### Parameter Declaration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
</table>
| STATUS    | OUTPUT      | WORD      |                 | Status code  
For the meaning in conjunction with the parameters NDR and ERROR, refer to DP_DIAG codes (Page 239). |
| DIAGLNG   | OUTPUT      | BYTE      |                 | This contains the actual length (in bytes) of the data made available by the PROFIBUS CP, regardless of the buffer size specified in the DIAG parameter. The following applies to job types with DTYPE 4, 5 and 6. Here, DIAGLNG always has the value "1". The value returned in the DIAG parameter is not relevant for the evaluation in these cases. In these cases, the relevant value is contained in the STATUS parameter. |

#### 5.2.4.4 Job types - DP_DIAG

#### Job types

The following overview of the specifications for DTYPE, STATION and DIAGLNG shows the permitted or useful entries.

Table 5-8 Job types for DP_DIAG

<table>
<thead>
<tr>
<th>DTYPE</th>
<th>Corresponds to job</th>
<th>Parameter STATION</th>
<th>DIAGLNG</th>
<th>Acknowledgement code (contained in the STATUS parameter; shown in Table &quot;DP_DIAG codes&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Read DP station list</td>
<td>---</td>
<td>- ignored -</td>
<td>With the DP station list, you obtain information in the user program on the status and availability of DP slaves. The information in the DP station list relates to all DP slaves assigned to the DP master by the configuration.</td>
</tr>
<tr>
<td>1</td>
<td>Read DP diagnostics list</td>
<td>---</td>
<td>- ignored -</td>
<td>The DP diagnostics list informs the user program about the DP slaves with new diagnostics data.</td>
</tr>
<tr>
<td>2</td>
<td>Read current DP single diagnostic data</td>
<td>1...126</td>
<td>&gt;=6</td>
<td>The current DP single diagnostics informs the user program of the current diagnostics data of a DP slave.</td>
</tr>
<tr>
<td>3</td>
<td>Read older DP single diagnostic information</td>
<td>1...126</td>
<td>&gt;=6</td>
<td>The older DP single diagnostics informs the CPU program of the older diagnostics data of a DP slave. This data is stored on the PROFIBUS CP and read according to the &quot;last in - first out&quot; principle in the ring buffer. The structure of the ring buffer is explained below. If changes occur quickly in the DP slave diagnostic data, this function allows the diagnostic data of a DP slave to be acquired and evaluated in the CPU program of the DP master.</td>
</tr>
</tbody>
</table>
5.2 Program blocks for DP (distributed I/O) with S7-300

<table>
<thead>
<tr>
<th>DTYPE</th>
<th>Corresponds to job</th>
<th>Parameter</th>
<th>DIAGLNG</th>
<th>Acknowledgement code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read the operating status requested with DP-CTRL job (CTYPE=4)</td>
<td>STATION</td>
<td>=1</td>
<td>With this job, the DP operating status can be read that was set previously with the DP-CTRL job (CTYPE=4). Note: The operating status that is read out does not necessarily match the current operating status. The following statuses are possible:</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>• RUN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• CLEAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• STOP (is mapped to the OFFLINE status) ^)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• OFFLINE</td>
</tr>
</tbody>
</table>

| 5     | Read DP status for CPU STOP | STATION | =1       | With this job you can find out the DP status to which the PROFIBUS CP changes if the CPU changes to STOP: |
|       |                     |           |         | • RUN                  |
|       |                     |           |         | • CLEAR                |
|       |                     |           |         | • STOP (is mapped to the OFFLINE status) ^) |
|       |                     |           |         | • OFFLINE              |

As default, the PROFIBUS CP changes to the DP status CLEAR if the CPU changes to STOP.

| 6     | Read DP status for CP STOP | STATION | =1       | With this job you can find out the DP status to which the PROFIBUS CP changes if the CP changes to STOP: |
|       |                     |           |         | • STOP (is mapped to the OFFLINE status) ^) |
|       |                     |           |         | • OFFLINE              |

As default, the PROFIBUS CP changes to the DP status OFFLINE if the CP changes to STOP.

| 7     | Read input data         | 1...126   | >=1      | With this job, the DP master (class 2) reads the input data of the DP slave. This function is also known as shared input. |
| 8     | Read output data        | 1...126   | >=1      | With this job, the DP master (class 2) reads the output data of a DP slave. This function is also known as shared output. |
| 10    | Read current status of the DP slave | 1...126   | >=0      | With this job, you can read out the current status of the DP slave. The following statuses are possible: |
|       |                     |           |         | • The DP master exchanges data with the DP slave cyclically. |
|       |                     |           |         | • The DP master reads the input data of the DP slave cyclically. |
|       |                     |           |         | • The DP master reads the output data of the DP slaves cyclically. |
|       |                     |           |         | • The DP master is not currently processing this DP slave cyclically. |

^) The STOP status is no longer supported on the latest modules (as of module type DA02).
5.2.4.5  Ring buffer for diagnostics data - DP_DIAG

Ring Buffer for Diagnostic Data

The following diagram illustrates how diagnostic data is read using the “read older DP single diagnostic data” function. The first access reads the most recent of the older diagnostic data.

![Image 5-3 Ring Buffer for Diagnostic Data](image.png)

When the current diagnostic data is read out, the read pointer is reset to the first older diagnostic data.

5.2.4.6  DP_DIAG codes

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00xxH</td>
<td>No diagnosis data available</td>
</tr>
<tr>
<td>8FxxH</td>
<td>Error (User program must query the output parameter RET_VAL)</td>
</tr>
</tbody>
</table>

Note

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.
### Table 5-9  DP_DIAG codes

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Possible with DTYPE</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>2-10</td>
<td>Job active. DP master not started due to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• CP STOP or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• &quot;no parameter assignment&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(occurs here instead of the code 0,1,8183H).</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8182H</td>
<td>0</td>
<td>Triggering job pointless. DP master not started due to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• CP STOP or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• &quot;no parameter assignment&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(occurs here instead of the code 0,1,8183H).</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8182H</td>
<td>1</td>
<td>No new diagnostic data exist. DP master not started due to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• CP STOP or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• &quot;no parameter assignment&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(occurs here instead of the code 0,1,8183H).</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>0, 1 and 4-9</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>With DTYPE 2, 3 and 10, error-free execution is indicated by a status code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>other than &quot;0&quot;. Below you will see the detailed status codes for error-free execution for the range: 82XXH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If an error occurs in execution, you receive status codes in the following ranges: 80XXH, 83XXH, 8FXXH</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8222H</td>
<td>7, 8</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The length of the DP slave data that was read is not the same as the data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>length expected by the DP master based on the module list of the DP slave in the CP database.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8227H</td>
<td>7, 8</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Message: No data exists.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8231H</td>
<td>4, 5, 6</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Message: The DP status is already &quot;RUN&quot;</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8232H</td>
<td>4, 5, 6</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Message: The DP status is already &quot;CLEAR&quot;</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8233H</td>
<td>4, 5, 6</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Message: The DP status is already STOP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The STOP status is now the OFFLINE status (here code 8234H).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Please read the information in the manual as well.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8234H</td>
<td>4, 5, 6</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Message: The DP status is already &quot;OFFLINE&quot;</td>
</tr>
</tbody>
</table>
### Program blocks for PROFIBUS

#### 5.2 Program blocks for DP (distributed I/O) with S7-300

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Possible with DTYPE</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>823AH</td>
<td>2, 3, 7, 8</td>
<td>Job completed without error. Message: 241 or 242 bytes of data were read. 240 bytes of data are available.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8241H</td>
<td>2, 3, 10</td>
<td>Job completed without error. Message: The specified DP slave was not configured.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8243H</td>
<td>2, 3, 10</td>
<td>Job completed without error. Message: The module list of the DP slave in the CP database only contains empty modules.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8245H</td>
<td>2, 3, 10</td>
<td>Job completed without error. Message: The DP slave is in the &quot;read input data cyclically&quot; mode.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8246H</td>
<td>2, 3, 10</td>
<td>Job completed without error. Message: The DP slave is in the &quot;read output data cyclically&quot; mode. Note: This is the default code for the named diagnostics types if there is no special situation to signal.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8248H</td>
<td>2, 3, 10</td>
<td>Job completed without error. Message: The DP slave is deactivated due to a DP mode change (e.g. CP mode selector set to STOP).</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>824AH</td>
<td>2, 3, 10</td>
<td>Job completed without error. Message: The DP slave is deactivated due to a DP_CTRL job in the user program.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8090H</td>
<td>0-10</td>
<td>Logical base address of the module is invalid</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B0H</td>
<td>0-10</td>
<td>The module does not recognize the data record or is changing from RUN --» STOP.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B1H</td>
<td>0-10</td>
<td>Specified data record length incorrect</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C0H</td>
<td>0-10</td>
<td>Data record cannot be read</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C1H</td>
<td>0-10</td>
<td>The specified data record is being processed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C2H</td>
<td>0-10</td>
<td>Too many jobs pending</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C3H</td>
<td>0-10</td>
<td>Resources (memory) occupied</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C4H</td>
<td>0-10</td>
<td>Communication error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80D2H</td>
<td>0-10</td>
<td>Logical base address wrong</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>0-10</td>
<td>DP master not configured.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8184H</td>
<td>0-10</td>
<td>System error or bad parameter type.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8311H</td>
<td>&gt;=2</td>
<td>DTYPE parameter outside range of values.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8313H</td>
<td>2, 3, 7, 8, 10</td>
<td>STATION parameter outside range of values.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8321H</td>
<td>&gt;=2</td>
<td>The DP slave is not providing any valid data.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8326H</td>
<td>7, 8</td>
<td>The DP slave has more than 242 bytes of data available. The PROFIBUS CP supports a maximum of 242 bytes.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8335H</td>
<td>7, 8</td>
<td>The PROFIBUS CP is in PROFIBUS status: &quot;Station not in ring&quot;.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8341H</td>
<td>2, 3, 7, 8, 10</td>
<td>The specified slave was not configured</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8342H</td>
<td>7, 8</td>
<td>The DP slave with the PROFIBUS address specified in the STATION parameter is not obtainable.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8349H</td>
<td>7, 8</td>
<td>The DP master is in the OFFLINE mode.</td>
</tr>
</tbody>
</table>
### NDR, ERROR, STATUS Possible with DTYPE Meanings

<table>
<thead>
<tr>
<th>NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Possible with DTYPE</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8F22H</td>
<td>0-10</td>
<td>Area length error reading a parameter (e.g. DB too short)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F23H</td>
<td>0-10</td>
<td>Area length error writing a parameter (e.g. DB too short)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F24H</td>
<td>0-10</td>
<td>Range error when reading a parameter</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F25H</td>
<td>0-10</td>
<td>Area error writing a parameter</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F28H</td>
<td>0-10</td>
<td>Orientation error when reading a parameter</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F29H</td>
<td>0-10</td>
<td>Alignment error writing a parameter</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F30H</td>
<td>0-10</td>
<td>Parameter is in the writeprotected 1st current data block</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F31H</td>
<td>0-10</td>
<td>Parameter is in the writeprotected 2nd current data block</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F32H</td>
<td>0-10</td>
<td>The DB number in the parameter is too high</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F33H</td>
<td>0-10</td>
<td>DB number error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F3AH</td>
<td>0-10</td>
<td>Area not loaded (DB)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F42H</td>
<td>0-10</td>
<td>Timeout reading a parameter from the I/O area</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F43H</td>
<td>0-10</td>
<td>Timeout writing a parameter to the I/O area</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F44H</td>
<td>0-10</td>
<td>Address of the parameter to be read locked in the access track</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F45H</td>
<td>0-10</td>
<td>Address of the parameter to be written is disabled in the access track</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F7FH</td>
<td>0-10</td>
<td>Internal error, e.g. illegal ANY reference</td>
</tr>
</tbody>
</table>

### 5.2.5  DP_CTRL

#### 5.2.5.1  Meaning and call - DP_CTRL

**Meaning of the block**

The DP_CTRL program block transfers control jobs to the PROFIBUS CP. You specify a job field (CONTROL parameter) to specify the control job in greater detail.

The following types of job are possible:

- Global control acyclic/cyclic;
- Delete older diagnostic data;
- Set current DP mode;
- Set DP mode for PLC/CP STOP;
- Read input/output data cyclically;
- Set the operating mode of the DP slave.
There are restrictions relating to the job types listed here (please refer to the information in the manual for the module).

**Note**

FC DP_CTRL is only of practical use in the DP master mode.

**Connector**

As long as this block is running, it must not be supplied with new job data.

**Call interface**

![Diagram showing DP_CTRL block with ports: CPLADDR, CONTROL, DONE, ERROR, STATUS, BOOL, WORD]

**Example in STL representation**

```stl
CALL fc 4 (
  CPLADDR:= W#16#0120,
  CONTROL:= P#db14.dbx0.0 byte 30,
  DONE := M 70.0,
  ERROR := M 70.1,
  STATUS := MW 72);
```

**5.2.5.2 How DP_CTRL works**

**Sequence / handling on the call interface**

The DP_CTRL function call is processed within the cyclic execution of the user program as shown below:

The job is triggered with the first call. Diagnostic data is only returned in the acknowledgment of one of the subsequent calls.
After calling DP_CTRL, you obtain one of the following condition code patterns as the reaction:

- **DONE=0, ERROR=0, STATUS=8181**
  As long as the code combination DONE=0, ERROR=0 and STATUS=8181 is set, the job parameters must not be modified.

- **DONE=1**
  The parameter value DONE=1 indicates that the job was executed. Additional information is possible in the STATUS parameter.

- **DONE=0, ERROR=1**
  An error has occurred. The error message is located in STATUS.
5.2.5.3  Explanation of the formal parameters - DP_CTRL

Explanation of the formal parameters

The following table explains all the formal parameters for the DP_CTRL function:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Range of values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLADDR</td>
<td>INPUT</td>
<td>WORD</td>
<td></td>
<td>Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.</td>
</tr>
<tr>
<td>CONTROL</td>
<td>INPUT</td>
<td>ANY (only the following are permitted as VARTYPE: BYTE, WORD and DWORD)</td>
<td>The length must be set from 1 to 240</td>
<td>Specifies the address and length of the CONTROL job field Address of the data area. References the following alternatives: • PI area • Memory bit area • Data block area The length must be at least as long as the number of parameters.</td>
</tr>
<tr>
<td>DONE</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Job executed without error.</td>
<td>Indicates whether the job was sent and completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to DP_CTRL condition codes (Page 250)</td>
</tr>
<tr>
<td>ERROR</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>0: - 1: Error</td>
<td>Error code For the meaning in conjunction with the DONE and STATUS parameters, refer to DP_CTRL condition codes (Page 250)</td>
</tr>
<tr>
<td>STATUS</td>
<td>OUTPUT</td>
<td>WORD</td>
<td></td>
<td>Status code For the meaning in conjunction with the DONE and ERROR parameters, refer to DP_CTRL condition codes (Page 250)</td>
</tr>
</tbody>
</table>
Structure of the CONTROL job field

The control job has the following structure:

CONTROL address

<table>
<thead>
<tr>
<th>CTYPE</th>
<th>Param. 1st byte</th>
<th>Param. 2nd byte</th>
<th>Param. nth byte</th>
</tr>
</thead>
</table>

Refer to 'Parameter' (number and name)

Example of the job field

With a job field as shown below, a cyclic global control job SYNC and Unfreeze is sent for group 4 and group 5 without the autoclear option.

<table>
<thead>
<tr>
<th>DB 14</th>
<th>CTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0 01H</td>
<td>Command Mode</td>
</tr>
<tr>
<td>Byte 1 24H</td>
<td>Group Select</td>
</tr>
<tr>
<td>Byte 2 18H</td>
<td>Autoclear</td>
</tr>
<tr>
<td>Byte 3 00H</td>
<td></td>
</tr>
</tbody>
</table>

The length in the ANY pointer must be at least 4 (in the example, 30 has been selected).
5.2.5.4 Job types - DP_CTRL

Job types

Permitted or feasible specifications for the job are shown in the following overview based on the specification for CTYPE and the information in the job field.

<table>
<thead>
<tr>
<th>CTYPE</th>
<th>Corresponds to job</th>
<th>Parameter in job field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Name</td>
<td>Number</td>
</tr>
<tr>
<td>0</td>
<td>Trigger global control</td>
<td>1. byte: command mode 2nd byte: group select (See section following this table.)</td>
<td>2</td>
</tr>
</tbody>
</table>
| 1 *) | Trigger cyclic global control | 1. byte: command mode 2nd byte: group select 3rd byte: autoclear (See section following this table.) | 3 | The sending of cyclic global control jobs to the DP slaves selected with group select is triggered on the PROFIBUS CP.

The autoclear parameter is only evaluated with the SYNC global control job. If at least one DP slave in the selected group is not in the data transfer phase and autoclear=1 is set, the CLEAR mode is activated. In other words, the output data of the DP slaves is set to "0".

The following global jobs can be activated in the command mode parameter:
• SYNC
• FREEZE
• CLEAR (CLEAR-Bit = 1) - is not supported (please read the information in the manual as well)

or deactivated:
• UNSYNC
• UNFREEZE
• UNCLEAR (CLEAR bit = 0)

It is possible to specify more than one job in the command mode parameter.

An active cyclic global control job can only be terminated by a further global control job (cyclic or acyclic). To terminate the job set in the command mode, the job must be canceled. For example, the SYNC job is canceled by an UNSYNC job. |
<table>
<thead>
<tr>
<th>CTYPE</th>
<th>Corresponds to job</th>
<th>Parameter in job field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Delete older DP single diagnostic data</td>
<td>1. byte: Slave address 1 to 126 127 = all slaves</td>
<td>1</td>
</tr>
</tbody>
</table>
| 4     | Set current DP mode | 1. byte: RUN = 00H CLEAR = 01H OFFLINE = 03H RUN with AUTOCLEAR = 04H RUN without AUTOCLEAR = 04H | 1 | The DP mode can be set with this job as follows:  
  - RUN  
  - CLEAR  
  - OFFLINE  
  
The AUTOCLEAR parameter means that the DP master class 1 changes to the CLEAR status automatically when the following condition is met: at least one of the DP slaves with which the DP master class 1 wants to exchange data is not in the data transfer phase.  
The RUN without AUTOCLEAR parameter resets AUTOCLEAR.  
Notes:  
The STOP = 02H mode is no longer supported on the later modules (as of module type DA02). STOP = 02H is mapped to the OFFLINE mode. |
| 5     | Set DP mode for CPU STOP | 1. byte: RUN = 00H CLEAR = 01H OFFLINE = 03H | 1 | This job specifies which DP mode the PROFIBUS CP changes to if the CPU changes to STOP:  
  - RUN  
  - CLEAR  
  - OFFLINE  
  
As default, the PROFIBUS CP changes to the DP status CLEAR if the CPU changes to STOP.  
This mode remains set during a CP mode change from RUN --> STOP --> RUN.  
Notes:  
The STOP = 02H mode is no longer supported on the later modules (as of module type DA02). STOP = 02H is mapped to the OFFLINE mode. |
| 6     | Set DP mode for CP STOP | 1. byte: OFFLINE=03H | 1 | This job specifies which DP mode the PROFIBUS CP changes to if the CP changes to STOP:  
  - OFFLINE  
  
As default, the PROFIBUS CP changes to the DP status OFFLINE if the CP changes to STOP.  
This mode remains set during a CP mode change from RUN --> STOP --> RUN.  
Notes:  
The STOP = 02H mode is no longer supported on the later modules (as of module type DA02). STOP = 02H is mapped to the OFFLINE mode. |
### Program blocks for PROFIBUS

#### 5.2 Program blocks for DP (distributed I/O) with S7-300

<table>
<thead>
<tr>
<th>CTYPE</th>
<th>Corresponds to job</th>
<th>Parameter in job field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 *)</td>
<td>Read input data cyclically (DP master class 2)</td>
<td>1. byte: slave address 1 to 125</td>
<td>This job is not supported. Please read the information in the manual as well.</td>
</tr>
<tr>
<td>8 *)</td>
<td>Read output data cyclically (DP master class 2)</td>
<td>1. byte: slave address 1 to 125</td>
<td>This job is not supported. Please read the information in the manual as well.</td>
</tr>
<tr>
<td>9</td>
<td>Terminate cyclic processing of the DP slave by the DP master (class 1, class 2)</td>
<td>1. byte: slave address 1 to 125</td>
<td>This job terminates the cyclic reading of the input data or output data of the addressed DP slave or the data transfer (DP master class 1). The DP slave is then no longer processed by the PROFIBUS CP acting as DP master (class 2). This deactivates the DP slave.</td>
</tr>
<tr>
<td>10</td>
<td>Start cyclic processing as DP master (class 1)</td>
<td>1. byte: slave address 1 to 125</td>
<td>The PROFIBUS CP acting as the DP master (class 1) then assigns parameters to the addressed DP slave and starts cyclic data transfer (writing outputs/reading inputs). This activates the DP slave.</td>
</tr>
</tbody>
</table>

*) This CTYPE is no longer supported on the latest modules (as of module type DA02).

### 5.2.5.5 Command mode and group select - DP_CTRL

#### Structure of command mode

In the command mode parameter, you specify the modes for input and output data for the cyclic and acyclic global control jobs.

The meaning is as follows:

1 = activated
0 = not activated

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>not used</td>
</tr>
<tr>
<td>1</td>
<td>SYNC</td>
</tr>
<tr>
<td>2</td>
<td>UNSYNC</td>
</tr>
<tr>
<td>3</td>
<td>FREEZE</td>
</tr>
<tr>
<td>4</td>
<td>UNFREEZE</td>
</tr>
<tr>
<td>5</td>
<td>CLEAR</td>
</tr>
<tr>
<td>6</td>
<td>not used</td>
</tr>
</tbody>
</table>
Structure of group select

In the group select parameter, you specify the group to be addressed by the control job specified in the command mode parameter. The group select parameter occupies the second byte in the control job. Each bit defines a possible DP slave group.

The meaning is as follows:

1 = assigned
0 = not assigned

Bit number: 7 6 5 4 3 2 1 0
Group: [8 7 6 5 4 3 2 1]

5.2.5.6 DP_CTRL condition codes

Condition codes

The following table shows the return codes formed by the DONE, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

---

Table 5- 10 DP_CTRL condition codes

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Possible with CTYPE</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>8181H</td>
<td>0..10</td>
<td>Job active. DP master not started due to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• CP STOP or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• &quot;no parameter assignment&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The code described here occurs instead of one of the codes described</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>later:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 8183H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 8333H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 8334H</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0000H</td>
<td>0..10</td>
<td>Job completed without error.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8214H</td>
<td>0, 1</td>
<td>Job completed without error. Message: Cyclic global control job is sent as acyclic global control job</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8215H</td>
<td>0, 1</td>
<td>Job completed without error. The slaves addressed in the selected group are all deactivated.</td>
</tr>
</tbody>
</table>
### Program blocks for PROFIBUS

#### 5.2 Program blocks for DP (distributed I/O) with S7-300

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Possible with CTYPE</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>8219H</td>
<td>0, 1</td>
<td><strong>Job completed without error.</strong> An attempt was made to send an already active cyclic global control again. The global control continues unchanged.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8228H</td>
<td>0, 1</td>
<td><strong>Job completed without error.</strong> Message: The DP slaves addressed in the selected groups do not have any input modules.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8229H</td>
<td>0, 1</td>
<td><strong>Job completed without error.</strong> Message: The DP slaves addressed in the selected groups do not have any output modules.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8231H</td>
<td>4, 5, 6</td>
<td><strong>Job completed without error.</strong> Message: The DP status is already &quot;RUN&quot;</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8232H</td>
<td>4, 5, 6</td>
<td><strong>Job completed without error.</strong> Message: The DP status is already &quot;CLEAR&quot;</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8233H</td>
<td>4, 5, 6</td>
<td><strong>Job completed without error.</strong> Message: The DP status is already &quot;STOP&quot;</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8234H</td>
<td>4, 5, 6</td>
<td><strong>Job completed without error.</strong> Message: The DP status is already &quot;OFFLINE&quot;</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8235H</td>
<td>4</td>
<td><strong>Job completed without error.</strong> Message: The DP status is already &quot;RUN&quot; with activated AUTOCLEAR</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8236H</td>
<td>4</td>
<td><strong>Job completed without error.</strong> Message: The DP status is already &quot;RUN&quot; with deactivated AUTOCLEAR</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8241H</td>
<td>7-10</td>
<td><strong>Job completed without error.</strong> Message: The specified DP slave was not configured.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8243H</td>
<td>7-10</td>
<td><strong>Job completed without error.</strong> Message: The DP slave is already deactivated since the module list of the DP slave in the CP database only contains empty modules.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8245H</td>
<td>7-10</td>
<td><strong>Job completed without error.</strong> Message: The DP slave is already in the &quot;read input data cyclically&quot; mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8246H</td>
<td>7-10</td>
<td><strong>Job completed without error.</strong> Message: The DP slave is already in the &quot;read output data cyclically&quot; mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8248H</td>
<td>7-10</td>
<td><strong>Job completed without error.</strong> Message: The module list of the DP slave in the CP database contains input, output, or input/output modules.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8249H</td>
<td>7-10</td>
<td><strong>Job completed without error.</strong> Message: This slave is deactivated due to a change in the DP mode.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>824AH</td>
<td>7-10</td>
<td><strong>Job completed without error.</strong> Message: The DP slave is already deactivated due to a DP_CTRL job in the CPU program</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8090H</td>
<td>0..10</td>
<td>No module with this address exists.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8091H</td>
<td>0..10</td>
<td>Logical address not at a double word boundary.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B0H</td>
<td>0..10</td>
<td>The module does not recognize the data record.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80B1H</td>
<td>0..10</td>
<td>The specified data record length is incorrect.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C0H</td>
<td>0..10</td>
<td>The data record cannot be read.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C1H</td>
<td>0..10</td>
<td>The specified data record is currently being processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C2H</td>
<td>0..10</td>
<td>There are too many jobs pending.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>80C3H</td>
<td>Resources occupied (memory).</td>
<td></td>
</tr>
</tbody>
</table>
### Program blocks for PROFIBUS

#### 5.2 Program blocks for DP (distributed I/O) with S7-300

<table>
<thead>
<tr>
<th>DONE</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Possible with CTY</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>0..10</td>
<td>The DP master is not configured... Note: If the DP master is in &quot;STOP&quot; status, the status 8181H can also be output.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8184H</td>
<td></td>
<td>System error or illegal parameter type...</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8311H</td>
<td>0..10</td>
<td>CTY parameter outside the range of values</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8312H</td>
<td>0..10</td>
<td>The length of the area in the CONTROL parameter is too short.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8313H</td>
<td>3, 7, 8, 9, 10</td>
<td>The slave address parameter is outside the range of values.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8315H</td>
<td>0, 1</td>
<td>All DP slaves of the group specified in the global control are deactivated (always occurs with an empty group).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8317H</td>
<td>8</td>
<td>The length of the configured output data is greater than the configured receive area of the DP slave. Activating the slave mode &quot;Read output data&quot; is not possible.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8318H</td>
<td>0, 1, 4, 5, 6</td>
<td>The parameter 1st byte of the job data field is outside the range of values. With GLOBAL CONTROL, CLEAR was used with SYNC or a GLOBAL CONTROL with CLEAR set was sent to group 0.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>831AH</td>
<td>0, 1</td>
<td>At least one DP slave cannot handle FREEZE.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>831BH</td>
<td>0, 1</td>
<td>At least one DP slave cannot handle SYNC.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8333H</td>
<td>0, 1</td>
<td>This job is not permitted in the DP &quot;STOP&quot; mode. Note: If no DP master is configured, the status 8181H can also be output.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8334H</td>
<td>0, 1</td>
<td>This job is not permitted in the DP &quot;OFFLINE&quot; mode. Note: If no DP master is configured, the status 8181H can also be output.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8335H</td>
<td>0, 1</td>
<td>The PROFIBUS CP is in PROFIBUS status: &quot;Station not in ring&quot;.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8339H</td>
<td>0, 1</td>
<td>At least one DP slave in the selected group is not in the data transfer phase.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>833CH</td>
<td>1</td>
<td>Cyclic global control must not be used in the &quot;PLC &lt;-&gt; CP free running&quot; mode. This error does not occur on the CP 3425 because this mode is not possible with this CP (PBUS data records are always used for data transfer).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8341H</td>
<td>7-10</td>
<td>The specified DP slave was not configured.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8183H</td>
<td>0..10</td>
<td>DP master not configured.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8184H</td>
<td></td>
<td>System error or bad parameter type.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F22H</td>
<td>0..10</td>
<td>Area length error reading a parameter (e.g. DB too short).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F23H</td>
<td>0..10</td>
<td>Area length error writing a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F24H</td>
<td>0..10</td>
<td>Area error reading a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F25H</td>
<td>0..10</td>
<td>Area error writing a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F28H</td>
<td>0..10</td>
<td>Alignment error reading a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F29H</td>
<td>0..10</td>
<td>Alignment error writing a parameter.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F30H</td>
<td>0..10</td>
<td>The parameter is in the writeprotected first current data block.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F31H</td>
<td>0..10</td>
<td>The parameter is in the writeprotected second current data block.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F32H</td>
<td>0..10</td>
<td>Parameter contains a DB number that is too high.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F33H</td>
<td>0..10</td>
<td>DB number error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>8F3AH</td>
<td>0..10</td>
<td>Area not loaded (DB).</td>
</tr>
</tbody>
</table>
5.3 Configuration limits / resources required for the program blocks (PROFIBUS)

Required resources

Note

Note the version information of the blocks. The currently supplied block versions may differ from those shown here. Blocks with other versions have different resource requirements.

You will find information on the current block versions under entry ID:

Table 5-11 Information for FCs / FBs with S7400

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AG_SEND</td>
<td>1.2</td>
<td>FC5</td>
<td>732</td>
<td>576</td>
<td>540</td>
<td>20</td>
</tr>
<tr>
<td>AG_RECV</td>
<td>1.2</td>
<td>FC6</td>
<td>656</td>
<td>522</td>
<td>486</td>
<td>20</td>
</tr>
<tr>
<td>AG_LSEND</td>
<td>3.1</td>
<td>FC50</td>
<td>1044</td>
<td>846</td>
<td>810</td>
<td>52</td>
</tr>
<tr>
<td>AG_LRECV</td>
<td>3.1</td>
<td>FC60</td>
<td>1190</td>
<td>992</td>
<td>956</td>
<td>58</td>
</tr>
</tbody>
</table>
### Program blocks for PROFIBUS

#### 5.3 Configuration limits / resources required for the program blocks (PROFIBUS)

Table 5-12 Information for FCs / FBs with S7-300

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DP_SEND</td>
<td>3.0</td>
<td>FC1</td>
<td>1066</td>
<td>886</td>
<td>850</td>
<td>42</td>
</tr>
<tr>
<td>DP_RECV</td>
<td>3.0</td>
<td>FC2</td>
<td>1144</td>
<td>950</td>
<td>914</td>
<td>46</td>
</tr>
<tr>
<td>DP_DIAG</td>
<td>3.0</td>
<td>FC3</td>
<td>1956</td>
<td>1638</td>
<td>1602</td>
<td>58</td>
</tr>
<tr>
<td>DP_CTRL</td>
<td>3.0</td>
<td>FC4</td>
<td>1532</td>
<td>1292</td>
<td>1256</td>
<td>52</td>
</tr>
<tr>
<td>AG_SEND</td>
<td>4.2</td>
<td>FC5</td>
<td>1976</td>
<td>1664</td>
<td>1628</td>
<td>50</td>
</tr>
<tr>
<td>AG_RECV</td>
<td>4.7</td>
<td>FC6</td>
<td>1440</td>
<td>1206</td>
<td>1170</td>
<td>40</td>
</tr>
</tbody>
</table>
6.1 Overview of uses

Overview

The following function blocks are available for an S7 station involved in FMS communication. The list shows the block numbers as they are when supplied. You can change these block numbers.

<table>
<thead>
<tr>
<th>Function block</th>
<th>Can be used in the function of the PROFIBUS CP as:</th>
<th>Meaning / function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Block number</td>
<td>FMS client</td>
</tr>
<tr>
<td>IDENTIFY</td>
<td>FB2</td>
<td>X</td>
</tr>
<tr>
<td>READ</td>
<td>FB3</td>
<td>X</td>
</tr>
<tr>
<td>REPORT</td>
<td>FB4</td>
<td>-</td>
</tr>
<tr>
<td>STATUS</td>
<td>FB5</td>
<td>X</td>
</tr>
<tr>
<td>WRITE</td>
<td>FB6</td>
<td>X</td>
</tr>
</tbody>
</table>

Difference between S7300 and S7400

Different FBs are supplied for the S7300 and S7400. Make sure you access the appropriate block library (SIMATIC_NET_CP) depending on whether you are creating a user program for an S7300 or an S7400.
6.2 FMS block parameters

FB call interfaces

The following sections describe the call interface for each FB as shown below:

Depending on the FB type, the FB has different parameters of the type INPUT, OUTPUT or INOUT.

The following tables explain the meaning, data type, range of values and memory area of all block parameters.

INPUT parameters

<table>
<thead>
<tr>
<th>INPUT parameters</th>
<th>Meaning</th>
<th>Data type</th>
<th>Value range/memory area</th>
<th>Used in FB</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ</td>
<td>Edge signal for executing the block</td>
<td>BOOL</td>
<td>0=FALSE; 1=TRUE 0-&gt;1: &quot;Start&quot;/I,Q,M,D,L</td>
<td>2 3 4 5 6</td>
</tr>
<tr>
<td>ID</td>
<td>This identifier identifies the FMS connection.</td>
<td>DWORD</td>
<td>0001 0001 .. FFFF FFFF / I,Q,M,D,L</td>
<td>2 3 4 5 6</td>
</tr>
</tbody>
</table>
### INPUT parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Data type</th>
<th>Value range/memory area</th>
<th>Used in FB</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR_1</td>
<td>The parameter addresses the remote communications variable to be read or written. Depending on the configuration on the FMS server, a name or index can be specified.</td>
<td>ANY</td>
<td>String: Max. length = 254 bytes e.g. <code>&lt;102&gt;</code> (index access) &quot;SLAVE2&quot; (named access) DB</td>
<td>2 3 4 - 6</td>
</tr>
<tr>
<td>SD_1</td>
<td>Address of a local data area from which the variables will be transferred.</td>
<td>ANY</td>
<td>This type corresponds to a reference to a DB, I/O process image or bit memory area. Example: SD_1 := P#DB17.DBX0.0 BYTE 16 In this example, the first 16 bytes of DB17 are transferred. I,Q,M,D,L,C,T,Dbx</td>
<td>- - 4 - 6</td>
</tr>
<tr>
<td>RD_1</td>
<td>Address of a local data area to which the variables will be transferred.</td>
<td>ANY</td>
<td>This type corresponds to a reference to a DB, I/O process image or bit memory area. Example: SD_1 := P#DB17.DBX0.0 BYTE 16 In this example, the first 16 bytes of DB17 are transferred. I,Q,M,D,L,Dbx Note on array of bytes for S7-300: If there is an odd number of bytes to be read, the length of the receive area must be configured up to the next higher even number of bytes. Example: For an array[1..13] of bytes, reserve a receive buffer size of 14 bytes.</td>
<td>- 3 - -</td>
</tr>
</tbody>
</table>
## OUTPUT parameters

<table>
<thead>
<tr>
<th>OUTPUT parameters</th>
<th>Meaning</th>
<th>Data type</th>
<th>Range of values/mem. area</th>
<th>Used in FB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DONE</td>
<td>Indicates that the job is completed.</td>
<td>BOOL</td>
<td>0=FALSE</td>
<td>- - 4 - 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1=TRUE: Job completed;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I,Q,M,D,L</td>
<td></td>
</tr>
<tr>
<td>NDR</td>
<td>Indicates reception of data.</td>
<td>BOOL</td>
<td>0=FALSE</td>
<td>2 3 - 5 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1=TRUE: New data were accepted;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I,Q,M,D,L</td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>Indicates whether or not an error occurred.</td>
<td>BOOL</td>
<td>0=FALSE</td>
<td>2 3 4 5 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1=TRUE: Error occurred;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I,Q,M,D,L</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>Provides detailed information about warnings or errors after the job has been completed.</td>
<td>WORD</td>
<td>You will find detailed decoding information in the sections following.</td>
<td>2 3 4 5 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I,Q,M,D,L</td>
<td></td>
</tr>
</tbody>
</table>

## INPUT/OUTPUT parameters

<table>
<thead>
<tr>
<th>INOUT parameters</th>
<th>Meaning</th>
<th>Data type</th>
<th>Range of values/mem. area</th>
<th>Used in FB</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHYS</td>
<td>Indicates the physical status of the partner device (VFD).</td>
<td>BYTE</td>
<td>0...3</td>
<td>- - - 5 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I,Q,M,D,L</td>
<td></td>
</tr>
<tr>
<td>LOG</td>
<td>Indicates the logical status of the partner (VFD).</td>
<td>BYTE</td>
<td>0...3</td>
<td>- - - 5 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I,Q,M,D,L</td>
<td></td>
</tr>
<tr>
<td>LOCAL</td>
<td>&quot;local detail&quot; parameter of the partner</td>
<td>ANY</td>
<td>This detail can be up to 16 bytes long.</td>
<td>- - - 5 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I,Q,M,D,L</td>
<td></td>
</tr>
<tr>
<td>VENDOR</td>
<td>Name of the device vendor.</td>
<td>STRING</td>
<td>Length&lt;255</td>
<td>2 - - - -</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>MODEL</td>
<td>Name of the device model.</td>
<td>STRING</td>
<td>Length&lt;255</td>
<td>2 - - - -</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>REVISION</td>
<td>Version (revision) of the device.</td>
<td>STRING</td>
<td>Length&lt;255</td>
<td>2 - - - -</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

### Memory area

The abbreviated forms for the memory areas in the table correspond to the following:

<table>
<thead>
<tr>
<th>Short form</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
</tr>
<tr>
<td>Q</td>
<td>Output</td>
</tr>
<tr>
<td>M</td>
<td>Bit memory</td>
</tr>
</tbody>
</table>
### Short form, Type

<table>
<thead>
<tr>
<th>Short form</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Temporary local data</td>
</tr>
<tr>
<td>D</td>
<td>Data block area</td>
</tr>
<tr>
<td>C</td>
<td>Counter</td>
</tr>
<tr>
<td>T</td>
<td>Timer</td>
</tr>
<tr>
<td>DBX</td>
<td>Data block</td>
</tr>
</tbody>
</table>

### FB output parameters during the CP startup (S7400)

When the FB is called (REQ:0->1, EN_R=1) while the PROFIBUS CP is starting up (for example due to a power cycle or activating a switch) the following output parameters are possible:

- **DONE = 0**
- **NDR = 0**
- **ERROR = 1**
- **STATUS = 0001** (connection has not been established yet) or **STATUS = 0607** (Get-OV still running)

### 6.3 IDENTIFY

#### 6.3.1 Meaning and call - IDENTIFY

**Meaning of the block**

With the IDENTIFY function block, you can fetch the following information about the partner device (with S7 stations about the CPU):

- Name of the device vendor.
- Name of the device model.
- Version (revision) of the device.

Depending on the information you receive, you could, for example:

- Set the local program function to match the performance and response of the partner
- Set communication parameters
6.3 IDENTIFY

Call interface

Example in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call FB 2, DB 22 {</td>
<td>//IDENTIFY block call with instance DB</td>
</tr>
<tr>
<td>REQ := M 1.0,</td>
<td>//Edge signal for executing the FB</td>
</tr>
<tr>
<td>ID := DW#16#10001,</td>
<td>//Matched with configuration of the FMS connection</td>
</tr>
<tr>
<td>NDR := M 1.1,</td>
<td>//Indicates when “new data accepted”</td>
</tr>
<tr>
<td>ERROR := M 1.2,</td>
<td>//Indicates error in execution</td>
</tr>
<tr>
<td>STATUS := MW 20,</td>
<td>//Detailed error decoding</td>
</tr>
<tr>
<td>VENDOR := &quot;SLAVE2&quot;.VENDOR_ABBILD,</td>
<td>//Data area for vendor name</td>
</tr>
<tr>
<td>MODEL := &quot;SLAVE2&quot;.MODEL_ABBILD,</td>
<td>//Data area for model</td>
</tr>
<tr>
<td>REVISION := &quot;SLAVE2&quot;.REV_ABBILD );</td>
<td>//Data area for revision</td>
</tr>
</tbody>
</table>

Additional information

"SLAVE2"
is the symbolic name of a data block. This name is defined in the corresponding symbols table.
VENDOR_IMAGE, MODEL_IMAGE and REVISION_IMAGE are variables of the data type STRING. These are defined in the "SLAVE2" data block.

6.3.2 How IDENTIFY works

Operating principle

The following flow chart illustrates the normal sequence of an IDENTIFY job.
The job is activated by a (positive-going) edge change at the parameter REQ.
Each IDENTIFY job of the user program is acknowledged by the PROFIBUS CP with a value in the output parameters NDR, ERROR and STATUS.
6.4 READ

6.4.1 Meaning and call - READ

Meaning

The READ function block reads data from a data area of the communication partner specified by a name or index depending on the assignment of parameters for the job. The data that is read is saved locally in a data block, an area in the process image of the inputs/outputs or in a bit memory area.

Requirement: Configuration of communications variables

The structure of the variables on the communications partner (FMS server) is fixed. When the FMS connection is established, the structure description is read out from the communications partner. This is then available on the PROFIBUS CP to convert the data to the FMS representation.
The structure description is only read when the connection is established if the communications variable was selected during configuration of the FMS connection.

### Access rights

Remember that access rights can be set for the data transfer. Data transmission is then only possible if the FMS client has been assigned suitable rights.

### FB call interface

The following sequence chart shows the normal sequence of data reception triggered with READ in the user program.

The job is activated by a (positive-going) edge change at the parameter REQ.

Every READ job in the user program is acknowledged by the PROFIBUS CP with values in the output parameters NDR, ERROR and STATUS.

#### Example in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call FB 3, DB 29 {</td>
<td>//READ block call with instance DB</td>
</tr>
<tr>
<td>REQ := M 1.0,</td>
<td>//Signal edge change to execute the FB</td>
</tr>
<tr>
<td>ID := DW16#10001,</td>
<td>//Compared with configuration of</td>
</tr>
<tr>
<td></td>
<td>//FMS connection</td>
</tr>
<tr>
<td>VAR_1 := &quot;SLAVE2&quot;.INDEX,</td>
<td>//Addresses K variable that will be read</td>
</tr>
<tr>
<td>RD_1 := &quot;PROZESS&quot;.Motor1,</td>
<td>//Addresses data area as destination</td>
</tr>
<tr>
<td>NDR := M 1.1,</td>
<td>//Confirmation of execution</td>
</tr>
<tr>
<td>ERROR := M 1.2,</td>
<td>//Indicates incorrect execution</td>
</tr>
<tr>
<td>STATUS := MW 20 };</td>
<td>//Detailed error decoding</td>
</tr>
</tbody>
</table>

### 6.4.2 How READ works

#### Operating principle

The following sequence chart shows the normal sequence of data reception triggered with READ in the user program.

The job is activated by a (positive-going) edge change at the parameter REQ.

Every READ job in the user program is acknowledged by the PROFIBUS CP with values in the output parameters NDR, ERROR and STATUS.
Guarantee of data transfer

The diagram shows that the reading out of the data is confirmed with the code NDR=1, ERROR=0 and STATUS=0000.

Positive confirmation of the read job does not necessarily mean that the read job was registered by the partner application.

6.5 REPORT

6.5.1 Meaning and call - REPORT

Meaning of the block

The REPORT function block allows unconfirmed transmission of variables by an FMS server. This job type is used particularly for transmission on broadcast/multicast FMS connections.

The structure of the variables to be reported must be configured locally on the FMS server.
S7 station as communications partner

To allow the reported variables to be accepted by the communications partner, the variables must be entered during configuration of the communications partner (FMS client).

Call interface

Example in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call FB 4, DB 28 {</td>
<td></td>
</tr>
<tr>
<td>REQ := M 1.0,</td>
<td>//REPORT block call with instance DB</td>
</tr>
<tr>
<td>ID := DW#16#10001,</td>
<td>//Signal edge change to execute the FB</td>
</tr>
<tr>
<td>VAR_1 := &quot;SLAVE2&quot;.INDEX,</td>
<td>//Compared with configuration of FMS connection</td>
</tr>
<tr>
<td>SD_1 := &quot;PROZESS&quot;.Motor1,</td>
<td>//Names the C variable to be reported</td>
</tr>
<tr>
<td>DONE := M 1.1,</td>
<td>//Addresses the source data area</td>
</tr>
<tr>
<td>ERROR := M 1.2,</td>
<td>//Confirmation of execution</td>
</tr>
<tr>
<td>STATUS := MW 20 };</td>
<td>//Indicates incorrect execution</td>
</tr>
</tbody>
</table>

Note

The parameter SD_1 is used to address the data area from which the variable values are read and reported. According to the FMS conventions, you also need to specify the variable index on the FC interface. The consistency of this information is not, however, checked when the call is executed.

6.5.2 How REPORT works

Operating principle

The following flow chart shows the normal sequence of a data transfer transferred by REPORT in the user program.

The job is activated by a (positive-going) edge change at the parameter REQ.
Every REPORT job in the user program is acknowledged by the PROFIBUS CP with values in the output parameters DONE, ERROR and STATUS.

6.6 STATUS

6.6.1 Meaning and call - STATUS

Meaning of the block

The STATUS function block allows status information to be requested from the communications partner on the specified FMS connection.
The following information is available:

- The logical status of the VFD; for example information whether communication is possible.
- The physical status of the VFD; information about the status of the device.
- Device-specific information; normally provides vendor-specific information.

The following table provides information about the codes that a device can supply as a result of the status request:

<table>
<thead>
<tr>
<th>Device</th>
<th>Message version</th>
<th>Log</th>
<th>Phys</th>
<th>Local detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>S7 with PROFIBUS CP</td>
<td>1</td>
<td>00H: Ready for communication CP in RUN, CPU in RUN</td>
<td>10H: Ready for communication, CPU in RUN</td>
<td>No entry</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>02H: Limited services, CP in RUN, CPU in STOP</td>
<td>13H: Maintenance required, CPU in STOP</td>
<td>No entry</td>
</tr>
<tr>
<td>Third-party device</td>
<td>The following are possible:</td>
<td>00H: Ready for communication</td>
<td>10H: Operational</td>
<td>- vendor specific -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>02H: Number of services limited</td>
<td>11H</td>
<td>Partly operational</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12H</td>
<td>Not operational</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13H</td>
<td>Maintenance required</td>
</tr>
</tbody>
</table>

**Call interface**

![Call interface diagram](image-url)
Example in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call FB 5, DB 21 (</td>
<td>//STATUS block call with instance DB</td>
</tr>
<tr>
<td>REQ := M 1.0,</td>
<td>//Signal edge change to execute the FB</td>
</tr>
<tr>
<td>ID := DW#16#10001,</td>
<td>//Compared with configuration of FMS connection</td>
</tr>
<tr>
<td>NDR := M 1.1,</td>
<td>//Indicates when new data is accepted</td>
</tr>
<tr>
<td>ERROR := M 1.2,</td>
<td>//Indicates incorrect execution</td>
</tr>
<tr>
<td>STATUS := MW 20,</td>
<td>//Detailed error decoding</td>
</tr>
<tr>
<td>PHYS := MB 22,</td>
<td>//Data area for physical status</td>
</tr>
<tr>
<td>LOG := MB 23,</td>
<td>//Data area for logical status</td>
</tr>
<tr>
<td>LOCAL := P#DB18.DBX0.0 WORD8 );</td>
<td>//Data area for &quot;local detail&quot;</td>
</tr>
</tbody>
</table>

6.6.2 How STATUS works

Operating principle

The following flow chart shows the normal sequence of a STATUS job.

The job is activated by a (positive-going) edge change at the parameter REQ.

Every STATUS job in the user program is confirmed by the PROFIBUS CP with values in the output parameters NDR, ERROR and STATUS.

![Flow chart showing the normal sequence of a STATUS job](image)
6.7 WRITE

6.7.1 Meaning and call - WRITE

Meaning
The WRITE FB transfers data from a specified local data area to a data area on the communication partner. The local data area can be a data block, an area in the process input or output image or a bit memory area. (See also parameter SD_1, FMS block parameter (Page 256))

The data area of the communication partner is specified using a variable name or a variable index.

Requirement: Configuration of communications variables
The structure of the variables on the communications partner (FMS server) is fixed. When the FMS connection is established, the structure description is read out from the communications partner. This is then available on the PROFIBUS CP to convert the data to the FMS representation.

The structure description is only read when the connection is established if the communications variable was selected during configuration of the FMS connection.

Access rights
Remember that access rights can be set for the data transfer. Data transmission is then only possible if the FMS client has been assigned suitable rights.

Call interface

WRITE

<table>
<thead>
<tr>
<th>BOOL</th>
<th>REQ</th>
<th>DONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWORD</td>
<td>ID</td>
<td>ERROR</td>
</tr>
<tr>
<td>ANY</td>
<td>VAR_1</td>
<td>STATUS</td>
</tr>
<tr>
<td>ANY</td>
<td>SD_1</td>
<td></td>
</tr>
</tbody>
</table>
Example in STL representation

<table>
<thead>
<tr>
<th>STL</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>call FB 6, DB 28</td>
<td>WRITE block call with instance DB</td>
</tr>
<tr>
<td>REQ := M 1.0,</td>
<td>Signal edge change to execute the FB</td>
</tr>
<tr>
<td>ID := DW#16#10001,</td>
<td>Compared with configuration of FMS connection</td>
</tr>
<tr>
<td>VAR_1 := &quot;SLAVE2&quot;.INDEX,</td>
<td>Names the C variable to be written</td>
</tr>
<tr>
<td>SD_1 := &quot;PROZESS&quot;.Motor1,</td>
<td>Addresses the source data area</td>
</tr>
<tr>
<td>DONE := M 1.1,</td>
<td>Confirmation of execution</td>
</tr>
<tr>
<td>ERROR := M 1.2,</td>
<td>Indicates incorrect execution</td>
</tr>
<tr>
<td>STATUS := MW 20</td>
<td>Detailed error decoding</td>
</tr>
</tbody>
</table>

### 6.7.2 How WRITE works

**Operating principle**

The following flow chart shows the normal sequence of a data transfer triggered with WRITE in the user program.

The job is activated by a (positive-going) edge change at the parameter REQ.

Every WRITE job in the user program is confirmed by the PROFIBUS CP with values in the output parameters DONE, ERROR and STATUS.
Guarantee of data transfer

The diagram also shows that with the confirmation DONE=1, ERROR=0 and STATUS=0000, data transfer to the communications partner and entry in the remote data area is functioning correctly.

A positive confirmation of the job does not necessarily mean that the data has already been received and processed by the partner application.

6.8 Condition codes and error messages - FMS blocks

Structure of the tables

The following tables explain the condition codes and error codes that must be handled in your user program. The meanings of the parameters DONE/NDR, ERROR and STATUS are explained in FMS block parameters (Page 256)

To provide a better overview, the error codes are listed as follows:

<table>
<thead>
<tr>
<th>Error detected locally</th>
<th>Error detected by FMS partner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouped according to:</td>
<td></td>
</tr>
<tr>
<td>- Error class (explanation, see table below)</td>
<td></td>
</tr>
<tr>
<td>- Error code / meaning (see table below)</td>
<td></td>
</tr>
</tbody>
</table>

Error-free job execution

If the job was executed free of errors, the parameters on the FB interface have the following values:

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0x0000</td>
<td>Job completed without errors</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0x000B</td>
<td>Job active</td>
</tr>
</tbody>
</table>
**Error classes**

The possible error codes are grouped into the following error classes:

<table>
<thead>
<tr>
<th>Error class</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>Indicates errors or problems involving the following:</td>
</tr>
<tr>
<td></td>
<td>• FB parameter assignment</td>
</tr>
<tr>
<td></td>
<td>• Block execution in the CPU and CP</td>
</tr>
<tr>
<td>Application</td>
<td>Indicates errors or problems on the interface between the user program and FB.</td>
</tr>
<tr>
<td>Definition</td>
<td>Indicates errors that usually involve inconsistencies between the user program and FMS configuration.</td>
</tr>
<tr>
<td>Components</td>
<td>Indicates resource problems on the PROFIBUS CP.</td>
</tr>
<tr>
<td>Service</td>
<td>Indicates errors or problems in conjunction with the requested FMS service.</td>
</tr>
<tr>
<td>Access</td>
<td>Indicates denied access to objects due to the following:</td>
</tr>
<tr>
<td></td>
<td>• Absence of access rights</td>
</tr>
<tr>
<td></td>
<td>• Hardware problems</td>
</tr>
<tr>
<td></td>
<td>• Other inconsistencies</td>
</tr>
<tr>
<td>OD (object directory)</td>
<td>Indicates problems accessing the object dictionary of the VFD.</td>
</tr>
<tr>
<td>VFD status</td>
<td>Unspecified error on the VFD</td>
</tr>
<tr>
<td>otherwise</td>
<td>Other errors</td>
</tr>
</tbody>
</table>

**6.8.1 Error detected locally**

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0001</td>
<td>Communications problem: For example, communications bus connection not established.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0002</td>
<td>The function cannot be executed: Either negative acknowledgment from CP or error in the sequence, for example communications bus error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0003</td>
<td>The connection is not configured (invalid ID specified). If the connection is configured, the error message indicates that the permitted parallel job processing limit has been exceeded. Example: SAC=0 is configured and a REPORT job is sent.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0004</td>
<td>The receive data area is too short or the data types do not match.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0005</td>
<td>A reset request has been received from the CP (BRCV).</td>
</tr>
</tbody>
</table>
### 6.8 Condition codes and error messages - FMS blocks

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0006</td>
<td>The corresponding job execution on the CP is in the DISABLED state or a reset request has been received from the CP; the transfer is therefore incomplete.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0007</td>
<td>Corresponding job execution on the CP is in the wrong state. For REPORT: The error is specified in greater detail in the diagnostic buffer.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0008</td>
<td>Job execution on the CP signals an error accessing the user memory.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x000A</td>
<td>Access to local user memory not possible (for example, the DB was deleted).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x000C</td>
<td>When the underlying BSEND or BRCV SFBs were called, an instance DB that does not belong to SFB12/SFB13 was specified or no instance DB was used, but rather a global DB.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0014</td>
<td>Not enough work or load memory available.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0200</td>
<td>Unspecified application reference error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0201</td>
<td>The configured connection cannot be established at present; for example LAN connection not established.</td>
</tr>
</tbody>
</table>

**Table 6-3** "Definition" error class

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0300</td>
<td>Unspecified definition error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0301</td>
<td>Object with requested index/name is not defined.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0302</td>
<td>Object attributes are inconsistent.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0303</td>
<td>Name exists already.</td>
</tr>
</tbody>
</table>

**Table 6-4** "Resources" error class

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0400</td>
<td>Unspecified resource error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0401</td>
<td>No memory available.</td>
</tr>
</tbody>
</table>

**Table 6-5** "Service" error class

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0500</td>
<td>Unspecified service error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0501</td>
<td>Conflict due to object status.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0502</td>
<td>Configured PDU size exceeded.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0503</td>
<td>Conflict due to object restrictions.</td>
</tr>
</tbody>
</table>
### Program blocks for PROFIBUS FMS

#### 6.8 Condition codes and error messages - FMS blocks

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0504</td>
<td>Inconsistent parameters.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0505</td>
<td>Illegal parameters.</td>
</tr>
</tbody>
</table>

**Table 6-6  “Access” error class**

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0600</td>
<td>Unspecified access error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0601</td>
<td>Invalid object or no OD loaded;</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0602</td>
<td>Hardware fault</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0603</td>
<td>Object access was denied.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0604</td>
<td>Invalid address.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0605</td>
<td>Inconsistent object attributes.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0606</td>
<td>Object access not supported.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0607</td>
<td>Object does not exist in OD or GetOD still active.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0608</td>
<td>Type conflict or variable content outside permitted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>range of values</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0609</td>
<td>Access using names not supported.</td>
</tr>
</tbody>
</table>

**Table 6-7  “Object dictionary” (OD) error class / VFD Status/Reject error class**

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0700</td>
<td>Unspecified OD error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0701</td>
<td>Permitted name length exceeded.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0702</td>
<td>Overflow of the object dictionary.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0703</td>
<td>Object dictionary is write protected.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0704</td>
<td>Overflow of the extension length.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0705</td>
<td>Overflow of the object description length.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0706</td>
<td>Processing problem.</td>
</tr>
</tbody>
</table>

**Table 6-8  “Other” error class**

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0100</td>
<td>Unspecified VFD status error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0108</td>
<td>RCC/SAC/RAC error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0106</td>
<td>Service not supported.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0105</td>
<td>PDU length error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x0102</td>
<td>Bad FMS-PDU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x0800</td>
<td>Unspecified error.</td>
</tr>
</tbody>
</table>
6.8.2 Errors detected by FMS partner

Table 6-9 Application error class

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x8200</td>
<td>Unspecified application reference error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8201</td>
<td>Application (e.g. user program) cannot be reached.</td>
</tr>
</tbody>
</table>

Table 6-10 Definition error class

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x8300</td>
<td>Unspecified definition error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8301</td>
<td>Object with requested index/name is not defined.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8302</td>
<td>Object attributes are inconsistent.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8303</td>
<td>Name exists already.</td>
</tr>
</tbody>
</table>

Table 6-11 Resources error class

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x8400</td>
<td>Unspecified resource error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8401</td>
<td>No memory available.</td>
</tr>
</tbody>
</table>

Table 6-12 Service error class

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x8500</td>
<td>Unspecified service error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8501</td>
<td>Conflict due to object status.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8502</td>
<td>Configured PDU size exceeded.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8503</td>
<td>Conflict due to object restrictions.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8504</td>
<td>Inconsistent parameters.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8505</td>
<td>Illegal parameters.</td>
</tr>
</tbody>
</table>

Table 6-13 Access error class

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x8600</td>
<td>Unspecified access error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8601</td>
<td>Invalid object.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8602</td>
<td>Hardware error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8603</td>
<td>Object access was denied.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8604</td>
<td>Invalid address.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8605</td>
<td>Inconsistent object attributes.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8606</td>
<td>Object access is not supported.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8607</td>
<td>Object does not exist.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8608</td>
<td>Type conflict or variable content outside permitted range of values</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8609</td>
<td>Access using names is not supported.</td>
</tr>
</tbody>
</table>
### 6.9 Quantity framework / resource requirements of FBs (PROFIBUS FMS)

Table 6-14  OD (object dictionary) error class

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x8700</td>
<td>Unspecified OD error.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8701</td>
<td>Permitted name length exceeded.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8702</td>
<td>Overflow of the object dictionary.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8703</td>
<td>Object dictionary is write protected.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8704</td>
<td>Overflow of the extension length.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8705</td>
<td>Overflow of the object description length.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x8706</td>
<td>Processing problem.</td>
</tr>
</tbody>
</table>

Table 6-15  VFD status error class / "Other" error class

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x8100</td>
<td>Unspecified VFD status error.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DONE/NDR</th>
<th>ERROR</th>
<th>STATUS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x8000</td>
<td>Unspecified error detected by partner.</td>
</tr>
</tbody>
</table>
6.9 Quantity framework / resource requirements of FBs (PROFIBUS FMS)

Note
Note the version information of the blocks. The currently supplied block versions may differ from those shown here. Blocks with other versions have different resource requirements.

You will find information on the current block versions under entry ID:


<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IDENT</td>
<td>1.4</td>
<td>2</td>
<td>1658</td>
<td>1364</td>
<td>1328</td>
<td>136</td>
<td>464</td>
<td>196</td>
</tr>
<tr>
<td>READ</td>
<td>1.5</td>
<td>3</td>
<td>2474</td>
<td>2086</td>
<td>2050</td>
<td>130</td>
<td>606</td>
<td>338</td>
</tr>
<tr>
<td>REPORT</td>
<td>1.5</td>
<td>4</td>
<td>2184</td>
<td>1818</td>
<td>1782</td>
<td>156</td>
<td>588</td>
<td>332</td>
</tr>
<tr>
<td>STATUS</td>
<td>1.3</td>
<td>5</td>
<td>1656</td>
<td>1390</td>
<td>1354</td>
<td>112</td>
<td>438</td>
<td>190</td>
</tr>
<tr>
<td>WRITE</td>
<td>1.5</td>
<td>6</td>
<td>2486</td>
<td>2094</td>
<td>2058</td>
<td>142</td>
<td>632</td>
<td>358</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IDENT</td>
<td>1.6</td>
<td>2</td>
<td>1462</td>
<td>1254</td>
<td>1218</td>
<td>86</td>
<td>306</td>
<td>158</td>
</tr>
<tr>
<td>READ</td>
<td>1.5</td>
<td>3</td>
<td>1998</td>
<td>1700</td>
<td>1664</td>
<td>64</td>
<td>218</td>
<td>70</td>
</tr>
<tr>
<td>REPORT</td>
<td>1.6</td>
<td>4</td>
<td>2036</td>
<td>1732</td>
<td>1696</td>
<td>76</td>
<td>230</td>
<td>72</td>
</tr>
<tr>
<td>STATUS</td>
<td>1.6</td>
<td>5</td>
<td>1430</td>
<td>1244</td>
<td>1208</td>
<td>60</td>
<td>182</td>
<td>46</td>
</tr>
<tr>
<td>WRITE</td>
<td>1.6</td>
<td>6</td>
<td>2028</td>
<td>1724</td>
<td>1688</td>
<td>76</td>
<td>230</td>
<td>72</td>
</tr>
</tbody>
</table>
This chapter provides an overview of the previous issues of this manual and the functional expansions of the program blocks.

This was new in release 07
This manual release contained corrections and new information.

Changes / additions:
- Names of the PROFIenergy program blocks for the S7-300 in STEP 7 Professional
- Configuration limits / resources required for the program blocks for Ethernet and PROFIBUS FMS
- Reaction of the SIMATIC NET CPs to ICMP frames

This was new in release 06
This manual release contained corrections.

Corrections:
- PROFIenergy program blocks

This was new in release 05
This manual release contained corrections and new information.

Changes / additions:
- PROFIenergy

    New program blocks for PROFIenergy functions

This was new in release 04
This manual release contained corrections and new information.

Changes / additions:
- New program block AG_CNTEX
- Adaptation to the new STEP 7 Professional configuration software

    In the new configuration software, program blocks are always specified by their symbolic names. This manual follows this convention by using the names of the program blocks in most situations. The assignment of the names and block numbers as used in STEP 7 V5.5 is nevertheless retained.

- Expanded program block FTP_CMD

    You can set up SSL-secured FTP connections.
This was new in release 03

This manual release contained corrections and new information.

Changes / additions:

- Reference to FAQ entry on block handling.
- Special features when using the FB for programmed connections and IP configuration in fault-tolerant systems (H systems)
- DP_SEND / DP_RECV explanation of status code $8180_{16}$ expanded.
- DP_CTRL: The missing description of job type CTYPE = 4 was added.
  Information on CTYPE 1, 7 and 8 was added because these are no longer supported as of module type DA02.
- DP_CTRL: Explanation of the following status codes was expanded: $8181_{16}$, $8183_{16}$, $8333_{16}$, $8334_{16}$

This was new in release 02

This manual release contained corrections and new information.
The following was added to the previous release:

Changes / additions:

- In the section on Industrial Ethernet
  A new function block FB56 is available for ERPC communication with ERPC-CPs.
- In the section on FBs / FCs for FTP services
  The description of the data block file DB is now both in the online help and in this manual.
- In the section on FBs for programmed connections and IP configuration
  The section now includes a full description of the required configuration data block.
  The option of configuring a device name for certain CP types has also been added here.
- In the section on FBs / FCs for PROFINET IO
  Here, the option of transferring detailed status information with the FCs PNIO_SEND and PNIO_RECV has been added. This allows you to achieve a faster reaction on the interface.

This was new in release 01

This manual grouped together the block descriptions that were previously in the manuals for S7 CPs structured according to network types. These manuals will no longer include the block descriptions.
Compared with the previously valid versions of the manuals for S7 CPs, version 1 includes the following innovations in the block descriptions:

- In the section on Industrial Ethernet
  
  A new function block FB 40 is available for the FTP client mode of advanced CPs. Using this FB, complete FTP job sequences can be created efficiently in the user program.

- In the section on PROFINET IO
  
  Parameter modification in the blocks for PROFINET IO
  
  - FC11 PNIO_SEND (block version 2.0)
  - FC12 PNIO_RECV (block version 2.0)

  These two functions must be used for CPs that use PROFINET IO controller and device mode at the same time.
Finding the SIMATIC NET documentation

- **Catalogs**
  
  You will find the order numbers for the Siemens products of relevance here in the following catalogs:
  
  - SIMATIC NET Industrial Communication / Industrial Identification, catalog IK PI
  - SIMATIC Products for Totally Integrated Automation and Micro Automation, catalog ST 70

  You can request the catalogs and additional information from your Siemens representative.

  You can go to the Industry Mall on the Internet at the following address:
  
  Link: [https://mall.industry.siemens.com](https://mall.industry.siemens.com)

- **Documentation on the Internet**

  You will find SIMATIC NET manuals on the Internet pages of Siemens Automation Customer Support:


  Go to the required product group and make the following settings:

  "Entry list" tab, Entry type "Manuals / Operating Instructions"

- **Documentation from the STEP 7 installation**

  Manuals that are included in the online documentation of the STEP 7 installation on your PG/PC can be found in the start menu ("Start" > "All Programs" > "Siemens Automation" > "Documentation").

---

**B.1 On configuring, commissioning and using the CP**

/1/

SIMATIC NET
S7 CPs for Industrial Ethernet
Configuring and Commissioning
Manual Part A - General Applications
Configuration Manual
Siemens AG
B.2 On programming

/2/

SIMATIC NET
Program blocks for SIMATIC NET S7 CPs
Programming Manual
Siemens AG

/3/

SIMATIC NET
Program blocks for SIMATIC NET S7 CPs
Version history, reference document
Siemens AG

/4/

SIMATIC
Programming with STEP 7
Siemens AG
( Part of the STEP 7 documentation package STEP 7 Basic Knowledge)
( Part of the online documentation in STEP 7)

/5/

SIMATIC
System and Standard Functions for S7-300/400 - Volume 1/2
Reference manual
Siemens AG
( Part of the STEP 7 documentation package STEP 7 Basic Knowledge)
( Part of the online documentation in STEP 7)
On the Internet under the following entry ID:
## Index

**A**
- Access coordination, 41
- AG_CNTEX, 58
- AG_CNTRL, 47

**B**
- Block library, 202

**C**
- CONF_DB, 124
- CONF_DB see configuration data block, 107
- Configuration data block, 107
- Connection ID, 113
  - Possible values, 113
- Connection types
  - Parameter fields for, 113
- CP address, 13, 15

**E**
- E-mail connection
  - Parameter field for, 117

**F**
- FB, 13, 15
  - Condition codes and error messages reported by the FMS partner, 274
  - Condition codes and error messages: locally detected errors,
    - IDENTIFY, 259
    - READ, 261
    - REPORT, 263
    - STATUS, 265
    - WRITE, 268
- FC blocks
  - Block number, 15
  - General information, 15
- FCs for access coordination with FETCH/WRITE, 41
- FMS
  - Condition codes and error messages, 270, 270
  - Module parameters, 256

**G**
- Glossary, 5

**I**
- ISO-on-TCP connection
  - Parameter field for, 116

**J**
- Job buffer, 207
- Job header, 23, 207, 209

**P**
- PE_DS3_Write_ET200S_CP, 14
- PE_Mode_ID, 168, 193
- Ping command, 58, 61
- PNIO_ALARM (FB54) block
  - Formal parameter, 164
- PNIO_RECV (FC12)
  - Formal parameter, 151
- PNIO_RW_REC (FB52)
  - Formal parameter, 160
- PNIO_SEND (FC11) block
  - Formal parameter, 143
- PROFIenergy controller, 167
- PROFIenergy device, 167

**R**
- Replacing a module, 15, 17

FTP connection
- Parameter field for, 118

Functions (FCs)
- AG_LOCK / AG_UNLOCK, 41
- AG_RECV / AG_LRECV, 215
- AG_RECV / AG_LRECV / AG_SRECV, 32
- AG_SEND / AG_LSEND, 209
- AG_SEND / AG_LSEND / AG_SSEND, 25
- Block number, 13
  - for configured connections, 21
- For SIS5 connection, 207
- General information, 13
Index

S
Service_Request_ID, 177
SIMATIC NET glossary, 5
Subfield types, 119
System data
  Parameter field for, 112

T
TCP connection
  Parameter field for, 114

U
UDP connection
  Parameter field for, 115