MOBY I®

ASM 421 Interface Module

Technical Description

Table of Contents

Brief description of the MOBY file handler 1

ASM 421 hardware 2

Programming the ASM 421 module 3

Cold start and restart 4

Checking entry and exit of MDS in SLG transmission window 5

System and transmission times 6

Troubleshooting and error messages 7

Warnings 8

Published in March 2002
Safety Guidelines

This manual contains notices which you should observe to ensure your own personal safety, as well to protect the product and connected equipment. These notices are highlighted in the manual by a warning triangle and are marked as follows to the level of danger.

! **Danger**
indicates that death, severe personal injury or substantial property damage will result if proper precautions are not taken.

! **Warning**
indicates that death, severe personal injury or substantial property damage can result if proper precautions are not taken.

! **Caution**
indicates that minor personal injury or property damage can result if proper precautions are not taken.

**Note**
draws your attention to particularly important information on the product, handling the product, or to a particular part of the documentation.

Qualified Personnel

Only qualified personnel should be allowed to install and work on this equipment. Qualified persons are defined as persons who are authorized to commission, to ground, and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

Correct Usage

Note the following:

! **Warning**
This device and its components may only be used for the applications described in the catalog or technical description, and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens.

This product can only function correctly and safely if it is transported, stored, set up, and installed correctly, and operated and maintained as recommended.

Trademarks

SIMATIC® and MOBY® are registered trademarks of SIEMENS AG.

Copyright © Siemens AG 1995, 2002 All rights reserved

The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Disclaimer of Liability

We have checked the contents of this manual for agreement with the hardware and software described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual are reviewed regularly and any necessary corrections included in subsequent editions. Suggestions for improvement are welcomed.

Siemens AG
Automation & Drives (A&D)
Systems Engineering (A&D SE)
P.O. Box 23 55, D-90713 Fuerth

Siemens Aktiengesellschaft
Order No. 6GT2 097-3AH00-0DA2
Contents

1 Brief description of the MOBY file handler ................................................. 2
2 ASM 421 hardware .................................................................................. 5
   2.1 Technical data ................................................................................. 6
   2.2 Pin assignments ............................................................................... 7
   2.2.1 Pin-outs of X1 backplane connector ........................................... 7
   2.2.2 ASM 421 connector system ......................................................... 8
   2.3 Setting the mode of operation ......................................................... 9
   2.4 Hardware planning .......................................................................... 10
   2.4.1 System configuration ................................................................. 10
   2.4.2 Power supply ............................................................................. 12
   2.4.3 TTY cabling ............................................................................... 13
   2.4.4 RS422 cabling ........................................................................... 15
   2.4.5 V.24 cabling .............................................................................. 16
   2.4.6 STG cabling ............................................................................... 16
   2.4.7 DI/DO cabling ............................................................................ 17
3 Programming the ASM 421 module ......................................................... 18
   3.1 3964R procedure ........................................................................... 18
   3.2 Telegram format ............................................................................. 20
   3.3 Blocking of long data sequences ....................................................... 21
   3.4 File handler commands .................................................................... 22
   3.4.1 FORMAT ................................................................................... 23
   3.4.2 CREATE ................................................................................... 24
   3.4.3 QUEUE-WRITE ......................................................................... 25
   3.4.4 WRITE/UPDATE ....................................................................... 27
   3.4.5 READ ....................................................................................... 29
   3.4.6 QUEUE-READ ........................................................................... 31
   3.4.7 DELETE .................................................................................... 34
   3.4.8 COVER ..................................................................................... 35
   3.4.9 DIR ........................................................................................... 36
   3.4.10 MDS status .............................................................................. 37
   3.4.11 ASM Status ............................................................................. 38
   3.4.12 TRACE .................................................................................... 39
   3.4.13 ATTRIB ................................................................................... 40
   3.4.14 NEXT ....................................................................................... 41
   3.4.15 RESET ..................................................................................... 42
   3.4.16 LOAD/MOVE ........................................................................... 44
   3.5 Telegrams from file handler .............................................................. 46
   3.5.1 Start-up message ......................................................................... 46
   3.5.2 Change in proximity status ......................................................... 46
4 Cold start and restart ............................................................................... 47
5 Checking entry and exit of MDS in SLG transmission window ............... 49
   5.1 MDS recognition by field scanning ............................................... 50
   5.2 Without entry/exit monitoring ......................................................... 51
   5.3 MDS control using field scanning and 1 DI ................................... 52
   5.4 MDS control using 2 DIs .................................................................. 53
   5.5 Without entry/exit monitoring ......................................................... 54
   5.6 Special Operating Mode (Test Operation) ....................................... 55
   5.7 Diagram showing possible MDS states at an SLG ......................... 56
6 System and transmission times ............................................................... 59
7 Troubleshooting and error messages ....................................................... 61
8 Warnings ............................................................................................... 67
1 Brief description of the MOBY file handler

The ASM 421 is a serial interface module for the MOBY I identification system. It can be driven from any computer, PC or PLC using the 3964R procedure. The MOBY I file handler is implemented on the ASM 421.

What is the file handler?

The file handler is a data management system for the MOBY I identification system. It consists of a program that runs on the interface module. The MDS (mobile data carrier) and SLG (read-write unit) components of MOBY I are identical with and without file handler.

For better understanding of the MOBY file handler, it works in much the same way as a floppy disk system.

<table>
<thead>
<tr>
<th>MOBY I File Handler</th>
<th>Floppy Disk System</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MDS:</strong></td>
<td><strong>Floppy Disk:</strong></td>
</tr>
<tr>
<td>Data is stored electronically in the MDS on a memory chip (RAM or EEPROM).</td>
<td>Data is written magnetically on a floppy disk.</td>
</tr>
<tr>
<td><strong>SLG:</strong></td>
<td><strong>Floppy Drive:</strong></td>
</tr>
<tr>
<td>The SLG modulates the data stream arriving from the ASM and demodulates the data stream arriving from the MDS.</td>
<td>The floppy drive converts the data impulses arriving from the PC into a write string for the read/write head (or vice versa)</td>
</tr>
</tbody>
</table>

The positioning of the MDS must lie within the transmission window of the SLG. This amounts to several centimeters.

The MOBY I interface module (ASM 421) provides the user interface. A USART module on the ASM produces the serial data stream for the SLG.

The floppy disk must be positioned very exactly over the read/write head as this generates only very small magnetic fields.

A PC controls the floppy drive and exchanges data with it via a parallel interface. The coding and decoding of the data takes place in a floppy controller in the PC.
Similarities between MOBY file handler and a floppy disk

- The user does not access physical memory addresses
- The user addresses the data via logical names, which consist of up to eight letters
- The conversion of the logical names into physical memory addresses is carried out by software in the interface module:
  File handler ↔ Disk-Operating-System (DOS)
- Contiguous data blocks are called files
- The management of files of differing lengths is carried out by a software procedure (file handler)
- The memory is divided into 3 areas:

  ![Diagram of memory areas]

  - System area
  - Directory
  - User data

  The system area contains general system data:
  - Volume name
  - Format and size of memory
  - Data coding

  All file names and file lengths are stored in the directory. The file handler calculates the physical addresses of the user data from these entries.

  The largest memory area is reserved for user data, which can be stored in any form (ASCII, binary, hexadecimal, decimal, etc.). The user data is managed by the file handler in blocks. On a floppy, these blocks are known as "sectors" or "logical blocks".

Advantages of the file handler for the MOBY I user.

- When specifying a system, it is only necessary to define the names (file names) of the individual data objects that are to be stored on the MDS.
- The project engineer does not need to structure MDS memory at all.
- The list of data objects can be changed or extended as required during detailed planning. The MDS data structure does not need to be taken into account.
- The lengths of the individual data objects do not need to be known.
- The description of each data object is a name of up to eight letters. The choice of name can be directly related to the object (e.g., WORKSNO; SERIALNO; QUALITY; ADDRESS; ORDER; Drillprog; Turnprog; Millprog;...)
- The programmer does not need to work with absolute addresses. Several sources of programming errors are thus excluded (e.g. double addressing, input errors leading to wrong addresses and file lengths). The file handler converts the file names into physical MDS addresses.
- Details of the files stored on the MDS can be requested by the "directory" command.
- Data objects can be accessed simply and rapidly using the STG 4F service and test unit. This is of great advantage to commissioning and maintenance personnel.
- The documentation of the MDS memory structure is very simple and easy to understand, consisting only of a list of file names.
- The user can allocate access rights. Unauthorized deletion and overwriting of user data is thus avoided.
Formats of different MDS memories:

MDS units with various memory sizes are available with MOBY I. Each type of MDS must be formatted before processing with the file handler. During formatting, the MDS is given a data structure, divided into system area, directory and user data. The MDS type is passed to the file handler by the FORMAT command (see section 3.4.1).

The following table shows how the file handler structures the various types of MDS:

<table>
<thead>
<tr>
<th>Type (Hex)</th>
<th>MDS capacity in bytes (gross)</th>
<th>Max. no. of files</th>
<th>User data in bytes (net)</th>
<th>Number of blocks</th>
<th>Block size in bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>62</td>
<td>1</td>
<td>27</td>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>81H</td>
<td>42 *</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>02H</td>
<td>62</td>
<td>2</td>
<td>12</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>03H</td>
<td>128</td>
<td>3</td>
<td>60</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>83H</td>
<td>112 *</td>
<td>3</td>
<td>45</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>04H</td>
<td>2.045</td>
<td>16</td>
<td>1.680</td>
<td>105</td>
<td>16</td>
</tr>
<tr>
<td>84H</td>
<td>1.778 *</td>
<td>16</td>
<td>1.440</td>
<td>90</td>
<td>16</td>
</tr>
<tr>
<td>05H</td>
<td>8,189</td>
<td>32</td>
<td>7,456</td>
<td>233</td>
<td>32</td>
</tr>
<tr>
<td>85H</td>
<td>7,154 *</td>
<td>32</td>
<td>6,464</td>
<td>202</td>
<td>32</td>
</tr>
<tr>
<td>06H</td>
<td>32,765</td>
<td>64</td>
<td>31,488</td>
<td>246</td>
<td>128</td>
</tr>
<tr>
<td>86H</td>
<td>28,658 *</td>
<td>64</td>
<td>27,520</td>
<td>215</td>
<td>128</td>
</tr>
</tbody>
</table>

* With ECC operation
2 ASM 421 hardware

**Pin assignments to SLG**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>reserve</td>
</tr>
<tr>
<td>2</td>
<td>+ transmit</td>
</tr>
<tr>
<td>3</td>
<td>+ receive</td>
</tr>
<tr>
<td>4</td>
<td>reserve</td>
</tr>
<tr>
<td>5</td>
<td>– receive</td>
</tr>
<tr>
<td>6</td>
<td>– transmit</td>
</tr>
<tr>
<td>7</td>
<td>ground (0 V)</td>
</tr>
<tr>
<td>8</td>
<td>+ 24 V</td>
</tr>
<tr>
<td>9</td>
<td>reserve</td>
</tr>
<tr>
<td>housing</td>
<td>cable screen</td>
</tr>
</tbody>
</table>

housing = connector housing
2.1 Technical data

Environmental conditions:
- operating temperature: 0 °C to + 55 °C
- storage temperature: – 40 °C to + 70 °C

Degree of protection to IEC 529: IP 00

Serial interface (to computer/PC):
- transmission rate: 2400 to 19200 Baud
- procedure: 3964R; odd/even parity
- cable lengths:
  - TTY 1000 m (shielded)
  - RS422 1000 m (shielded)
  - V.24 30 m (shielded)

Serial interface (to SLG):
- transmission rate (gross): 19200 Baud
- procedure: Asynchronous: 8 bit data: even parity
  MOBY I: MDS protocol
- cable lengths: SLG dependent; max. 1000 m
  (see cable configurations in manual for configuration, installation and service)

Power supply: 20 to 30 V DC

Current input:
- max. off-load current: 200 mA
  (no SLG; DO's off-load)

Dimensions:
- L x W x H 160 x 100 x 20 (mm)

DI/DO; Select; Request; Error; ANW:
- digital inputs:
  - 3 non-floating
    logical “0”: 0 V to 8 V
    logical “1”: 15 V to 24 V
    (Ri = 10 kOhm)
    delay: < 10 ms

- digital outputs:
  - 5 non-floating
    (internal power supply)
    short-circuit proof
    I_{max} = 200 mA (per DO; or for 2 DO's)
2.2 Pin assignments

2.2.1 Pin-outs of X1 backplane connector

### Interface (type dependent)

<table>
<thead>
<tr>
<th></th>
<th>V.24</th>
<th>RS422</th>
<th>TTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>b4</td>
<td>—</td>
<td>R+</td>
<td>+EM</td>
</tr>
<tr>
<td>b6</td>
<td>RxD</td>
<td>D+</td>
<td>+SE</td>
</tr>
<tr>
<td>b8</td>
<td>—</td>
<td>D–</td>
<td>—*</td>
</tr>
<tr>
<td>b10</td>
<td>DIO</td>
<td>free</td>
<td></td>
</tr>
<tr>
<td>b12</td>
<td>ASM has user data or acknowledgement (Request)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b14</td>
<td>Error code (same as red LED)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b16</td>
<td>SLG receive +</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b20</td>
<td>SLG transmit +</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b22</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>b24</td>
<td>Protective ground (shield)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b26</td>
<td>0 V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Pin b8 must not be connected in TTY mode!
** These data connections to the SLG are available as options on the backplane connector. They are identical to the corresponding connections on the 9-pole submin. D connector.
*** EAKO = entry/exit checking (see chapter 5)
**** The presence of an MDS is indicated with 0 V level on this pin.
2.2.2 ASM 421 connector system

Connector: The connector used with the ASM 520 is a 48-way male connector conforming to DIN 41612 design F, whose soldering pins are bent through 90°. Rows z and b only are connected and soldered directly to the ASM 520.

Socket: Female connectors conforming to DIN 41612 design F must be used as mating components for the ASM 421 connector. In principle, all 48-way “female connectors conforming to DIN 41612 design F” are suitable.

Various types of female connectors are available, e.g.:

- with solder terminations
- with a screw connection
- as a wire wrap version

e.g. SFL 0.5 / F 32 / 2 B:

![Diagram of female connector]

Features:

- 3 rows of connectors
- Number of contacts: 32 (2 rows, each with 16 contacts)
- Contact principle: insulation displacement connector with double-sided contact spring
- Catch hooks and elements to ensure secure contact between male and female connectors
2.3 Setting the mode of operation

Switch bank S1:

- **ON** = "1"
- **OFF** = "0"

1 = Proximity message from ASM; (Z-message)
0 = no proximity message

**Access rights of this ASM station**

- 00 Step “D” (= Delete); all commands allowed
- 01 Step “W” (= Write); Formatting of MDS and deletion of files are not allowed
- 10 Step “R” (= Read only); only read operations on the MDS are allowed
- 11 = reserved for 8-E41

- 0 = Siemens file handler is set up
- 1 = reserved for 8-E41

1 = Master
0 = Slave

1 = odd parity
0 = even parity

**Baud rate:**

- 00 = 9600 Baud
- 01 = 4800 Baud
- 10 = Reserved for 38.4 Kbaud
- 11 = 19200 Baud

**Setting of the 3964R procedure to the user**
2.4 Hardware planning

2.4.1 System configuration

The SLG read/write device can be connected to either the X1 backplane connector or the 9-way subminiature D connector X2.

The SLG read/write device can be connected to either the X1 backplane connector or the 9-way subminiature D connector X2.

Please note:
The Select and Request lines may be omitted if only one ASM 421 is being driven from a serial interface. The external wiring of unnecessary signals can also be omitted.

* Daisy-chain configurations are only possible with the RS422 and TTY interfaces.

Two standard configurations are available
– rack with 4 slots
– rack with 8 slots.
ASM 421 interface modules can only be daisy-chained if they are using an RS422 or TTY interface. This allows up to 16 interface modules to be driven from one serial interface.

The computer/PC must have a digital input (DI) and a digital output (DO) for each ASM 421.

**Mode of operation:**

**Programming the ASM:**
The computer/PC “selects” the module and then sends it the task telegram.

**Command processing:**
The ASM then processes the command. How long this takes depends on the command and can last any amount of time (when there is no MDS in the field).

**Send result to user:**
By setting the “Request” flag, the ASM indicates that it wants to send a telegram. The computer/PC recognizes this and “selects” the module. The selected ASM then sends the result telegram.

![Diagram of timing sequence](attachment:diagram.png)

- **t₁ ≥ 100 ms**: Wait time from select until data transmission.
- **t₂ ≥ 0**: Wait time from data transmission until deselection of ASM 421
- **t₃ Depends on command**: Processing of user command by the ASM 421
- **t₄ ≥ 0**: Time from “request active” until selection of ASM 421
- **t₅ = 0 ... 30 ms**: Time from selection of the module until transmission of the result telegram

**Important:**
- At the result telegram the ASM only interrogates the “select” at the beginning of a transmission. If the “select” is deactivated during transmission of the telegram, the ASM will nevertheless send the complete telegram to the computer.
- This ensures that no garbled telegrams from the ASM are transmitted.
- The select line must be set high for the entire time that an outgoing telegram is being sent.
### 2.4.2 Power supply

The cable between the ASM 421 and the SLG read/write device is a 6-core shielded cable, of which 4 cores are dedicated to the data interface and 2 to the power supply for the SLG read/write device. The maximum length of the data cables can, depending on the physical interface, be anything up to 1000 m. The maximum permitted cable run is generally shorter if the 24 V supply for the SLG read/write device is used by the ASM 421. The maximum cable length also depends on the level of voltage drop. The following table gives an overview of permitted cable lengths:

<table>
<thead>
<tr>
<th>Conductor cross-section [mm²]</th>
<th>Conductor dia. [mm]</th>
<th>Resistance Ω/km *)</th>
<th>SLG 40 / SLG 41 (I=90 mA)</th>
<th>SLG 42 (I=180 mA)</th>
<th>SLG 43 (I=250 mA)</th>
<th>SLG 44 (I=80 mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Max. cable length when U₉ = 24 V</td>
<td>Max. cable length when U₉ = 30 V</td>
<td>Max. cable length when U₉ = 24 V</td>
<td>Max. cable length when U₉ = 30 V</td>
</tr>
<tr>
<td>0.07</td>
<td>0.3</td>
<td>550</td>
<td>120</td>
<td>240</td>
<td>40</td>
<td>100</td>
</tr>
<tr>
<td>0.2</td>
<td>0.5</td>
<td>185</td>
<td>360</td>
<td>720</td>
<td>120</td>
<td>300</td>
</tr>
<tr>
<td>0.5</td>
<td>0.8</td>
<td>70</td>
<td>950</td>
<td>1000</td>
<td>310</td>
<td>790</td>
</tr>
<tr>
<td>0.8</td>
<td>1.0</td>
<td>50</td>
<td>1000</td>
<td>1000</td>
<td>440</td>
<td>1000</td>
</tr>
<tr>
<td>1.5</td>
<td>1.4</td>
<td>24</td>
<td>1000</td>
<td>1000</td>
<td>920</td>
<td>1000</td>
</tr>
</tbody>
</table>

*) The values for resistance are average values and relate to the feed and return conductors. An individual conductor has half the quoted resistance.

If cable runs longer than those specified in the table are used, the 24 V supply for the SLG read/write device must not be used by the ASM 421. Instead, a 24 V supply must be fed directly to the SLG. (See manual for configuration, installation and service; chapter 3.10; cable and connector allocation.)

**Highlighted field:**

The standard shielded LiYCY 6 x 0.25 cable recommended by SIEMENS. This cable is available from SIEMENS under the order number 6GT2090-0A...1).

**24 V supply to ASM 421:**

The maximum length of the power supply cable for the ASM 421 is restricted to 20 m. If a longer cable is used, or several ASMs are supplied from a single cable, the voltage drop on the supply cable to both the ASM 421 and SLG read/write device must be taken into account. If necessary, the cross section of the supply cable should be increased.

1) According to length code
2.4.3 TTY cabling

The ASM module does not contain any power sources for energizing the active part of the TTY interface. Should, however, it become necessary for the ASM to power the interface, the ASM should be wired as follows to make it active:

The resistors R should each have a value of 470 Ohms to produce a current loop of 20 ... 30 mA (P = 0.5 W).
Example of a daisy-chain configuration (see [2.4.1])

*) This line must be looped through from the computer to the last ASM!

Where possible, the computer should handle the active part of the TTY interface, in which case the resistors shown on ASM no. 1 will not be needed.

If this is not possible (as shown in the diagram), the active part of the interface must be simulated in the connector of ASM no. 1 using 4 resistors (R = 470 Ohm).

The numbers in the diagram refer to pins in the 48-way female connector.

**Note:** With this cabling, all the wired ASMs must actually be connected, otherwise the circuit will remain open!
2.4.4 RS422 cabling

*) The cable can be terminated on the receiving side with jumpers z4 – b4 and z8 – z6. This improves the interference immunity of the system particularly when long cables are used (i.e., > 200 m). The jumpers may only be wired on the last ASM 421 when daisy chain operation is used (see chapter 2.4.1).

**) The indicated terminal resistors (R = 200 to 1000 Ohm) must be present on the computer side. These resistors can be wired on the ASM side when short cables are used. This means jumpers z6 – z8 and b4 – z4 are omitted and 2 jumpers are added (i.e., b4 – b6 and z6 – b8).

Example of a daisy-chain configuration (see 2.4.1)

---

Subject to change without notice!
2.4.5 V.24 cabling

V.24 control lines (e.g. DSR, DTR, RTS, CTS) are not supported by the ASM. The acknowledgement of data is handled by the procedure.

2.4.6 STG cabling

The ASM 421 / RS422 can be operated directly with the STG 4F. The complete path from the STG 4F via the ASM 421 to the SLG will thus be tested.
2.4.7 DI/DO cabling

Cable length
max. 100 m
(shielded or unshielded)

Relay (use break-contact in load circuit of relay), horn, lamps, etc.

Cable length
220 V
24 V

Proximity switch

Error code

DI 1

DI 0

z2

z18

b18

z12

b12

b2

Imax = 200 mA
(per DO or total current)
3 Programming the ASM 421 module

3.1 3964R procedure

The 3964R procedure provides secure data transmission across a point-to-point connection. Security is achieved by transmitting the data one block at a time with parity check, block check character (BCC) and acknowledgement of receipt. All characters from 00hex to FFhex can be present in the data block.

Character frame:

- Transmission: asynchronous
- Baud rate: 2400, 4800, 9600, 19200 Baud
- Data bits: 8
- Parity: odd/even (switch selectable)
- Stop bit: 1

<table>
<thead>
<tr>
<th>Control characters in the 3964R procedure:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>STX</td>
</tr>
<tr>
<td>DLE ETX</td>
</tr>
<tr>
<td>DLE</td>
</tr>
<tr>
<td>NAK</td>
</tr>
<tr>
<td>DLE DLE</td>
</tr>
</tbody>
</table>
Block Transmission Sequence:

Sender

<table>
<thead>
<tr>
<th>1st user byte</th>
<th>2nd user byte</th>
<th>last user byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>STX</td>
<td>DLE</td>
<td>Data block</td>
</tr>
</tbody>
</table>

Receiver

<table>
<thead>
<tr>
<th>DLE</th>
<th>BCC</th>
<th>ETX</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLE</td>
<td>BCC</td>
<td>DLE</td>
</tr>
</tbody>
</table>

$t_z = \text{Watchdog timer on ASM}\rightarrow 300 \text{ ms}$

Data security

To protect data against transmission errors, a parity bit is attached to each character transmitted. It is supplemented by an odd or even quantity of “ones”. In addition, the sum of the respective bits of each character in a block is supplemented by a further bit to even parity (EXOR operation on all characters). The resulting block check character (BCC) is itself secured by the character parity principle and is transmitted at the end of the block. All characters in the block are saved except the start control character STX.

Repeat counter

If no acknowledgement or a negative acknowledgement is received, the telegram will be sent again. The 3964R has a repeat counter. If a block cannot be correctly transmitted after a number of repeats, the transmission will be terminated. The telegram is lost or the user receives an error message.

Initiation conflict

Since both stations have equal rights, an initiation conflict can arise if both partners try to transmit at the same time. To avoid this, one station is given a higher priority (master) than the other (slave). This can be set using switch 4 (see section 2.3).

Watchdog timers

Two watchdog timers are used to monitor the data transfer:

- **Acknowledgement watchdog timer $t_Q = 300 \text{ ms}$**
  The acknowledgement watchdog timer is started after transmission of the control character STX, or DLE ETX BCC.
  If a positive acknowledgement fails to arrive within $t_Q$, the whole data block is sent again. If the transmission has still not been performed correctly after three attempts, an error message is sent to the user or the telegram is rejected by the ASM and an error indicated by the red LED.

- **Character watchdog timer $t_z = 300 \text{ ms}$**
  The receiving ASM 421 monitors the arrival of the individual characters with time constant $t_z$.
  If a character fails to arrive within the watchdog period, receiving is aborted and an error indicated by the red LED.
3.2 Telegram format

The user communicates with the file handler via telegrams. Telegrams from the user are checked, interpreted and processed by the file handler. The user then receives an acknowledgement telegram containing user data and status information.

The following schematic shows the general telegram format:

![Telegram Format Schematic]

**Command code:**

- 'I' : FORMAT
- 'B' : CREATE
- 'Q' : QUEUE-WRITE
- 'W' : WRITE
- 'U' : UPDATE
- 'R' : READ
- 'D' : DELETE
- 'Y' : ATTRIB
- 'C' : COVER
- 'G' : DIR
- 'N' : NEXT
- 'X' : RESET
- 'F' : MDS-STATUS
- 'S' : ASM-STATUS
- 'L' : LOAD
- 'M' : MOVE
- 'T' : TRACE
- 'H' : Start-up message from ASM
- 'Z' : Special telegram (ANW message)

**Reserved for sequence number:**

The file handler transmits the parameter LFN from the command in the acknowledgement. It is not evaluated by the file handler.

**Reserved for receiver recognition:**

Presently always 00

**Reserved for sender identifier:**

(MOBY I: code 'F' is reserved for STG)

* The file handler interchanges sender and receiver identifiers.
3.3 Blocking of long data sequences

Since it is not possible to work with telegrams of indefinite length, the telegram is formed into blocks, i.e. telegrams are transmitted in segments.

Blocking can be used with READ, WRITE, UPDATE, QUEUE-WRITE, DIR and TRACE commands. Blocking is controlled by the parameters ADB (no. of data blocks) and DBN (current data block no.) in the telegram heading.

The file description immediately follows the DBL parameter in the first command telegram to the file handler (DBN = 1). It contains the file name, start address and length of the file. The file description is omitted from subsequent telegram segments (DBN > 1).

Schematic sequence (example WRITE):

```
0003 User data
```

1st command telegram from user to ASM

Processing of 1st telegram segment by the file handler on the ASM 421

```
0001
```

1st acknowledgement telegram from ASM to user

The user processes the acknowledgement and issues the 2nd command segment

```
0003 User data
```

2nd command telegram from user to ASM

Processing of 2nd telegram segment by the file handler on the ASM 421

```
0002
```

2nd acknowledgement telegram from ASM to user

The user processes the acknowledgement and issues the 3rd command segment

```
0003 User data
```

3rd or last command telegram from user to ASM

Processing of last telegram segment and terminating the command by the file handler on the ASM 421

```
0003
```

3rd or last acknowledgement tel. from ASM to user

The command has been processed.
The maximum block length (maximum length of a telegram segment) can be set using the RESET command. The default value for the block length is 128. The maximum block length that can be set is 255.

The blocking sequence can be aborted using the RESET command. The blocking sequence will be ended. No further acknowledgement of the aborted command will be issued.

All other commands within a blocking sequence will be rejected by the ASM 421 with an error message. The present status of the blocking sequence remains unchanged.

3.4 File handler commands

The following table lists all file handler commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>CC</th>
<th>Purpose</th>
<th>Prio*</th>
<th>Telegram format (to ASM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMAT</td>
<td>I</td>
<td>Format and initialize MDS</td>
<td>D</td>
<td>00 00 0000 T Y 0001 0001 OC</td>
</tr>
<tr>
<td>CREATE</td>
<td>B</td>
<td>Create file on MDS</td>
<td>WD</td>
<td>00 00 0000 B Y 0001 0001 OC</td>
</tr>
<tr>
<td>QUEUE-WRITE</td>
<td>Q</td>
<td>Set up complete data carrier</td>
<td>D</td>
<td>00 00 0000 Q T AD8 0001 DBL Option Res Res Leng. Data</td>
</tr>
<tr>
<td>WRITE</td>
<td>W</td>
<td>Write part or complete file</td>
<td>WD</td>
<td>00 00 0000 W T AD8 0001 DBL F.name Adr. Leng. Data</td>
</tr>
<tr>
<td>UPDATE</td>
<td>U</td>
<td>Update complete file</td>
<td>WD</td>
<td>00 00 0000 U T AD8 0001 DBL F.name Adr. Leng. Data</td>
</tr>
<tr>
<td>READ</td>
<td>R</td>
<td>Read data from file</td>
<td>RWD</td>
<td>00 00 0000 R Y 0001 0001 0E F.name Adr. Leng.</td>
</tr>
<tr>
<td>QUEUE-READ</td>
<td>E</td>
<td>Read complete data medium</td>
<td>RWD</td>
<td>00 00 0000 E Y 0001 0001 0B Option Res Leng. Data</td>
</tr>
<tr>
<td>DELETE</td>
<td>D</td>
<td>Delete file</td>
<td>D</td>
<td>00 00 0000 D Y 0001 0001 08 F.name</td>
</tr>
<tr>
<td>ATTRIB</td>
<td>Y</td>
<td>Set file attribute</td>
<td>WD</td>
<td>00 00 0000 Y Y 0001 0001 09 F.name Attr.</td>
</tr>
<tr>
<td>COVER</td>
<td>C</td>
<td>Protect MDS structure</td>
<td>D</td>
<td>00 00 0000 C Y 0001 0001 09 F.name Attr.</td>
</tr>
<tr>
<td>DIR</td>
<td>G</td>
<td>Read MDS directory</td>
<td>RWD</td>
<td>00 00 0000 G Y 0001 0001 00</td>
</tr>
<tr>
<td>ASM-STATUS</td>
<td>S</td>
<td>Read ASM status</td>
<td>RWD</td>
<td>00 00 0000 S Y 0001 0001 00</td>
</tr>
<tr>
<td>MDS-STATUS</td>
<td>F</td>
<td>Read MDS status</td>
<td>RWD</td>
<td>00 00 0000 F Y 0001 0001 00</td>
</tr>
<tr>
<td>RESET</td>
<td>X</td>
<td>Reset command to ASM</td>
<td>RWD</td>
<td>00 00 0000 X Y 0001 0001 07 Mode AKC ECC No. Prior. Reg.</td>
</tr>
<tr>
<td>NEXT</td>
<td>N</td>
<td>Process the next MDS</td>
<td>RWD</td>
<td>00 00 0000 N Y 0001 0001 00</td>
</tr>
<tr>
<td>TRACE</td>
<td>T</td>
<td>MDS absolute read</td>
<td>RWD</td>
<td>00 00 0000 T Y 0001 0001 06 Startadr Leng.</td>
</tr>
<tr>
<td>LOAD</td>
<td>L</td>
<td>Transmit directory to ASM</td>
<td>RWD</td>
<td>00 00 0000 L T AD8 0001 DBL Leng. Data</td>
</tr>
<tr>
<td>MOVE</td>
<td>M</td>
<td>Retrieve directory from ASM</td>
<td>RWD</td>
<td>00 00 0000 M T 0001 0001 00</td>
</tr>
<tr>
<td>Startup</td>
<td>H</td>
<td>Reported by ASM when power returns</td>
<td>RWD</td>
<td>00 00 0000 V 00 0001 0001 00 **</td>
</tr>
<tr>
<td>Presence</td>
<td>Z</td>
<td>Proximity message from ASM</td>
<td>RWD</td>
<td>00 00 0000 Z 00 0001 0001 05 00 00 00 00 ANW **</td>
</tr>
</tbody>
</table>

* The priority and access rights of the interface module are switch selectable (see section 2.3).
** This message is from ASM only. Switch 8 must be on as described in section 2.3.
3.4.1 FORMAT

The FORMAT command erases the complete MDS. The directory, file allocation table (FAT) and system area on the MDS will be re-created. Some counter contents in the system area will be retained. The current counter contents can be interrogated by the MDS STATUS command.

Telegram to the file handler:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
<th>11/18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>CI</td>
<td>ADB</td>
<td>DBN</td>
<td>DBL</td>
<td>Volume</td>
<td>MDS type</td>
<td>OPTIONS</td>
<td>INIT mode</td>
<td>INIT value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'I'</td>
<td>'I'</td>
<td>0001</td>
<td>0001</td>
<td>OCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The complete MDS will be erased and written with this data if the parameter 'INIT mode' has the value 'L'.

'K' = Not initialized; only DIR and FAT will be re-created on the MDS. The old user data remains physically in memory.

'L' = Initialized; the whole MDS will be written with the data in 'INIT value'.

01H = 62 bytes without ECC
81H = 42 bytes with ECC
02H = 62 bytes without ECC
03H = 128 bytes without ECC
83H = 112 bytes with ECC
04H = 2045 bytes without ECC
05H = 8189 bytes without ECC
85H = 7154 bytes with ECC
06H = 32765 bytes without ECC
86H = 28658 bytes with ECC

MDS name; 8 ASCII characters (20H-7EH)

Acknowledgement:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Status</td>
<td>ADB</td>
<td>DBN</td>
<td>DBL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>'I'</td>
<td>00</td>
<td>0001</td>
<td>0001</td>
<td>00</td>
</tr>
</tbody>
</table>

Important: A new MDS must be formatted before being used in the system.
3.4.2 CREATE

A new file is established on the MDS by the CREATE command. An entry will be made in the MDS directory for each CREATE command. DIR and FAT in the MDS will be updated.

Telegram to the file handler:

```
0 1 2/3 4 5 6/7 8/9 10 11/18 19 20/22
```

<table>
<thead>
<tr>
<th>CC</th>
<th>CI</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'B'</td>
<td>'T'</td>
</tr>
</tbody>
</table>

F: name Attribute Data length to be res.

The file handler reserves the file length given here. If 000000H is given as the data length, the file handler reserves a block as standard procedure.

**Note:** Specification of a data length is not mandatory. If the data written later exceeds the data length given here, the file handler will automatically reserve more memory for the file. MDS memory is managed dynamically by the file handler.

File name; 8 ASCII characters (20H - 7EH)

Acknowledgement:

```
0 1 2/3 4 5 6/7 8/9 10
```

<table>
<thead>
<tr>
<th>CC</th>
<th>Status</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>'B'</td>
<td>00</td>
</tr>
</tbody>
</table>

| 0001 0010 |
3.4.3 QUEUE-WRITE

This command combines several WRITE commands into one command. This gives you the following advantages:

1. Writing of several files is faster since writing is not dependent on the serial transmission to the ASM.
2. The files are provided with a fixed length.
3. Filling in any pattern is possible.
4. Attributes can be assigned.

This function cannot be executed unless the addressed MDS has been formatted correctly. In addition, this command is aborted with error message C006 if the addressed MDS leaves the transmission window prematurely (handling same as for the FORMAT command).

Telegram to the file handler:

```
<table>
<thead>
<tr>
<th>CC</th>
<th>CI</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
<th>Option</th>
<th>xxxxxxx</th>
<th>xxxxxxx</th>
<th>Length of the file entries</th>
<th>File entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>Q</td>
<td>T</td>
<td>ADB</td>
<td>0001</td>
<td>DBL</td>
<td>Disregard</td>
<td></td>
</tr>
</tbody>
</table>
```

Option: The following applies if the data carrier already contains an existing file structure:

- Option = 0000H: The file structure is overwritten or deleted with the QUEUE-WRITE command.
- Option = 0001H: The file structure is added to the existing file structure with the QUEUE-WRITE command. A check is made to determine whether the file to be set up already exists.

File name: File name

Attribute: Attribute for this file. The attribute is not set until the data have been written.

Reserved: 1 byte is reserved.

Skip: Skip specifies how many bytes are between the previous file entry and the next file entry. Example: skip = 00hex (i.e., the file name of the next file entry immediately follows the data of the previous file entry).

Length: Length in bytes of the subsequent data for the specified file.

Data: The data to be written. These data can also be a filling pattern. The amount of data in bytes must correspond to the "length" parameter.

Note:

1. The start address of the QUEUE-WRITE command always starts at the address 000000 hex in the respective file.
2. One block is always reserved by the file handler if the length is 0 hex. Since the file entry does not contain data when the length = 0, the QUEUE-WRITE command corresponds to a multiple execution of the CREATE command in this case.
Acknowledgement:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC Status</td>
<td>ADB</td>
<td>DBN</td>
<td>DBL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>Q</td>
<td>00</td>
<td>ADB</td>
<td>0001</td>
<td>00</td>
</tr>
</tbody>
</table>

Handling of Errors:

If the file handler detects a syntax error in the structure of the transmitted file entry, the QUEUE-WRITE command is aborted with an appropriate error message. In addition, the QUEUE-WRITE command is aborted with the option = 0001 hex if the file handler determines that the file to be set up already exists.

Error Identification for Parameter Error in File Entry:

```
K xx xx
```

“xxx” stands for the number of the file entry in which an error was found. Counting of the file entries begins with 1.

Example: Error message “K002” → Error in the second file entry.
3.4.4 WRITE/UPDATE

Telegram to the file handler:

<table>
<thead>
<tr>
<th>CC</th>
<th>CI</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
<th>File name</th>
<th>Start addr.</th>
<th>Length</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'W'</td>
<td>'U'</td>
<td>ADB</td>
<td>0001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

User data in any format

Offset address within the file:

- 000000 → Normal write command for the complete file.
- xxxxxx → Only part of the existing file is to be overwritten. The start address must remain within the bounds of the existing file.
- FFFFFFF → The automatic append function is activated by the start address –1. New data will be appended to the existing file.

File name; 8 ASCII characters (20H-7EH)
The file must already be entered in the directory.

Length of the data following in the telegram

Number of command segments for the write command. The user must calculate ADB correctly.

'W' → WRITE: All or only part of a file can be written with the write command. If only part of the file is written, old file data will remain at the beginning or end of the file.

'U' → UPDATE: The UPDATE command always writes a complete file. The start address must always be given a value of 000000. If the new file data is shorter than the old data, the new file has the length of the new data. The memory already reserved for this file nevertheless remains.

Acknowledgement:

<table>
<thead>
<tr>
<th>CC</th>
<th>Status</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>0000</td>
<td>'W'</td>
<td>00</td>
<td>0001</td>
</tr>
</tbody>
</table>
Example:
The file ‘EXAMPLE’ is to be written from position 0 with a length of 500 bytes. Assumption: the file already exists, there is enough free memory available on the MDS and the write operation will be performed without error. 514 bytes of user data are to be transmitted (500 + 14 for name, start address, length).
The block length specified as 255 bytes results in 3 telegram segments with user data lengths of 230, 244 and 26 bytes.
3.4.5 READ

Telegram to the file handler:

0 1 2/3 4 5 6/7 8/9 10 11/18 19/21 22/24

<table>
<thead>
<tr>
<th>CC</th>
<th>CI</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
<th>File name</th>
<th>Start addr</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'R'</td>
<td>'I'</td>
<td>0001</td>
<td>0E</td>
<td></td>
</tr>
</tbody>
</table>

Length of file data to be read (in bytes)

000000 → The presence of the file in the directory is checked. No data are transmitted to the user. If the file is not present, an error message results.

xxxxxx → The amount of data specified here is read from the MDS. If the MDS does not contain as much data as specified here, the actual amount of data is transmitted to the user in the DBL and a WARNING is set in the status byte.

FFFFFF → If a length of –1 is passed, the file handler reads the complete file. (The start address must be given a value of 000000). The actual data length will be passed to the user in the DBL parameter. If a start address not equal to 000000 is entered, the file handler reads the file from the specified address to the end of file.

Offset address within the file (hex).
A data file always begins at address 000000. If only a part of the file is read in order to give rapid access to important MDS data, the start address can be set here.

Important: The start address must lie within the bounds of the existing file.

File name; 8 ASCII characters (20H-7EH)

* Important: A value of 0001H must always be entered for the parameters ADB and DBN in the first telegram of a read command. The file handler calculates the actual number of ADBs and sends this to the user in the first acknowledgement telegram (see example on the next page).

Acknowledgement:

0 1 2/3 4 5 6/7 8/9 10 11/255

<table>
<thead>
<tr>
<th>CC</th>
<th>Status</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>'R'</td>
<td>00</td>
<td>xxxx</td>
</tr>
</tbody>
</table>

The actual number of telegram segments for this read operation is passed to the user here.
Example:

The file ‘EXAMPLE’ is to be read from the MDS.
Assumptions: the file exists and the exact length is not known.

1st telegram segment to file handler:

<table>
<thead>
<tr>
<th>CC</th>
<th>CI</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
<th>File name</th>
<th>Start addr.</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>R’</td>
<td>0</td>
<td>'EXAMPLE'</td>
<td>000000</td>
<td>FFFFFFFH</td>
</tr>
</tbody>
</table>

Acknowledgement:

<table>
<thead>
<tr>
<th>CC Status</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
<th>Data (244 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>R’</td>
<td>00</td>
</tr>
</tbody>
</table>

2nd telegram segment to file handler:

<table>
<thead>
<tr>
<th>CC</th>
<th>CI</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>R’</td>
<td>0</td>
</tr>
</tbody>
</table>

Acknowledgement:

<table>
<thead>
<tr>
<th>CC Status</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
<th>Data (244 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>R’</td>
<td>00</td>
</tr>
</tbody>
</table>

3rd telegram segment to file handler:

<table>
<thead>
<tr>
<th>CC</th>
<th>CI</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>R’</td>
<td>0</td>
</tr>
</tbody>
</table>

Acknowledgement:

<table>
<thead>
<tr>
<th>CC Status</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
<th>Data (12 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>R’</td>
<td>00</td>
</tr>
</tbody>
</table>

The total file length can be calculated by the user from the sum of all DBLs:
F4H + F4H + 0CH = 1F4H → decimal: 500 bytes
3.4.6 QUEUE-READ

The QUEUE-READ command is used to read several files (max. of 15) with one command from the MDS. The complete MDS with all files can also be read as an option.

Telegram to the filehandler:

```
0 1 2/3 4 5 6/7 8/9 10 11/12 13/21 22/24 25..254
KK 0000 'E' 'I' 0001 0001 DBL Option in reserve Length Data
```

The user data contain the files to be read (max. of 16 file entries).

Length of the subsequent data starting at byte 25 (in hex format)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>No option. The data contain the files to be read.</td>
</tr>
<tr>
<td>0001</td>
<td>(Res. for QUEUE-WRITE)</td>
</tr>
<tr>
<td>0002</td>
<td>All files of the MDS are read. No data starting at byte 25.</td>
</tr>
<tr>
<td>0004</td>
<td>If the actual file length is longer than the file length reserved in the data, no error message is generated.</td>
</tr>
<tr>
<td>0008</td>
<td>The user data contain the files that are not to be read. This means that all files of the data memory are read except the files specified in the command. The &quot;length&quot; parameter in the file entry can be disregarded for this option.</td>
</tr>
<tr>
<td>0010</td>
<td>The read file data are organized by word and are directly appended to each other. This means that a skip byte with the value 20hex is added to odd length files. The &quot;skip&quot; parameter has the value 0001 in the acknowledgment. With this option &quot;length&quot; must always be set to FFFFFFF in the file entries.</td>
</tr>
</tbody>
</table>

The options are bit-coded. Several options can be set at the same time.

Example: 000C, 0012, 0018, and so on.

Number of subcommands is always 0001 with the command. The filehandler supplies the correct number (ADB) in the first acknowledgment telegram.

File name Res. = 00 Length Reserved Other file entries

One file entry = 14 bytes

8 bytes 1 byte 3 bytes 2 bytes

The file name must always be specified.

FFFFFFFF Read the complete file.

xxxxxx Reserved data length in the acknowledgment telegram.

The file data in the acknowledgment are always xxxxxx bytes (xxxxx_{max} = 4096). The result is shown below.

File length < xxxxxx:

The acknowledgment telegram is filled with 20hex up to the parameterized length xxxxxx. The number of filled bytes is supplied by the filehandler in the data word "skip."

File length > xxxxxx:

The file data are only supplied in the acknowledgment telegram up to a length of xxxxxx. The remaining data are cut off. When option 0004 is set there is no error message. Otherwise the QUEUE-READ is terminated at this point with error F006.
Acknowledgment:

The total length of all data is calculated from the sum of all DBLs in the acknowledgment telegrams.

The real number of subtelegrams for the QUEUE-READ operation is supplied to the user here.

The file name is identical to the file name in the command (with option 0008 the unparameterized file names appear here).

Optional: The skip bytes are only inserted when skip has a value other than 0000. The contents of the skip byte are always “blank” (20hex).

Number of data bytes after the data up to the beginning of the next file name. Skip is always valid. With some options skip always has the value 0000.

Length of the file data for this file. The data may cover several subtelegrams.

Remarks:
The file entries with the file data can be placed anywhere in the data of the acknowledgment telegram. For example, part of the file name may be located in the first acknowledgment telegram. The rest of the file name is then located in the next acknowledgment telegram.
Example:

Five files are to be read from the MDS.

<table>
<thead>
<tr>
<th>1st subtelegram on filehandler:</th>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
<th>11/12</th>
<th>13/21</th>
<th>22/24</th>
<th>25 to 94</th>
</tr>
</thead>
<tbody>
<tr>
<td>KK KI ADB DBN DBL Option In reserve Length Data</td>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'E'</td>
<td>'T'</td>
<td>0001</td>
<td>0001</td>
<td>54H</td>
<td>0000</td>
<td>000000</td>
<td>000046H</td>
<td>File entries</td>
</tr>
</tbody>
</table>

\[= \text{header} + 5 \text{ file entries} = 14 + 5 \times 14 \]

Acknowedgment: 0 1 2/3 4 5 6/7 8/9 10 11 to 254

| KK Status ADB DBN DBL Data (244 bytes) | xx | 00 | 0000 | 'E' | 00 | 0003 | 0001 | F4H |

<table>
<thead>
<tr>
<th>2nd subtelegram on filehandler:</th>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>KK KI ADB DBN DBL</td>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'E'</td>
<td>'T'</td>
<td>0003</td>
<td>0002</td>
<td>00</td>
</tr>
</tbody>
</table>

| KK Status ADB DBN DBL Data (244 bytes) | xx | 00 | 0000 | 'E' | 00 | 0003 | 0002 | F4H | Con't: File entries + data |

<table>
<thead>
<tr>
<th>3rd subtelegram on filehandler:</th>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>KK KI ADB DBN DBL</td>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'E'</td>
<td>'T'</td>
<td>0003</td>
<td>0003</td>
<td>00</td>
</tr>
</tbody>
</table>

| KK Status ADB DBN DBL Data (12 bytes) | xx | 00 | 0000 | 'E' | 00 | 0003 | 0003 | 0CH | Con't: File entries + data |

The total length of the data sent to the user can be calculated from the sum of all DBLs. F4H + F4H + 0CH = 1F4H \rightarrow \text{Decimal: 500 bytes}

The user data must be analyzed to obtain the data length of the individual files.
3.4.7 DELETE

The file is removed from the MDS directory. The reserved memory area is released, and becomes available for other MDS operations. DELETE does not erase old user data.

A delete protected file cannot be deleted: it must have its delete protection attribute removed using the ATTRIB command before it can be deleted.

Telegram to the file handler:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
<th>11/18</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>CI</td>
<td>ADB</td>
<td>DBN</td>
<td>DBL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'D'</td>
<td>'I'</td>
<td>0001</td>
<td>0001</td>
<td>08H</td>
<td>F.name</td>
</tr>
</tbody>
</table>

Name of file to be deleted; 8 ASCII characters (20H-7EH)

Acknowledgement:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Status</td>
<td>ADB</td>
<td>DBN</td>
<td>DBL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>'D'</td>
<td>00</td>
<td>0001</td>
<td>0001</td>
<td>00</td>
</tr>
</tbody>
</table>
### 3.4.8 COVER

1. The file structure of an MDS can be protected against unauthorized access. This means that commands which cause changes in a checksum or create additional DIR or FAT entries can only be issued by authorized users (superusers).

An unauthorized access includes the following:
- Files are deleted or set up.
- File lengths are shortened or lengthened.
- Data carriers are formatted although they have been formatted already.
- Attributes are deleted or new ones are assigned.

After generating the valid file structure on the MDS, an authorized user (superuser) must protect this with the COVER command. This prevents other users from destroying checksums by accessing the data carrier. An illegal command is rejected with error message D022.

2. If the superuser wants to format the MDS or delete/set up files, he/she must first “open” the data carrier again with the COVER command.

3. A COVER command for an unformatted MDS is rejected with error D015.

4. Specification of a volume name (i.e., MDS name) with the COVER command ensures that a data carrier must be unmistakably identified by the superuser. The COVER command is rejected with error D023 when the two MDS names (i.e., data carrier name and volume name in the COVER command) are not identical. Specification of a user identifier (byte 19 in the command telegram) greater than 01 hex is rejected with error message D005.

**Telegram to the file handler:**

```
0 1 2/3 4 5 6/7 8/9 10 11/18 19
CC CI ADB DBN DBL
xx 00 0000 'C' 'I' 0001 0001 09 Volume User
```

01H: The MDS is to be protected against unauthorized access.
00H: Changes to the MDS file structure are permitted again.

**Acknowledgement:**

```
0 1 2/3 4 5 6/7 8/9 10
CC Status ADB DBN DBL
00 xx 0000 'C' 00 ADB 0001 00
```
3.4.9 DIR

Telegram to the file handler:

0 1 2/3 4 5 6/7 8/9 10

xx 00 0000 'G' 'I' 0001 0001 00

Acknowledgement:

0 1 2/3 4 5 6/7 8/9 10 11/18 19/20 21/24 25/32 33/35 36

00 xx 0000 'G' 00 0001 0001 xx 'MDs NAME' xxxx 00000015 ' FILE1' 00000A 00 Further file entries

MDS related data

A data record with the form as shown is transmitted to the user

Attribute for this file
(no attribute in this case)
00 = no attribute
01 = read only
02 = write once; overwriting not possible
04 = Fixed length. The length cannot be changed with write commands.
05 = Same as 04. The file may only be read.

Length of file in bytes
(10 bytes in this case)

File name; 8 ASCII characters

Free memory space in MDS in bytes
(4 bytes, hex)

Check sum calculated by file handler via DIR and FAT. The check sum is placed in the system area on the MDS.

Name of MDS; 8 ASCII characters

If the MDS directory contains a large number of files, the directory data will be transmitted as blocks (ADB > 1 in the first acknowledgement). The user must request the additional blocks.
3.4.10 MDS STATUS

Telegram to the file handler

<table>
<thead>
<tr>
<th>CC</th>
<th>CI</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>F</td>
<td>'I'</td>
</tr>
</tbody>
</table>

Acknowledgement:

<table>
<thead>
<tr>
<th>CC</th>
<th>Status</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>F</td>
<td>'I'</td>
</tr>
</tbody>
</table>

- **ECC corrections:**
  - Counts number of ECC corrections performed.
  - Only valid when working with ECC.
- **No. of times processed:**
  - This counter has a width of 24 bits. It is incremented by 1 at each SLG station before being processed for the first time.

**Indication (binary):**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Support battery:**
  - 0 = OK
  - 1 = poor
  - This bit is always set for MDS types with EEPROM.
- **Aux. battery:**
  - Only for MDS 507
  - 0 = OK
  - 1 = poor

**Free DIR entries:**
- Current count of free directory entries. This number of further files can be opened.
- (16 bits/hex).

**Current free capacity:**
- Free or not yet reserved memory capacity on the MDS.
- (24 bits/hex).

**MDS capacity:**
- Max. size of the user data area in the MDS in bytes.
- (24 bits/hex).

**MDS type:**
- The MDS was formatted with this MDS type by the FORMAT command (see section 3.4.1).
- (1 byte/hex).

**Volume:**
- Name of MDS; 8 ASCII characters (20H-7EH); the name was assigned during FORMAT.
3.4.11 ASM STATUS

The ASM STATUS command provides information from the file handler. The command can be used to look for errors. It shows the telegram traffic performed by the ASM 421.

Telegram to the file handler:

```
                   0  1  2/3  4  5  6/7  8/9  10
            CC  CI  ADB  DBN  DBL
xx 00 0000  'S'  'I'  0001  0001  00
```

Acknowledgement:

```
                   0  1  2/3  4  5  6/7  8/9 10  11/18  19/44  45/70  71
            CC  Status  ADB  DBN  DBL
00 xx 0000  'S'  00 0001 0001 3DH Vers. no. Last instruction Last ack. Connection status
```

Status information regarding state of SLG hardware (24 V supply present; fuse OK; cable to SLG OK; SLG OK)

- 00H → Connection to SLG is OK
- 01H → Fault in connection to SLG

26 bytes: byte 0 to byte 25 of the acknowledgement of the last command *

26 bytes: byte 0 to byte 25 of the last command to the ASM *

Version ident. of the file handler; 8 bytes ASCII

* Only the data are valid that pertain to the telegram header.
The data in the telegram can have any content.
3.4.12 TRACE

The TRACE command provides a physical dump of the MDS memory. It can be very useful for test purposes. The selected type of entry / exit checking is irrelevant as far as this command is concerned. It is always executed immediately. If a data block that exceeds the maximum telegram length is to be read, the data will be transmitted in blocks. Refer to the example in section 3.4.5 to see how the blocking function works.

Telegram to the file handler:

```
CC CI ADB DBN DBL Start addr. Length
xx 00 0000 'T' 'I' 0001 0001 06 xxxxxxH xxxxxxH
```

- Length of the data to be read (3 bytes hex)
- Physical start address in the MDS of the data to be read (3 bytes hex)

Acknowledgement:

```
CC Status ADB DBN DBL Data
00 xx 0000 'T' 00 0001 0001 xx Data
```

- If data length > telegram length: the file handler calculates ADB and sends the value to the user in the first acknowledgement telegram

The TRACE command is the only file handler command that does not return an error indicator in the acknowledgement when processing an MDS that is not formatted (but is initialized). In each case the acknowledgement contains the physical MDS addresses. The ECC mode should be set correctly with the RESET command before using the TRACE command.
3.4.13 ATTRIB

The ATTRIB command enables access rights to be assigned to individual files. Together with the access rights that can be set for each ASM with the DIL switches, each user can individually set the desired access rights. The access rights from the DIL switch and ATTRIB command should be ‘ORed’ together, i.e. both access rights must be correct in order to carry out the command.

Telegram to the file handler:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
<th>11/18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'Y'</td>
<td>'I'</td>
<td>0001</td>
<td>0001</td>
<td>09</td>
<td>F.name</td>
<td>Attribute</td>
</tr>
</tbody>
</table>

Assigning or changing access rights for this file
00H: no attribute for the data entered or delete existing attribute
01H: attribute “read only”
   The file can only be read (no DELETE or WRITE or UPDATE permissible).
02H: attribute “write once”
   The file can be written once and then only read
   (DELETE or WRITE or UPDATE not permitted).
04H: attribute “fixed length”
   The length of the file cannot be changed by write
   commands such as UPDATE, WRITE or APPEND
   (i.e., the checksum in the system area is not changed
   by a write access to this file.
05H: Same as 01H

When other access rights are specified in the ATTRIB command, this is rejected with error message D005.

File name: 8 ASCII characters (20H-7EH)

Acknowledgement:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>'Y'</td>
<td>00</td>
<td>0001</td>
<td>0001</td>
<td>00</td>
</tr>
</tbody>
</table>

Example:

DIL Switch: DELETE (all commands allowed)
Attribute of file “EXAMPLE”: 02H (write once)

The file “EXAMPLE” can be read and can be written once. Deletion of the file is not possible. However, the MDS can be completely erased with the FORMAT command.
3.4.14 NEXT

The NEXT command terminates the processing of an MDS. The command after the NEXT command will only be processed if a new MDS enters the transmission window of the SLG.

Telegram to the file handler:

```
    0 1  2/3  4  5  6/7  8/9  10
   xx 00 0000 'N' 'I' 0001 0001 00
```

Acknowledgement:

```
    0 1  2/3  4  5  6/7  8/9  10
   00 xx 0000 'N' 00 0001 0001 00
```

It is essential to program the NEXT command for the following types of entry / exit checking (EAKO):

- MDS recognition by field scanning: EAKO = 0 (see section 5.1)
- MDS control via 1 DI: EAKO = 2 and field scanning (see section 5.3)
- MDS control via 2 DI: EAKO = 3 (see section 5.4)

The NEXT command does not need to be programmed for the operating modes “timeout” and “without entry / exit checking”. Should it nevertheless be programmed, it will be processed in all its aspects by the file handler.

Further information about this command can be found in section 5.
3.4.15 RESET

Telegram to the file handler:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14/15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'X'</td>
<td>'t'</td>
<td>0001</td>
<td>0001</td>
<td>07</td>
<td>RESET mode</td>
<td>EAKO</td>
<td>ECC</td>
<td>SLG No.</td>
<td>Prior.</td>
<td>Block length</td>
</tr>
</tbody>
</table>

Or

| xx | 00 | 0000 | 'X' | 't' | 0001 | 0001 | 07 | RESET mode | EAKO | ECC | SLG No. | Prior. | Block length |

7ABTAST

Max. telegram length; the file handler will not send an acknowledgement telegram that exceeds the length given here.

Range of values:
64 ... 254
0 = Sets the default value of 128 bytes
If the data length exceeds the max. block length, the data will be transmitted in block form (see section 3.3).

SLG number of this ASM/SLG station: the value entered here will be placed in the system area of the MDS before command processing begins. It forms an important feature of the entry/exit checking. If an MDS moves outside the field and comes back, the file handler knows that it is the old MDS that has just been processed.

Range of values:
0000 = reserved
0001 ... FFFE = normal SLG no.
FFFF = Test function:
The next MDS can be the same one despite the NEXT command.

Type of entry/exit checking (1 byte, ASCII):
0 = MDS recognition by field scanning; NEXT command is mandatory (see section 5.1)
1 = Without entry/exit control; timeout on ASM 421 (see section 5.2)
2 = MDS control via 1 DI and field scanning (see section 5.3)
3 = MDS control via 2 DIs (see section 5.4)
4 = Without entry/exit control (see section 5.5)
5 = Special operating mode: If the check sum changes, system area may only be written, no NEXT (see section 5.6)

'B' = Break (hard reset); all MDS actions are immediately aborted. Data in the MDS may possibly no longer be consistent
'C' = Close (soft reset); The command currently being processed will be terminated and the directory updated. The data in the MDS are guaranteed to be consistent. The acknowledgement of the soft reset may be delayed.
Acknowledgement:

<table>
<thead>
<tr>
<th>CC Status</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
<th>Error code</th>
<th>ANW</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>'X'</td>
<td>0001</td>
<td>005</td>
</tr>
</tbody>
</table>

Proximity status (1 byte ASCII)
- ‘0’ = no MDS in the SLG field
- ‘1’ = MDS in SLG field

The error code is normally 0000. These 4 characters only contain information if the status byte (byte 5) shows an error.

**Important:**
The ASM 421 must receive a RESET telegram from the user whenever power is restored. Any other type of telegram will be rejected with an error.

A RESET command sets the parameters for the file handler on the ASM 421, which starts to function (the single blinking of the red LED ceases and the yellow LED must come on, if the connection to the SLG is OK). The RESET command is required when a command or a sequence of telegram blocks is to be aborted.

If a command is terminated by the file handler with an error, **no** RESET command is necessary.

**Please note:**
A RESET command declares the current MDS directory on the ASM to be invalid. For the next MDS to be processed, the directory must in each case first be read from the MDS. To avoid the time delay inherent in reading the directory, the MDS directory can be loaded by the user into the ASM with the LOAD command immediately after the RESET command (see section 3.4.16).

**Definition of \( \texttt{tABTAST} \):**
\( \texttt{tABTAST} \) is parameterized as a hex value.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Time value: 00-3F</th>
<th>Time factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td>00 = 0.01 sec</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>01 = 0.1 sec</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>10 = 1 sec</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>11 = 10 sec</td>
</tr>
</tbody>
</table>

For detailed explanation of \( \texttt{tABTAST} \), see the configuration manual for SLG 44/MDS 507/MDS 407E.
3.4.16 LOAD/MOVE

The LOAD and MOVE commands are optional file handler commands. These allow access times to MDS data following a cold start or restart to be optimized. The time spent reading a directory is eliminated for the first MDS to enter the field. The LOAD/MOVE commands are only meaningful if the same directory is present in all data carriers in the system.

LOAD:

A protected directory is transmitted from the user to the ASM. This command can be issued by the user after the RESET, to load the old directory into the ASM. When the first MDS to arrive is processed, the directory does not then need to be read in from the MDS. The time for the directory to be read does not need to be considered when designing the system. Especially important in dynamic operation: the first MDS to arrive after a RESET can immediately be processed at full transport speed.

The file handler checks the directory transmitted by the ‘L’ command for plausibility. Check sums and directory data must correspond. If this is not the case, the user receives the error message D007 in the acknowledgement. The file handler continues to function normally despite this error message. The directory transmitted by the ‘L’ command will be rejected. The file handler reads the directory from the first MDS to arrive.

Telegram to the file handler:

```
<table>
<thead>
<tr>
<th>0 1 2/3 4 5 6/7 8/9 10 11/12 13 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC CI ADB DBN DBL Data</td>
</tr>
<tr>
<td>xx 00 0000 'L' 'I' ADB 0001 DBL DIR + FAT</td>
</tr>
</tbody>
</table>
```

Data protected in user control by the MOVE command

The data block count is calculated from the length of the data (bytes 11/12). It must be calculated correctly by the user (see example in section 3.4.4).

Acknowledgement:

```
<table>
<thead>
<tr>
<th>0 1 2/3 4 5 6/7 8/9 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC Status ADB DBN DBL</td>
</tr>
<tr>
<td>00 xx 0000 'L' 00 ADB 0001 00</td>
</tr>
</tbody>
</table>
```

Warning: A RESET command deletes the directory stored in the ASM, i.e. the complete directory will be read from the next MDS following a RESET command.

An error message from any command can delete the directory stored in the ASM as well (see chapter 7).

In applications using the LOAD/MOVE commands, a LOAD command must therefore be programmed after each RESET command and after each error message.
**MOVE:**

The MOVE command provides the user with the data in the directory that has just been stored in the ASM (DIR + FAT + checksum). It can be issued by the user after bit 6 is set in the status byte of the acknowledgement of any command. For a more rapid cold start and restart, the user can store the data of the ‘M’ command in non-volatile memory (battery buffered or magnetic media).

**Telegram to the file handler:**

```
<table>
<thead>
<tr>
<th>CC</th>
<th>CI</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>xx</td>
<td>00</td>
<td>0000</td>
<td>'M'</td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00</td>
</tr>
</tbody>
</table>
```

**Acknowledgement:**

```
<table>
<thead>
<tr>
<th>CC</th>
<th>Status</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>xx</td>
<td>0000</td>
<td>'M'</td>
<td>00</td>
</tr>
<tr>
<td></td>
<td>xxxx</td>
<td></td>
<td></td>
<td>DBL</td>
</tr>
<tr>
<td></td>
<td>Length*</td>
<td>DIR + FAT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

The length of the DIR + FAT data can vary. It will be at its maximum when the maximum number of files has been created and the whole MDS is full of data.

\[
\text{Length} = (\text{DIR} + \text{FAT}) + 2 \text{ bytes}
\]

[Byte 11 + 12 of the telegram are already counted in the length]

The actual number of telegram segments is passed to the user here.

* Calculation of the actual amount of data to be moved by the MOVE command (see also table in Chapter 1):

\[
\text{Max. length} = [(\text{No. of blocks} \times 2) + (\text{max. no. of files} \times 14) + 9] \text{ bytes}
\]
3.5 Telegrams from file handler

In normal operation the user receives only one telegram (acknowledgement) if a command has previously been sent to the ASM. In two cases however, the ASM becomes active and sends a telegram to the user asynchronously:

– Start-up message
– Proximity change

3.5.1 Start-up message

Telegram from file handler:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>0000</td>
<td>H'</td>
<td>00</td>
<td>0001</td>
<td>0001</td>
<td>00</td>
</tr>
</tbody>
</table>

(see Chapter 4)

Note:
After sending a start-up message, the file handler firmware will only accept a RESET telegram. All other commands will be rejected with an error!

The file handler sends the start-up message whenever power is restored. If the user is not ready to receive, a number of repeats will be carried out within the framework of the 3964R procedure (see section 3.1).

If the user receives a start-up message during operation, the 24 V supply has failed for a short period. A previously programmed command will no longer be processed by the file handler.

3.5.2 Change in proximity status

A change in proximity status (Z message) will only be signalled by the file handler if this has been specified by setting the DIL switch on the module (see section 2.3).

Telegram from file handler:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2/3</th>
<th>4</th>
<th>5</th>
<th>6/7</th>
<th>8/9</th>
<th>10</th>
<th>11/12/13/14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>0000</td>
<td>Z'</td>
<td>00</td>
<td>0001</td>
<td>0001</td>
<td>05</td>
<td>Error code</td>
<td>ANW</td>
</tr>
</tbody>
</table>

Proximity status of an MDS:

'0' = No MDS present; the MDS has just moved out of the transmission window

‘1’ = MDS present; an MDS has moved into the transmission window of the SLG. (green LED on ASM is lit)

EXCEPTION: If the "connection to SLG" has been broken since proximity (ANW) is reported in this way. As a result when the user sends his command to the ASM, he receives the corresponding error message in the acknowledgement).

No error code is given for a proximity message (ASCII: '0000')

Important:
The proximity change message comes fully asynchronously from the ASM. The user must always be ready and able to accommodate this message.
4 Cold start and restart

Cold start and restart must be performed by the user and the ASM 421 in a coordinated manner. Only then can ASM and user operations be synchronized properly.

Cold start and restart on the ASM 421 side:

After switching on the power supply, the ASM 421 performs a comprehensive self-test. This takes about 2 seconds. During this time the ASM cannot communicate with the user. The ASM will not respond to any telegram from the user (also no NAK at the procedure level).

After the self-test, the ASM sends a start-up message (see section 3.5.1) and is immediately ready to receive.

If the start-up message is not acknowledged by the user (e.g. user is not ready or switched off) the message will be lost. As a result the red LED flashes 28 times.

The ASM now always expects to receive a RESET telegram containing the relevant parameters from the user (see section 3.4.15). Any other telegram will not be processed and will be rejected with an error.

Following the RESET, the ASM is able to process all the commands described in section 3.4.

Cold start and restart on the user (computer) side:

The following structogram shows a secure synchronization procedure for restarting the user program:

<table>
<thead>
<tr>
<th>Computer or PC switched on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send RESET telegram to ASM 421</td>
</tr>
<tr>
<td>Telegram acknowledged correctly ?</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>ASM is synchronized</td>
</tr>
<tr>
<td>Wait max. 3 seconds for start-up telegram from the ASM</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Send RESET telegram to ASM and fetch acknowledgement</td>
</tr>
<tr>
<td>- ASM was not switched off</td>
</tr>
<tr>
<td>- ASM has performed its self-test more quickly than the computer; the start-up telegram from the ASM is lost</td>
</tr>
<tr>
<td>- ASM is switched off or defective</td>
</tr>
<tr>
<td>- Message sent to user</td>
</tr>
<tr>
<td>- Start synchronization procedure with ASM cyclically</td>
</tr>
</tbody>
</table>
General evaluation of a start-up telegram by the computer:

The start-up telegram can be sent from the ASM to the user at any time (e.g. following a short power failure). To ensure stable operation, the user must always be in a position to accept and evaluate the incoming telegram.

<table>
<thead>
<tr>
<th>Telegram received from ASM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start-up message ?</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Send RESET telegram to ASM and fetch acknowledgement</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td>Correct acknowledgement; evaluation of telegram</td>
</tr>
<tr>
<td>Proximity message (Z telegram)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Was command active on the ASM ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Command is aborted</td>
</tr>
<tr>
<td>Fault message to user</td>
</tr>
<tr>
<td>Reissue command to ASM</td>
</tr>
<tr>
<td>No</td>
</tr>
</tbody>
</table>

Ready to receive next telegram
5 Checking entry and exit of MDS in SLG transmission window

This chapter describes various procedures for recognizing when an MDS is within range of an SLG. The user can use the procedure most appropriate for the application with the RESET command: EAKO (see section 3.4.15).

The following table shows the various types of entry/exit checking:

<table>
<thead>
<tr>
<th>EAKO</th>
<th>MDS entry</th>
<th>MDS exit</th>
<th>Command executed</th>
<th>NEXT command necessary?</th>
<th>Continuous MDS flow control by file handler</th>
<th>Has filehandler recognized defective MDS?</th>
</tr>
</thead>
<tbody>
<tr>
<td>“0”</td>
<td>Field scanning</td>
<td>Field scanning</td>
<td>If MDS present</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>“1”</td>
<td>User</td>
<td>User</td>
<td>Immediate; error if no MDS in field</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>“2”</td>
<td>DI0 on ASM</td>
<td>Field scanning</td>
<td>If MDS is present via DI0; error if field scanning does not detect MDS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>“3”</td>
<td>DI0 on ASM</td>
<td>DI1 on ASM</td>
<td>If MDS present</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>“4”</td>
<td>User</td>
<td>User</td>
<td>If MDS detected by field scanning</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>“5”</td>
<td>User</td>
<td>User</td>
<td>If MDS detected by field scanning</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

General procedure for an SLG:

For each SLG that is to process an MDS, the station-specific SLG number is written to the system area of the MDS. The SLG number is written automatically before the first command to be executed on the MDS. The file handler recognizes from the SLG number whether a new or an old data carrier is currently present in the transmission window of the SLG.

New MDS: SLG numbers (SLG no. on MDS and station-specific SLG no.) do not correspond
Old MDS: SLG numbers correspond
Current MDS: All active commands apply to this MDS
Next MDS bit: Internal status flag in the file handler
ANW bit: Presence bit; the status of this bit is indicated by the green LED and on the backplane connector (z18). As an option, the user can receive an ANW message if the ANW bit changes (Z message: switch-selectable parameter).

The old MDS remains current (all commands relate to this MDS) until it is closed by a NEXT command. All subsequent commands refer on this SLG to the next new MDS. This is achieved by having the file handler set the “next MDS bit” when it receives a NEXT command. If the old MDS reenters the SLG transmission window as the next MDS (SLG numbers correspond), no command is issued (the “next MDS bit” remains set). When a new MDS enters the transmission window, the “next MDS bit” will be reset and the station-specific SLG number will be written to the system area of the MDS (the new MDS thus becomes the old/current MDS).
5.1 MDS recognition by field scanning

EAKO = '0'

How field scanning works

The SLG scans the surrounding area for the presence of a data carrier. If the file handler recognizes a data carrier, the ANW bit will be set and a Z telegram sent to the user. After the MDS has left the transmission window, the ANW bit will be reset and a Z telegram will be sent to the user, while the SLG scans the field for the next MDS. If the data carrier remains exactly on the border of the transmission window, a hysteresis function ensures the ANW bit does not continuously toggle between on/off. This hysteresis function is handled by the processor on the ASM. Read/write commands are handled completely transparently by the ASM and have no effect on proximity detection.

Hysteresis field for proximity detection

Transmission window: Data exchange between MDS and SLG

h: Hysteresis: Area in which a just set ANW bit remains set.

L, B: Dimensions of the transmission window of an SLG read/write device at operational distance to MDS. (See manual for configuration, installation and service.)

h = 0.1 ... 15 mm (depends on MDS type)

a: The point at which the mobile data carrier is detected by the SLG. The pending MOBY command will now be processed on the MDS. The presence bit is set. The user can receive an ANW message if required (Z message).

b: The MOBY command must be completed by this point as the data carrier is about to leave the transmission window. The presence bit is still set.

c: The presence bit is reset. The user receives a Z message. The MDS has passed out of range of the SLG.
5.2 Without entry/exit monitoring

EAKO = ‘1’

Schematic layout:

In this mode of operation, the user must ensure (e.g. using proximity switch I) that an MDS is guaranteed to be present in the transmission window of the SLG before an MDS command is sent to the ASM. Any command now sent to the ASM will be executed immediately. If the ASM 421 does not find an MDS (presence bit = 0), the command will immediately be terminated with an error. This also occurs if the MDS is defective. No NEXT command is required when MDS processing is complete. The computer is responsible for controlling the entry and exit of the MDS.

Proximity switch II at the exit point is not absolutely necessary. The notification that an MDS has left the transmission window is provided by the ASM 421 through the ANW bit and/or the Z message.
5.3 MDS control using field scanning and 1 DI

EAKO = ‘2’

Schematic layout:

The entry of an MDS is detected by a proximity switch connected to DI 0. The switch should be positioned so that the MDS is guaranteed to be in the transmission window if the switch is activated. If this is not the case, or if the MDS is defective, an error will be reported. After processing the MDS, it is imperative that the user send a NEXT command to the ASM. The file handler assumes complete and continuous control of the MDS.
5.4 MDS control using 2 DIs

| EAKO = '3' |

Schematic layout:

Entry (E) of an MDS is detected by a proximity switch connected to DI 0. The MDS does not have to be within the transmission window of the SLG at this time. Command processing begins as soon as the MDS enters the transmission window. The MDS can move anywhere between points A and E without causing an error. When the MDS reaches point A (exit), the ASM 421 receives an impulse via DI 1. The ANW bit is reset. The MDS has passed the Exit.

The distances from the SLG to the entry and exit proximity switches can largely be chosen at will by the user. Note, however, that two MDSs can never be between points E and A at the same time.

After processing each MDS, it is imperative that the user sends a NEXT command. The file handler assumes complete and continuous control of the MDS. An error will be reported if:

- the MDS is defective
- two MDSs in succession have passed point E (or the same MDS twice)
- two MDSs in succession have passed point A (or the same MDS twice)
- if no NEXT command is issued following an E-A sequence

Note:
After a RESET, the presence of an MDS will only be detected (ANW bit = 1) if it is already within the transmission window of the SLG!
5.5 Without entry/exit monitoring

In this mode of operation, the user assumes control of the MDS. Proximity switch I signals to the computer that an MDS has entered. The MDS does not have to be within the transmission window of the SLG at this time. Command processing begins as soon as the MDS enters the transmission window. The user can check this by interrogating the ANW bit or the Z message. Proximity switch II at the MDS exit point is not absolutely necessary.
5.6 Special Operating Mode (Test Operation)

EAKO = '5'

In this mode of operation absolutely no checking of MDS entry/exit is carried out by the file handler. The MDS control must be handled entirely by the user. If a command is issued to the file handler, it will be stored until an MDS arrives in the transmission window of the SLG. The file handler then commences the processing of the command. When executing commands that do not change the DIR + FAT structure (i.e. no change to the checksum), the system area is not updated. This increases the processing speed in especially time-critical applications, e.g. ECC operation of EEPROM MDS.

Operating with EAKO 5 automatically precludes the use of the NEXT command.

**Note:**

The use of EAKO 5 is a special mode of operation and it should therefore be regarded as test operation.
5.7 Diagram showing possible MDS states at an SLG

The following codes are used in the diagram on the following page:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD</td>
<td>MDS absent</td>
</tr>
<tr>
<td>MDN</td>
<td>new/next MDS present</td>
</tr>
<tr>
<td>MDA</td>
<td>current or old (no longer current) MDS present</td>
</tr>
<tr>
<td>BD</td>
<td>Command present (the user has already transmitted an MDS command to the MDS)</td>
</tr>
<tr>
<td>BDN</td>
<td>Command not present (the ASM is not processing an MDS command from the user)</td>
</tr>
<tr>
<td>BIA</td>
<td>Command in progress (a command is now being executed)</td>
</tr>
<tr>
<td>ND</td>
<td>“Next MDS bit” set in the file handler (the ASM has received a NEXT command from the user). The processing of all subsequent commands takes place only with the next/new MDS.</td>
</tr>
<tr>
<td>NDN</td>
<td>“Next MDS bit” not set in the file handler (the ASM has not yet received a NEXT command).</td>
</tr>
</tbody>
</table>

Two process sequences will be described as examples:

A  
EAKO = 0; an MDS is present in the transmission window of the SLG on restart. The file “switch” is to be read dynamically by the SLG. The computer sets a switch (points) and then waits for the next MDS.

1. After a Restart and a RESET command from the user, the file handler writes the SLG no. to the MDS and goes into status (3). ANW bit = 1.
2. The user sends the READ “switch” command to the ASM. The file handler starts the command and changes to status (27).
3. After executing the command, the ASM sends the data it has read and goes into status (3).
4. After evaluating the data the user sets a switch (points). The MDS has meanwhile left the transmission window of the SLG: status (5); ANW bit = 0
5. The user sends a NEXT command: status (11). The processing of the MDS is now complete.
6. Immediately after the NEXT command, the user sends the command READ “switch”. The command is temporarily stored in the ASM: status (23). This status remains until a new MDS enters the transmission window of the SLG.
7. A new MDS enters the transmission window of the SLG: ANW bit = 1. The file handler begins processing immediately: the SLG number is written to the MDS and the execution of the command READ “switch” initiated: status (27).
8. This point is identical to point 3. The MDS processing cycle restarts from the beginning.

Status diagram for this application, provided no error occurs:
B EAKO = 1; The user assumes control of the MDS. No MDS is in the transmission window of the SLG on restart. Processing of the MDS takes place statically. The process consists of the commands: read “file A”, read “file B” and write “file C”.

1. After restart and RESET from the user, the file handler has the status \( 11 \).
2. The user detects the presence of a new MDS via a proximity switch connected to a digital input (DI) and stops the conveyor. At the same time, the file handler detects a new MDS: ANW bit = 1. The file handler begins processing the MDS: the SLG no. is written to the MDS. The file handler changes to status \( 3 \).
3. The user begins to process the commands: The command READ “file A” is sent to the ASM and is started by the file handler: Status \( 27 \).
4. Acknowledgement of command READ “file A” sent to the user: status \( 3 \).
5. Command READ “file B” to ASM: status \( 27 \).
6. Acknowledgement of command READ “file B” sent to the user: status \( 3 \).
7. Command WRITE “file C” to ASM: status \( 27 \).
8. Acknowledgement of command WRITE “file C” sent to the user: status \( 3 \). The user has now completed the processing of the MDS and restarts the conveyor.
9. The MDS leaves the transmission window of the SLG: ANW bit = 0; status \( 5 \).
10. The user detects the presence of a new MDS via a proximity switch on DI and stops the conveyor. At the same time, the file handler recognizes the presence of a new MDS: ANW bit = 1; status \( 1 \).
11. The user begins to send off the commands: the command READ “file A” is sent to the ASM. The file handler begins processing by writing the SLG no. to the MDS. Finally the READ command is issued: status \( 27 \).
12. This point is identical to point 4. The MDS processing cycle restarts from the beginning.

Status diagram for this application, as long as no error occurs:
6 System and transmission times

The following diagram shows the various timings inherent in a system:

**Internal PC times:**
- Processing time of user program
- Reaction times for acknowledgement of 3964R procedure

These times depend on the user system and cannot be influenced by MOBY.

**Serial data transmission from/to ASM 421:**
- 3964R procedure
- Baud rate:
  - 2400 Bd = 4.6 ms/byte
  - 4800 Bd = 2.3 ms/byte
  - 9600 Bd = 1.1 ms/byte
  - 19200 Bd = 0.6 ms/byte

**File handler processing time:**
ASM 421 processor run time. This is very short and can generally be ignored (< 1 msec).

**Data transmission between file handler and MDS:**
- The gross data rate is 19200 baud.
- Net data rate as shown in the following table:

<table>
<thead>
<tr>
<th>K</th>
<th>tB</th>
<th>Oper. type</th>
<th>Memory*</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>1</td>
<td>FH</td>
<td>32/2 KB R (S/L)</td>
</tr>
<tr>
<td>140</td>
<td>1.4</td>
<td>FH</td>
<td>8 KB E (L)</td>
</tr>
<tr>
<td>140</td>
<td>3.8</td>
<td>FH</td>
<td>8 KB E (S)</td>
</tr>
<tr>
<td>140</td>
<td>2.6</td>
<td>FH w. ECC</td>
<td>32/2 KB R (L)</td>
</tr>
<tr>
<td>220</td>
<td>3.5</td>
<td>FH w. ECC</td>
<td>8 KB E (L)</td>
</tr>
<tr>
<td>280</td>
<td>8.6</td>
<td>FH w. ECC</td>
<td>8 KB E (S)</td>
</tr>
</tbody>
</table>

1) Transmission time per byte

---

* R=RAM; E=EEPROM; S=write; L=read

1) Transmission time per byte
Example:
You want to know how much time elapses from the start of a read operation until the data is in the PC memory. The file to be read has a length of 350 bytes and is to be transmitted in 2 blocks. The maximum block length to the PC is 255 bytes. The mobile data carrier is an MDS 505 (32 kB). It is in the transmission window of the SLG. The data rate between PC and ASM is 4800 baud.

The command sequence can be divided into the following time slices:

<table>
<thead>
<tr>
<th>Description</th>
<th>Time in ms</th>
<th>See diag.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Issuing the read command and start of command by the 3964R driver in the PC</td>
<td>Application</td>
<td>A</td>
</tr>
<tr>
<td>2) Transmission of the command to the ASM:</td>
<td>71</td>
<td>B</td>
</tr>
<tr>
<td>25 user bytes + 6 procedure bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(response times in PC set to zero)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3) File handler processing time</td>
<td>—</td>
<td>C</td>
</tr>
<tr>
<td>4) Processing time of the MDS by the file handler</td>
<td>(940)</td>
<td>D</td>
</tr>
<tr>
<td>a) on the first occasion, when the MDS directory does not yet exist in the ASM</td>
<td>304</td>
<td></td>
</tr>
<tr>
<td>b) reading of 244 bytes file data (60 ms + n * 1 ms)</td>
<td>600</td>
<td>B</td>
</tr>
<tr>
<td>5) Transmission of data to the PC:</td>
<td>Application</td>
<td>A</td>
</tr>
<tr>
<td>(244 + 11) bytes user data + 6 procedure bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(the acknowledgement times in the PC are set to zero)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6) Processing time in the PC and issuing the second command segment to the 3964R driver</td>
<td>39</td>
<td>B</td>
</tr>
<tr>
<td>7) Transmission of the second command segment to the ASM:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 user bytes + 6 procedure bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(response times in PC set to zero)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8) File handler processing time</td>
<td>—</td>
<td>C</td>
</tr>
<tr>
<td>9) Processing time of the MDS by the file handler</td>
<td>106</td>
<td>D</td>
</tr>
<tr>
<td>Reading of 106 bytes (n * 1 ms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Reading the directory is definitely no longer necessary; the constant time for the processing of the system area of an MDS is not required for each subsequent command segment)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10) Transmission of data to the PC:</td>
<td>283</td>
<td>B</td>
</tr>
<tr>
<td>(106 + 11) bytes user data + 6 procedure bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(acknowledgement time in the PC is set to zero)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11) Processing time in the PC until the data is available to the user</td>
<td>Application</td>
<td>A</td>
</tr>
<tr>
<td>Total time of the read operation</td>
<td>1403</td>
<td></td>
</tr>
<tr>
<td>(the times of the PC are set to zero; the directory already exists on the ASM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>→ Read times between ASM and MDS</td>
<td>410</td>
<td></td>
</tr>
<tr>
<td>→ Communications between PC and ASM</td>
<td>993</td>
<td></td>
</tr>
</tbody>
</table>
7 Troubleshooting and error messages

An error message from the file handler always has the same format. The presence of an error is always indicated in the status byte (5th byte) of the acknowledgement telegram. It is 4 bytes long and in ASCII format.

Acknowledgement with error:

<table>
<thead>
<tr>
<th>CC Status</th>
<th>ADB</th>
<th>DBN</th>
<th>DBL</th>
<th>Error code</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 xx xxxx</td>
<td>x'</td>
<td>01</td>
<td>xxxx</td>
<td>xxxx 04</td>
</tr>
</tbody>
</table>

Error number: 000 ... 999

Error group:
- A = Protocol error
- B = SLG error
- C = MDS error
- D = Task related error
- E = Directory error
- F = File related error
- K = Error in the OUEUE-WRITE command

When terminated within a block sequence the block in which the error was detected is shown here.

A block sequence will automatically be interrupted by an error.

Error flag is set (Bit 0 in status byte = 1)
The error flag must always be interrogated by the user

Command code showing where the error occurred

In general:
- An error terminates block creation
- There are no “ignore” or “retry” options
- A “RESET” is not necessary following an error
- The parameters ADB and DBN contain the block number in which the error occurred
- If using the LOAD command:
  a LOAD command should be programmed after each error
  (As a rule, the directory in the ASM is invalid if the error occurred during a directory operation. If the error occurred in the data area, the directory in the ASM remains valid. If many errors occur, however, the user cannot know exactly whether the error occurred during the DIR processing or while processing the user data).
The following table contains a summary of all the errors from the ASM 421:

<table>
<thead>
<tr>
<th>Flashes of red LED</th>
<th>Error messages from the file handler</th>
<th>Kind of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0 06</td>
<td>Unknown error, the command code CC is invalid</td>
<td>PROTOCOL ERROR</td>
</tr>
<tr>
<td>A0 11</td>
<td>DBN not equal to 1 in first command block; CC or DBN incorrect for subsequent blocks; DBL is wrong or does not fit the transferred data length.</td>
<td></td>
</tr>
<tr>
<td>A0 15</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>A0 16</td>
<td>Command from another user being processed</td>
<td></td>
</tr>
<tr>
<td>B0 01</td>
<td>Fault in connection to the SLG</td>
<td>SLG ERROR</td>
</tr>
</tbody>
</table>
| B0 02             | 1. Error when EAKO = 1; no MDS present (Timeout)  
2. New MDS in transmission window; no NEXT command but command started (EAKO = 0,2)  
3. New MDS has left station without NEXT (EAKO = 0,2)  
4. EAKO = 2: MDS entry detected; however no MDS is to be found within the transmission window | |
| B0 05             | EAKO = 3: An MDS has exited the transmission window without having entered it previously. | |
| B0 06             | EAKO = 3: new MDS arrives while the old one is still in the transmission window | |
| B0 07             | EAKO = 3: 2 x MDS entered without one having exited | |
| C0 02             | RAM error in MDS | MDS ERROR |
| C0 06             | Proximity error | |
| C0 07             | Incorrect parameters in TRACE or FORMAT / command cannot be interpreted | |
| C0 08             | Too many sync attempts | |
| C0 09             | Too many send errors | |
| C0 10             | CRC send error | |
| C0 11             | reserved | |
| C0 12             | FORMAT, cannot initialize MDS | |
| C0 13             | FORMAT, timeout | |
| C0 14             | FORMAT, not initialized | |
| C0 15             | CMD address error | |
| C0 16             | ECC error | |
| C0 17             | General driver error | |
| D0 01             | Only RESET command permissible | TASK-RELATED ERROR |
| D0 05             | Invalid file or volume name | |
| D0 07             | The directory specified in the LOAD command is invalid; MOVE command isn’t possible | |
| D0 09             | Incorrect parameter in RESET command | |
| D0 14             | CREATE and WRITE: the user data area in the MDS is full | |
| D0 15             | Only FORMAT command possible; MDS not identified | |
| D0 18             | Start address in the command is outside the data area (start address > file length) | |
| D0 22             | Directory and/or FAT-changing access to an MDS protected with COVER is illegal. | |
| D0 23             | COVER: MDS name is incorrect. | |
| E0 01             | The MDS type is incorrect or unsuitable for the selected mode of operation (ECC). | DIRECTORY-RELATED ERROR |
| E0 02             | CREATE command; no more directory entries available | |
| E0 03             | CREATE command; file already exists in directory | |
| E0 05             | FAT block error detected in READ or WRITE; FAT is incorrect. | |
| F0 01             | Specified data not present | FILE-RELATED ERROR |
| F0 05             | WRITE error in file with “Read only” or “Write once” attribute | |
| F0 06             | Incorrect access rights or switch setting | |
| Kx xx             | 1. xxx = Number of the incorrect filr entry (See QUEUE-WRITE)  
2. QUEUE-WRITE command in parameterized incorrectly.  
3. The file to be set up already exists. | QUEUE-PARAMEETERIZATION |
| 27                | The ASM has a receive error | SERIAL INTERFACE |
| 28                | The ASM has a send error | |

* Errors with no digits in this column do not cause the red LED to flash.
Full error description

A0 06: The command code for the started command is not valid (not defined). The correct CC must be entered (see section 3.4).

A0 11:  
– for the first command block: DBN (= byte 8/9 in the telegram) does not have the value 0001.
– for subsequent block: DBN from the user is not in the correct ascending sequence or the CC parameter (= byte 4) is not appropriate for the command now being processed.

A0 16: The file handler is currently processing a command from another user. Command processing will be delayed until the other user is finished. The command may have to be issued again.

B0 01: Fault in connection to the SLG.
– cable between ASM and SLG is incorrectly wired or cable is damaged
– 24 V supply is not connected or is switched off
– fuse on the ASM is defective
– Hardware fault: ASM 421 or SLG
This error does not occur at the start of the system commands (RESET, NEXT, ASM STATUS).

B0 02: EAKO 1, 2:  
– A command has been started, but there is no MDS in the transmission window of the SLG.
– MDS 507; dialog battery is dead (Batt2 bit is not set; measure battery voltage).

EAKO 0, 2, 3:  
– the old/current MDS is out of the transmission window and the next/new MDS has entered the transmission window. A command has been started (not NEXT). This command refers to the new MDS, but the old/current MDS has not been terminated with NEXT.
– a new MDS entered the transmission window of the SLG and left it again without any command being processed. (MDS has “slipped through”).

EAKO 2: The arrival of an MDS was recognized by the proximity switch on DI 0; however no MDS is to be found in the transmission window of the SLG.

B0 05: Only when EAKO (I/O Control) = 3:
No MDS is to be found between points E and A; in this state the departure proximity switch has been activated.
– an MDS has exited the transmission window without having entered it previously.
– after the departure of an MDS, the MDS has been moved back and the departure proximity switch activated a second time.

B0 06: Only when EAKO = 3: MDS generates an entry signal at DI0 but the old MDS has not exited completely.
– entry (DI0) and exit (DI1) proximity switches are being triggered simultaneously
– entry and exit are too close together

B0 07: Only when EAKO = 3: two MDS entries have occurred at DI0, but no MDS exited at DI1
– entry and exit are too close together
– an MDS is stationary directly after entry and is moved backwards slightly by the conveyor
C0 02: Memory error message from MDS. The MDS has not yet been written or has lost its memory content due to battery failure (not with EEPROM MDS). Therefore:
– change the MDS (if the battery monitoring bit is set)
– initialize the MDS with the STG 4F
– format the MDS using the FORMAT command

C0 06: During certain important operations (e.g. writing system area of MDS, formatting MDS) the MDS must not leave the transmission window of the SLG, as otherwise the command will terminate with this error. Therefore:
– issue the command again
– MDS is standing in the border zone of the SLG transmission window
– at EAKO = 1: The MDS is not in the transmission window of the SLG at the start of a command.

C0 07: – The FORMAT or TRACE commands have been issued with incorrect parameters. The physical address requested does not exist in the MDS (MDS memory is smaller than specified in the command)
– With READ/ WRITE/ UPDATE: the pointer in the FAT is incorrect; it is pointing to a block that does not exist in the MDS.

C0 08: Field interference at the SLG. The SLG is receiving interference impulses from the surrounding area
– external interference field; the interference field can be detected by the “inductive field indicator” of the STG.
– the distance between two SLGs is too small and does not conform to the design guidelines
– the connection cable to the SLG is subject to interference, is too long or does not conform to specification
Or MDS 507; dialog battery is dead
– Check Batt2 bit
– Measure voltage on battery.

C0 09: Too many send errors have occurred. The MDS could not receive the command or the data from the SLG correctly, despite several attempts
– the MDS is standing exactly on the edge of the transmission window
– data transmission to the MDS is being affected by external interference

C0 10: – CRC send error. The monitoring system has detected a data transmission error. Cause of error as for C0 08.
– The MDS reports CRC errors very frequently. (MDS is on the edge of a transmission window. MDS or SLG is defective.)

C0 12: The MDS cannot execute the FORMAT command. The MDS is defective.

C0 13: The MDS must be in the SLG field when formatting, otherwise a timeout error occurs, i.e.:
– the MDS is standing exactly on the edge of the transmission window
– the MDS is using too much current (defective).
– EEPROM-MDS type parameterized incorrectly in FORMAT
Or MDS 507; dialog battery is dead
– Check Batt2 bit
– Measure voltage on battery.
C0 14: The MDS memory cannot be written
   – the MDS has a smaller memory than specified in the FORMAT command, i.e. enter the correct parameters for this type of MDS
   – the MDS memory is defective
   – an EEPROM MDS has been written too often and has reached the end of its life

C0 15: Address error. The address area of the MDS has been exceeded
   – the MDS is not of the correct type

C0 16: An ECC error has occurred. Data cannot be read from the MDS
   – the MDS has lost its data (MDS defective)
   – the MDS was not formatted by the ECC driver. Reformate the MDS
   – an EEPROM MDS has reached the end of its life. The data has been lost. Replace the MDS
   – The MDS left the field during the write access. The MDS is not positioned correctly.

C0 17: The file handler is not working correctly
   – check command format and command sequence
   – the ASM 421 hardware (firmware) is defective (faulty).

D0 01: The file handler will only accept RESET commands
   – the file handler has not yet been initialized by a RESET command
   – this situation can only be remedied by a RESET command

D0 05: The FORMAT, CREATE, ATTRIB, WRITE, UPDATE, COVER and QUEUE-WRITE commands have been issued with incorrect parameters
   – FORMAT with invalid volume name or MDS type
   – CREATE with invalid file name
   – invalid file attribute
   – WRITE/UPDATE with length 0 (DLNG=0)
   – QUEUE-WRITE with invalid option
   – COVER with invalid user (Only 1 and 2 are valid.)

D0 07: The data transferred by the LOAD command is incorrect:
   – the data from DIR + FAT do not match the transmitted checksum.
   – the DLNG is parameterized incorrectly in LOAD.
   MOVE command: the DIR + FAT do not fit the checksum. A MOVE command cannot be carried out. The data structure on the MDS may be faulty. ➔ Format the MDS.

D0 09: The RESET command has been transmitted to the file handler with incorrect parameters
   – check bytes 11 to 17 of the telegram as shown in section 3.4.15

D0 14: WRITE command: there is no longer enough memory available in the MDS. Not all the data has been written to the MDS
CREATE command: no data blocks can be reserved when creating a file. No more blocks are free

D0 15: The MDS could not be identified by the file handler. Reformat the MDS.

D0 18: The physical address requested lies outside the file. There is an error in the FAT. Reformat the MDS.

D0 22: The MDS is protected by the COVER command. A write command (e.g. UPDATE, CREATE) must not modify the memory structure and is therefore rejected.
D0 23: COVER command: the MDS name specified in the command does not correspond to the actual MDS name.

E0 01: 
– the type of MDS present at the SLG does not correspond to the ECC operating mode selected. The MDS must be reformatted to correspond to the desired ECC operating mode.
– the MDS is not a file handler MDS. Format the MDS.

E0 02: No more directory entries available. The file specified in the CREATE command cannot be created.
→ see table in Chapter 1

E0 03: The file specified with the CREATE command already exists in the directory (no duplicate names).

E0 05: A FAT block sequence error has been detected with a READ or WRITE command. The file allocation table (FAT) is incorrect. The MDS must be reformatted.

F0 01: Incorrect address with TRACE command.
– the file addressed by a command (e.g. WRITE) does not exist in the directory. The file must be created with CREATE.
– check file name (perhaps not in ASCII format)

F0 05: Attempt to write (WRITE command) to a read-only file (protected by an appropriate attribute).
– change the access rights with the ATTRIB command and then issue the WRITE/DELETE/UPDATE command again

F0 06: The RWD switch on the ASM does not give sufficient rights for this command. The command will be ignored → check switch according to section 2.3.

Kx xx: QUEUE WRITE parameter incorrect (check command telegram)
Option 0000 Hex: The file entry with number xxx or xxx + 1 parameterized in the command telegram is incorrect. The method of counting the file entries in the command telegram begins with 1.

Option 0001 Hex: The file entry with number xxx or xxx + 1 parameterized in the command telegram contains a file name that already exists on the MDS. The method of counting the file entries in the command telegram begins with 1.

Note: The file entries are counted in decimal format.
8 Warnings

English

⚠️ Warning

Hazardous voltages are present in this equipment during operation.

To ensure safe operation of the equipment, maintenance shall only be performed by qualified personnel in accordance with the instructions in the MOBY catalog\(^1\) and technical description.

Failure to observe these instructions can result in death, severe personal injury or substantial damage to property.

The following instructions and those on all product labels must be followed when carrying out any maintenance work.

- Always disconnect and earth the equipment before starting any maintenance.
- Use only spare parts authorized by the manufacturer.
- The servicing intervals as well as the instructions for repair and replacement shall be duly observed.
- A lithium battery is contained in mobile data memories with RAM. The following instructions must be observed:
  
  To avoid the risk of fire, explosion and severe burns, the battery should not be recharged, dismantled, exposed to heat over 100 degrees Celsius, ignited, or brought into contact with water.

  The special instructions must be followed when using heat-resistant data storage media.

\(^1\) Should you not be in possession of the MOBY catalog, it can be obtained through your local Siemens office.

Deutsch

⚠️ Warnung

Beim Betrieb elektrischer Geräte stehen zwangsläufig bestimmte Teile dieser Geräte unter gefährlicher Spannung.

Sicherer Betrieb der Geräte setzt voraus, dass diese von qualifiziertem Personal sachgemäß unter Beachtung der im MOBY-Katalog\(^1\) und der technischen Beschreibung enthaltenen Hinweise eingesetzt werden.

Bei Nichtbeachtung können Tod, schwere Körperverletzung oder erheblicher Sachschaden die Folge sein.

Beachten Sie daher auch bei Instandhaltungsmaßnahmen an diesem Gerät alle hier und auf dem Produkt selbst aufgeführten Hinweise.

- Vor Beginn jeglicher Arbeiten ist das Gerät vom Netz zu trennen und zu erden.
- Es dürfen nur vom Hersteller zugelassene Ersatzteile verwendet werden.
- Die vorgeschriebenen Wartungsintervalle sowie die Anweisungen für Reparatur und Austausch sind unbedingt einzuhalten.
- Bei einem mobilen Datenspeicher mit RAM ist eine Lithiumbatterie integriert, hierzu sind folgende Hinweise zu beachten:
  

Beim hitzefesten Datenträger sind die besonderen Hinweise zu berücksichtigen.

\(^1\) Sollten Sie nicht im Besitz des MOBY-Katalogs sein, so kann er über jede örtliche SIEMENS-Niederlassung bestellt werden.
Español

¡Precaución!

Durante el funcionamiento de los equipos eléctricos hay partes de los mismos que se encuentran forzosamente bajo tensión peligrosa.

Un funcionamiento seguro de los equipos presupone que han sido instalados correctamente por personal calificado observando las indicaciones contenidas en el Catálogo de los equipos MOBY y la Descripción técnica.

La observación de dichas indicaciones puede provocar la muerte, graves lesiones o daños materiales considerables.

En los soportes de datos con protección térmica es preciso observar las indicaciones particulares respectivas.

Antes de comenzar cualquier trabajo es preciso secionar de la red el equipo y ponerlo a tierra.

Solo deben utilizarse repuestos autorizados por el fabricante.

Es imprescindible observar los intervalos de mantenimiento especificados así como las instrucciones de reparación y reemplazo.

Evite riesgos de fuego, explosiones y quemaduras graves. La batería no debe ser recargada, desmontada, calentada a mas de 100°C.

Si no dispone del catálogo MOBY, estos pueden pedirse a través de cualquier sucursal local de SIEMENS.

1 Se debe tener en cuenta las indicaciones de seguridad que se encuentran forzosamente bajo tensión peligrosa.

Svensk

Varning

Vid drift av elektrisk utrustning ligger det alltid en farlig spänning på vissa delar av utrustningen.

Säker drift av utrustningen förutsätter att den utförs av kvalificerad perso-

m med uppmärksamhet på anvisningarna i MOBY-katalogen samt de

anvisningarna. Om dessa anvisningar ej beaktas kan följden bli dödsfall, svår kroppskada eller avsevärda materielskador.

Före allt arbete skall utrustningen skiljas från näten och jordas.

Bara reservdelar som godkänts av tillverkaren får användas.

Detta gäller följande anvisningar:

För värmebeständiga datamedier gäller speciella anvisningar, som måste beaktas.

1 Om Ni inte har ett exemplar av MOBY-katalogen så kan åtta beaktas.

3対変の機械も同様に注意を要します。