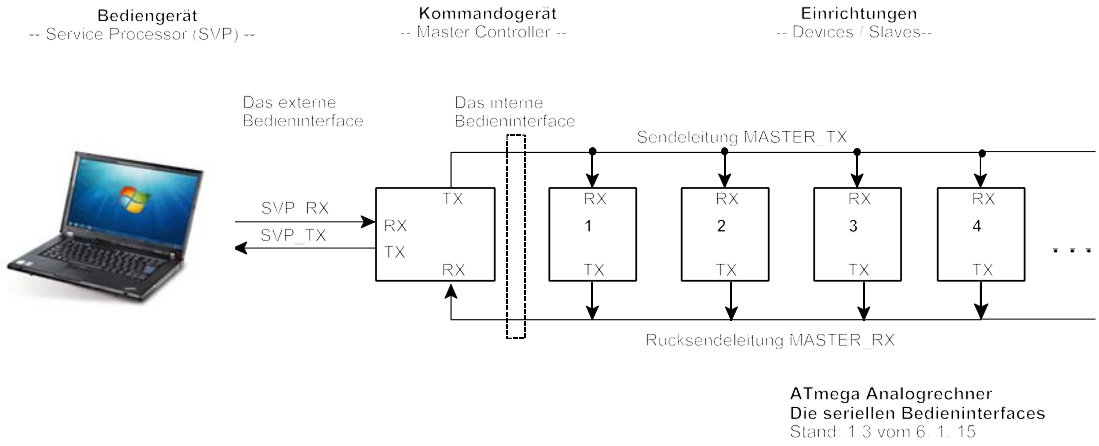
# Projekt Analogrechner
# Das externe Bedieninterface

*Stand: 10. 3. 2015*



## 1. Grundlagen

Das Bedieninterface nutzt die seriellen Schnittstellen der Mikrocontroller. Es gibt zwei Interfaces: das externe und das interne Bedieninterface. Jedes dieser Interfaces hat zwei Signalwege: die Sendeleitung (TX) und die Empfangsleitung (RX). Das externe Bedieninterface verbindet das externe Bediengerät mit dem Kommandogerät, das interne Bedieninterface verbindet das Kommandogerät mit den anderen Einrichtungen. Das externe Interface ist eine Punkt-zu-Punkt-Schnittstelle, das interne Interface ist eine Busschnittstelle. Alle anderen Einrichtungen empfangen die Daten vom Kommandogerät. Zu einer Zeit kann aber nur eine Einrichtung senden. Das Kommandogerät ist der Master, die anderen Einrichtungen sind die Slaves. Die Sendesignale der Einrichtungen werden nach dem Prinzip des Wired OR auf die Rücksendeleitung (MASTER RX) aufgeschaltet. Die Zeichen, die das Kommandogerät sendet, lösen in den anderen Einrichtungen Unterbrechungsbehandlungsroutinen aus. Alle Einrichtungen sehen alle Bytes, die vom Kommandogerät kommen. Nur das Kommandogerät sieht die Bytes, die die jeweils ausgewählte Einrichtung sendet.




Das Kommandogerät steuert das interne Bedieninterface nur dann an, wenn über das externe Bedieninterface entsprechende Kommandos gegeben werden. Das Kommandogerät ist dabei nur Durchreiche oder Signalwandler. Die eigentlichen Bedienfunktionen werden im externen Bediengerät implementiert. Wenn das Bedieninterface arbeitet, ruht das Kommandointerface und umgekehrt. Der Chef des Kommandogeräts ist das Bediengerät, der Chef der Slaves ist das Kommandogerät.

**Die Signale des externen Bedieninterfaces:**

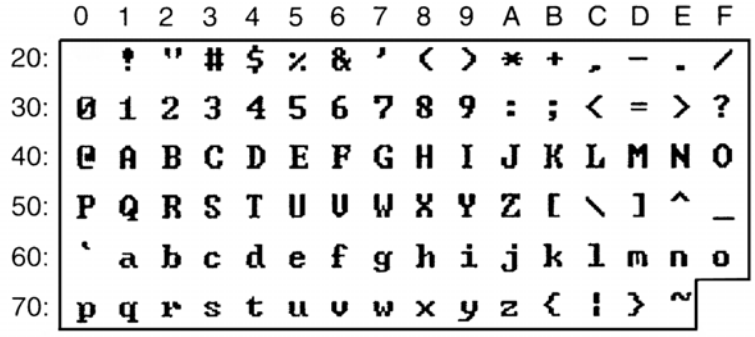
| Bezeichnung | Quelle | Nutzung |
|---|---|---|
| SVP_TX | Kommandogerät | Sendedaten vom Kommandogerät Empfangsdaten für das Bediengerät |
| SVP_RX | Bediengerät | Sendedaten vom Bediengerät, Empfangsdaten für das Kommandogerät |

Typische Ursachen für das Entstehen von Bedienanforderungen sind:

- das Betätigen des Anforderungsschalters der jeweiligen Einrichtung,
- das Auftreten interner Bedingungen (z. B. Vergleichsstop),
- eine Bedienhandlung am externen Bediengerät.

Nur das Bediengerät kann Aktionen an den Bedieninterfaces von sich aus auslösen. Das Kommandogerät wird am internen Bedieninterface nicht autonom wirksam. Das Bediengerät kann das Kommandogerät veranlassen, Kommandos und Daten vom externen zum internen Bedieninterface (und umgekehrt) 1:1 durchzureichen (Direktsteuermodus).

**Datenübertragung**
Die Datenübertragung beruht auf dem ASCII-Code. Binäre Daten werden hexadezimal übertragen (mit zwei Hexadezimalzeichen je Byte). Weitere Kommandos unterstützen die direkte (uncodierte) Übertragung von ASCII-Zeichen. Einige ASCII-Steuerzeichen werden zur Übertragungssteuerung verwendet (reservierte Steuerzeichen). Sie dürfen nicht im Datenstrom vorkommen. Alle anderen ASCII-Steuerzeichen sind im Datenstrom an sich zugelassen. Über das externe Bedieninterface sollten aber nur darstellbare ASCII-Zeichen übertragen werden (ASCII-Codes 21H...7FH).




**Reservierte Steuerzeichen des internen Bedieninterfaces:**

| Zeichen | Code | Wirkung |
|---|---|---|
| CR | 0DH | Allgemeines Abschlußzeichen. |
| CAN | 18H | Alle Einrichtungen rücksetzen und trennen (Puffer und Auswahlzustände). |
| EOT | 04H | Den ausgewählten Slave rücksetzen, aber nicht trennen. |
| ENQ | 05H | Maschinenfehlerreport senden. |
| ETB | 17H | Abschließen und Trennen vom aktuellen Slave. |
| EM | 19H | Hart rücksetzen. |
| BEL | 07H | Allgemeinauswahl (Broadcast). |

| Zeichen | Code | Wirkung |
|---|---|---|
| DC1 | 11H | Abfrage auf anhängige Bedienanforderungen; Ausschalten des Direktsteuermodus. |
| DC2 | 12H | Übergang in den Bedienbetrieb (Bedienschleife). |
| DC3 | 13H | Übergang in den Rechenbetrieb (Kommandoschleife). |
| DC4 | 14H | Einschalten des Direktsteuermodus. |
| ACK | 06H | Positive Antwort auf Anfrage (Anforderung anhängig) oder Kommandoende. |
| NAK | 15H | Negative Antwort auf Anfrage (keine Anforderung anhängig) oder Kommandoende mit Fehlermeldung. |

**Übertragungsreihenfolge:**

1. Hexadezimalzeichen: das erste Zeichen betrifft die höherwertige, das zweite die niederwertige Tetrade. C1H = 1100 0001B.
2. Wörter, Adressen usw.: das niedrigstwertige Byte wird als erstes übertragen (Little Endian).

**Byteübetragung (8 Bits) mit zwei Hexadezimalzeichen:**

| 1. Byte | 2. Byte |
|---|---|
| Bits 7 ... 4 | Bits 3 ... 0 |

**Ein 16-Bit-Wort:**

| 15 12 | 11 8 | 7 4 | 3 0 |
|---|---|---|---|

**Die hexadezimalen Bytes:**

| 1. Byte | 2. Byte | 3. Byte | 4. Byte |
|---|---|---|---|
| 7 4 | 3 0 | 15 12 | 11 8 |

**Daten im Speicher: Register, Objekte und Speicherbereiche**

Die Einrichtungen am Bedieninterface sind verschiedenartig. Die Programme werden sich immer wieder ändern. Damit ist auch stets mit Änderungen der Speicherbelegung zu rechnen. Deshalb müssen die anwendungsseitigen Zugriffe von den konkreten Adressen in der Einrichtung unabhängig sein. Schreib- und Lesekommandos mit direkten Adressen sind eigentlich nur zu Debugging-Zwecken vorgesehen. Die Awendungsprogrammierung sollte ausschließlich mit Objektzugriffen arbeiten. Die wichtigsten Objekte sind vordefiniert und mit fest zugeordneten Zeichen auswählbar. Dies sind die Parameter, die Variablen, die Register sowie die Festwerte der Maschinenidentifikation. Zusätzlich können bis zu 255 Objekte anwendungs- und einrichtungsspezifisch definiert werden. Sie sind über Objektnummern (Object Identifiers) auswählbar, die in der Einrichtung über eine Objekttabelle (Object Reference Table ORT) in Adressen umgesetzt werden. Die Größe der ORT ist fest, die ORT-Inhalte können über Kommandos geändert werden.

**Register**

Wenn in diesem Text von Registern die Rede ist, sind damit Bytes oder Wörter im Speicher gemeint. Jedes Register ist über einen Identifier aufrufbar, unabhängig von seiner Placierung im RAM.

**Parameter**

Im Kommandogerät werden 16 Parameter unterstützt. Jeder Parameter kann über eine Hexadezimalzahl (0...F) angesprochen werden, unabhängig von seiner Placierung im RAM. Parameter 0 ist stets der

Funktionscode, der bestimmt, welche Funktionen in der Kommandoschleife und bei Ausführung des Kommandos "Funktionsauslösung" (X – X) auszuführen sind. Das niedrigstwertige Byte dieses Parameters enthält den primären Funktionscode als Binärzahl[1]. Weitere Parameterbytes können ergänzende Codes, Steuerbits usw. enthalten. Funktionscode 00H bewirkt ganz elementare Abläufe zu Prüf- und Diagnosezwecken.

**Variable**
Im Kommandogerät werden 16 Variable unterstützt. Jede Variable kann über eine Hexadezimalzahl (0...F) angesprochen werden, unabhängig von ihrer Placierung im RAM.

*Hinweis:*
Das Kommandogerät steuert nur Abläufe, rechnet aber nicht selbst. Parameter und Variable des Kommandogeräts sind nur elementare Objekte, die Einstellwerte (Parameter) oder veränderliche Werte (Variable) aufnehmen können. Ein typischer Parameter ist die Anzahl der Rechenschritte eines Rechenzyklus (Endwert), eine typische Variable ist die Nummer des aktuellen Rechenschritts (Zählwert).

**Objekte als frei definierbare Speicherbereiche**
Die Objektdefinition erfolgt in der Objekttabelle (Object Reference Table ORT). Die maximale Anzahl der Objekte wird über eine Festwertparameter vorgegeben. Jedes Objekt wird über eine Objektnummer (Object Identifier) im Bereich von 1 bis 255 angesprochen (1 Byte oder 2 Hexadezimalzeichen). Objekt 0 ist reserviert. Die Objekttabelle enthält Objektdeskriptoren. Jeder Objektdeskriptor ist 5 Bytes lang. Er gibt die Anfangsadresse, die Länge und den Typ des Objekts an.

**Ein Objektdeskriptor:**

| 1. Byte | 2. Byte | 3. Byte | 4. Byte | 5. Byte |
|---|---|---|---|---|
| Adresse, Bits 7...0 | Adresse, Bits 15...8 | Länge, Bits 7...0 | Länge, Bits 15...8 | Typkennung |

Über den Objekttyp könne auch schreibgeschützte Objekte und Objekte in anderen Speichern (Flash, EEPROM) definiert werden. Anfänglich wird nur Objekttyp 0 unterstützt (RAM, kein Schreibschutz).

**Fehlerbehandlung**
Fehler werden angezeigt (Blinken der LED) und registriert (Fehlerregister). Manche Fehler der seriellen Kommandoausführung werden durch Rücksenden von NAK gemeldet. Die Fehlerbehandlung obliegt dem Bediengerät. Fehlerregister werden nur über Kommandos gelöscht.

*Fehleranzeige*
Wenn das Kommandogerät einen internen Fehler erkannt hat, läßt es seine LED schnell rot-grün blinken. Dieses Blinken hat Vorrang vor allen anderen LED-Signalen und kann nur mit Kommandos ausgeschaltet werden.

*Fehlermeldung*
Alle Einrichtungen können Fehler über Bedienanforderungen ans Bediengerät melden.

*Fehlererkennung seitens des Bediengerätes*
Von den Fehlermeldungen der Einrichtungen abgesehen gibt es drei grundsätzliche Möglichkeiten der Fehlererkennung:

---

1: Damit dieser Code auf einfache Weise ausgewertet werden kann (Vergleichen mit Festwerten, Funktionsverzweigung).

1. Durch Zeitkontrolle (Timeout).
2. Durch zyklisches Abfragen. So können im Rahmen von Bedienhandlungen ENQ-Kommandos gegeben werden, um die Fehlerregister abzufragen.
3. Durch Bedienereingriff (z. B. Auslösen der Fehlerbehandlung, wenn LEDs blinken und sonst nichts geschieht (Hängenbleiben)).

**Maschinenzustände im Kommandogerät**
Im Kommandogerät kann jeweils eine von zwei Grundsteuerschleifen laufen: die Kommandoschleife (Hauptsteuerschleife; Main State Loop) oder die Serviceschleife (Bedienschleife).

Das Kommandogerät schaltet hart zwischen beiden Schleifen um. Es wird nur dann umgeschaltet, wenn sich die jeweils aktive Schleife im Warte-Umlauf befindet. Eine Kontextrettung beim Umschalten ist somit unnötig.

Die einzige Möglichkeit einer Einrichtung, eine Bedienanforderung anzuzeigen, besteht darin, das ATN-Signal auf Low zu ziehen. Zuvor muß ein entsprechender Anforderungscode in das Byte ATN_REQUEST geladen werden. Das ATN-Signal bleibt so lange auf Low, bis ein entsprechendes Ausschaltkommando empfangen wird.

Bevor der Slave ATN auf Low zieht, muß er den zugehörigen Anforderungscode speichern. Es sind 255 primäre Anforderungscodes möglich. Bedarfsweise können primäre Anforderungscodes mit zusätzlichen Ddaten ergänzt werden. Solche Datenbereiche werden zweckmäßigerweise als Objekte organisiert.

**Maschinenzustände in der Kommandoschleife**
Das Kommandogerät befindet sich zu einer Zeit in einem von mehreren Maschinenzuständen. Alle Programmabläufe beruhen auf einer Hauptsteuerschleife (Kommandoschleife, Main State Loop), die zum jeweiligen Zustand verzweigt.

**Wodurch können Zustandswechsel veranlaßt werden?**

- durch Betätigen des Run-Halt-Reset- (RHR) -Schalters,
- durch Betätigen des Anforderungsschalters,
- durch Aktivierung der Leitung ATN#.
- durch Aktivierung der Leitung STOP#
- durch eine serielle Anforderung vom externen Bediengerät.

Von Zustand 1 an werden in der Grundsteuerschleife alle Anforderungen abgefragt. In Zustand 0 wird lediglich auf die serielle Anforderung vom Bediengerät gewartet (die Maschine ist noch nicht initialisiert, kann also auch noch nicht rechnen).

**STOP# und ATN#:**

- STOP# = Rechenende. Die Einrichtungen müssen in der Ausführung des Kommandos INITIATE STOP# ausschalten. STOP# löst keine Bedienanforderung aus.
- ATN# löst Bedienanforderung aus. Das Rechnen wird angehalten. Debuggingfunktionen, wie beispielsweise ein Stop auf Bedingung, müssen über ATN# implementiert werden.

**Externe Abfrage (Polling) des Kommandogerätes**
Das externe Bediengerät muß das Kommandogerät zyklisch nach anhängigen Anforderungen abfragen.

**Abfrageprotokoll**
Das Bediengerät sendet DC1. Liegt keine Anforderung vor, antwortet des Kommandogerät mit NAK. Liegt eine Anforderung vor, sendet das Kommandgerät den Inhalt des Ereignisregisters (EVENTS) und abschließend ACK. Danach wird die laufende Arbeit fortgesetzt. Der Ereignisregisterinhalt wird in zwei Hexadezimalzeichen übertragen. Die anhängigen Anforderungen sind aus der niederwertigen Tetrade (Bits 3...0) ersichtlich. Nähere Einzelheiten muß das Bediengerät gesondert abfragen (beispielsweise mit Poll- und Log-Out-Kommandos).

**Externe Bedienanforderungen**
Das Bediengerät sendet DC2. Im Kommandogerät wird das Ereignis EXTERNAL REQUESTgesetzt. Daraufhin geht das Kommandogerät in den Bedienbetrieb über (es beendet zuvor ggf. den letzten Rechenzyklus) und sendet ACK.

**Rechenbetrieb und Bedienbetrieb**
Befindet sich das Kommandogerät im Rechenbetrieb, so läuft die Kommandoschleife (Main State Loop). Befindet sich das Kommandogerät im Bedienbetrieb, so läuft die externe Serviceschleife (Bedienschleife, External Service Loop).

**Grundzustand und Unterbrechungszustand**
Die Kommandoschleife uund die Serviceschleife laufen im Grundzustand. Der Grundzustand ist jederzeit unterbrechbar. Eine Unterbrechung führt dazu, daß ein Unterbrechungsbehandler ausgeführt wird. Dann befindet sich das Kommandogerät im Unterbrechungszustand. Die Unterbrechungsbehandler selbst sind nicht unterbrechbar (ein Behandlungsablauf zu einer Zeit).

**Übergang zum Bedienbetrieb**
Das Kommandogerät geht dann in den Bedienbetrieb, wenn eine entsprechende Anforderung vorliegt. Jedes Kommando vom externen Bedieninterfae erzeugt eine solche Anforderung. Die Anforderung wird in der Kommandoschleife zyklisch abgefragt. Bedienbetrieb heißt, daß die Kommandschleife nicht mehr durchlaufen und das Recheninterface nicht angesteuert wird. Der jeweils vorhergehende Rechenzyklus ist aber noch durchlaufen worden. Das Recheninterface befindet sich in Ruhe. STROBE# ist Low (inaktiv), RDY ist High, der Kommandocode auf CMD2...0 ist Null .

**Rückkehr zum Rechenbetrieb**
Das Bediengerät sendet DC3, das Kommandogerät antwortet mit ACK. Daraufhin erfolgt eine Rückkehr aus der externen Serviceschleife in die Kommandoschleife. Dann beginnt das Kommandogerät wieder damit, das Recheninterface anzusteuern. Wenn das Kommandogerät in die Serviceschleife eingelaufen ist, kann es nur mit DC3 in die Kommandoschleife zurückgeführt werden. Falls erforderlich, muß das Bediengerät vor dem Senden von DC3 den Maschinenzustand einstellen (Log-In-Kommando), mit dem der Rechenbetrieb fortgesetzt werden soll.

# 2. Kommandos am externen Bedieninterface

**Kommandoübersicht:**

| Zeichen | Funktion | Erledigung | Antwort |
|---|---|---|---|
| DC1 | Abfrage auf Bedienanforderungen; Direktsteuerung ausschalten | Unterbrechung | Ereignisbyte + ACK oder NAK |
| DC2 | Übergang in den Bedienbetrieb | Grundzustand, Serviceschleife | ACK |
| DC3 | Übergang in den Rechenbetrieb | Grundzustand, Serviceschleife | ACK |
| DC4 | Direktsteuerung einschalten | Unterbrechung | ACK |
| CAN | Die serielle Eingabe initialisieren (Puffer, Kommandoanforderungen und Fehlerbedingungen löschen) | Unterbrechung | Keine |
| EOT | Die serielle Eingabe initialisieren (Puffer und Kommandoanforderungen löschen, aber nicht die Fehlerbedingungen) | Unterbrechung | Keine |
| EM | Im Kommandogerät Programmstart von Anfang an. Hardware-Rücksetzen in allen anderen Einrichtungen | Unterbrechung | Keine |
| ENQ | Maschinenfehlerlogout senden | Unterbrechung | Maschinenfehlerlogout + ACK |
| CR | Kommandos mit Datenübertragung | Grundzustand, Serviceschleife | ACK oder NAK. Zuvor ggf. Lesedaten |

# 3. Allgemeine Kommandos

**DC1:**
Abfrage auf Bedienanforderungen. Antwort ACK oder NAK. ACK = Anforderung vorhanden, NAK = keine Anforderung. Ist eine Anforderung anhängig, sendet das Kommandogerät zunächst das Ereignisbyte (EVENTS) in zwei Hexadezimalzeichen zurück. Darauf folgt ACK. Die Art der Anforderung geht aus der niederwertigen Tetrade (Bits 3...0) des Ereignisbytes (EVENTS) hervor.

Ausschalten der Direktsteuerung. Ist die Direktsteuerung eingeschaltet, so wird sie ausgeschaltet. Daraufhin werden alle externen Kommandos vom Kommandogerät ausgewertet. Die Direktsteuerung wird dann ausgeschaltet, wenn das des interne Bedieninterface frei ist, also das zuletzt gesendete Byte die USART verlassen hat. Das Kommandogerät sendet danach sofort ACK. Das Ereignisbyte wird nicht gesendet.

**DC2:**
Übergang in den Bedienbetrieb (vom Bediengerät angefordert). Setzt das Ereignis EXTERNAL_REQUEST. Der Übergang wird in der Grundsteuerschleife ausgelöst. Daraufhin wird die Antwort ACK gesendet.

**DC3:**
Übergang in den Rechenbetrieb (vom externen Bediengerät angefordert). Der Übergang wird in der Grundsteuerschleife ausgelöst. Antwort ACK. Das Bediengerät muß – falls erforderlich – zuvor die Einrichtungen (Slaves) in den Rechenbetrieb umschalten (Direktsteuerung mit Kommando DC3).

**CAN:**
Die seriellen Schnittstellen (extern und intern) initialisieren. Die seriellen Puffer, Kommandoanforderungen und Fehlerbedingungen löschen. Antwort: keine. Anwendung: Herstellen eines Ausgangszustands, Wiederanlaufversuche.

**EOT:**
Die externe serielle Schnittstelle initialisieren. Den seriellen Puffer und die Kommandoanforderungen löschen, aber nicht die Fehlerbedingungen. Antwort: keine. Anwendung: Herstellen eines Ausgangszustands, vor allem zur Fehleranalyse.

**EM:**
Das Kommandogerät startet die Programmausführung von Anfang an (Software-Rücksetzen). Das Kommando wirkt nur, wenn es im seriellen Puffer eine besondere Zeichenfolge vorfindet (Sicherheitskennung). Antwort: keine. Die erste Sicherheitskennung: m – s – t – r.

**ENQ:**
Das Kommandogeräts sendet einen fest formatierten Maschinenfehlerreport (Logout). Jedes Byte wird mit zwei Hexadezimalzeichen übertragen. Das erste Byte gibt den Typ der Einrichtung an. Die Übertragung wird mit ACK abgeschlossen. *Hinweis:* Diese Funktion wird deshalb über ein Sonderzeichen ausgelöst, damit der eigentliche Empfangspuffer erhalten bleibt und als Teil des Reports zurückgesendet werden kann.

**CR:**
Kommandos mit Datenübertragung. Das Kommandogerät antwortet mit ACK (Bestätigung) oder mit NAK (Abweisung oder Fehleranzeige). Typische Fehler, die zum Senden von NAK führen:

- ungültiger Kommandocode,
- falscher Füllstand des seriellen Puffers (zuviele oder zuwenige Bytes).

# 4. Kommandos zum Abfragen und Steuern

**P – CR: Poll**
Die Anforderungen werden abgefragt. Antwort: Anforderungsbytes + ACK. Es werden Bytes zurückgesendet, die die Anforderung näher kennzeichnen. Jedes Byte wird mit zwei Hexadezimalzeichen übertragen.

- Das erste Byte ist das Ereignisbyte (EVENTS).
- Ist eine Kommandogeräteanforderung (MC REQ) anhängig, folgt der Anforderungscode nach (REQUEST CODE).
- Ist eine Maschinenfehlerbedingung (MACHINE CHECK) anhängig, folgen vier Fehlerbytes nach: 1. Fehler der seriellen Schnittstellen (USART), 2. Fehler der seriellen Kommandos, Parameter usw. (SERIAL CHECKS), 3. interne Fehler (INTERNAL CHECKS), 4. Fehlercode (ERROR CODE).

Die Schalteranforderung des Kommandogeräts (MC SWITCH) und die ATN-Anforderung (ATN REQUEST) werden nur über die entsprechenden Bits im Ereignisbyte signalisiert; es werden keine weiteren Bytes übertragen.

**q – Anforderungsnummer – CR: Quit**
Die von der Anforderungsnummer (1 HEX-Zeichen) ausgewählte Anforderung wird gelöscht. Antwort: ACK oder NAK, je nach Anforderungsnummer und Zustand.

Die Anforderungsnummern betreffen:

1: Schalteranforderung. Das Kommandogerät sendet NAK zurück, wenn der Schalter noch betätigt ist. Die Anforderung wird erst dann gelöscht, nachdem der Schalter losgelassen wurde.
2: ATN-Anforderung. Ist das ATN-Signal des Kommandointerfaces noch aktiv (Low), wir die Anforderung nicht gelöscht und NAK zurückgesendet.
3: Kommandogeräteanforderung + Anforderungscode (REQUEST CODE).
4: Maschinenfehlerbedingung + die vier Fehlerbytes.
F: Alles zuvor Genannte.

**X – Kommandocode – Parameter – CR: Ein Steuerkommando ausführen (Execute)**
Die Zeichenkette aus Kommandocode und ( falls erforderlich) Parametern beschreibt ein Kommando, das im Kommandogerät selbst ausgeführt oder über das interne Bedieninterface ausgelöst wird. Die Anzahl der Parameter hängt vom Kommando ab. Manche Kommandos werden immer ausgeführt, manche nur unter bestimmten Bedingungen der Slaveauswahl. Kommandos, die das Zurücksenden von Daten bewirken, werden nur dann ausgeführt, wenn ein einziger Slave ausgewählt ist. Bei erfolgreicher Ausführung wird ACK gesendet, ansonsten NAK.

**X-Kommandos:**

| Kommando | Wirkung |
|---|---|
| X – RST | Auslösen eines Hardwarerücksetzens in den anderen Einrichtungen. ACK kommt, nachdem der Rücksetzimpuls wieder inaktiv geworden ist. |
| X – X | Starten eines beliebigen Ablaufs, der über Parameter P0 (Funktionscode) ausgewählt wird. |
| X – x | Starten eines beliebigen Ablaufs. Parameter P0 dient dabei als Startadresse. |
| | |
| | |
| | |
| | |
| | |
| | |

# 5. Kommandos zum Schreiben und Lesen

**Q – Testdaten – CR**: **Zurücksenden der Testdaten (Ping)**
Kommunikation mit dem Kommandogerät. Rücksendung: Alles, was gesendet wurde einschließlich des Kommandozeichens (Q). Abschluß mit ACK.

**W – L – x – CR**
Die Anforderungs-LED des Kommandogeräts schalten. x ist ein Steuerzeichen. Die zulässigen Steuerzeichen und deren Wirkungen:

- o = aus
- r = rot
- g = grün
- R = rot blinken
- G = grün blinken
- F = rot/grün blinken (schnell)

**W – H – x – CR**
Die Halt-Run-LED schalten. x ist ein Steuerzeichen. Die zulässigen Steuerzeichen und deren Wirkungen:

- o = aus
- r = rot
- g = grün
- R = rot blinken
- G = grün blinken
- F = rot/grün blinken (schnell)

**W – X – Adresse – CR – Schreibdaten (HEX) – CR**
Schreiben hexadezimal auf Adresse. Die Adresse ist 16 Bits lang. Sie wird mit vier HEX-Zeichen übertragen. Nach dem ersten CR antwortet die Einrichtung mit ACK. Dann folgen die hexadezimalen Schreibdaten. Zwei aufeinander folgende HEX-Zeichen ergeben ein binäres Byte. Das erste Zeichen enthält die höherwertige Tetrade, das zweite Zeichen die niederwertige. Es ist möglich, in den gesamten RAM zu schreiben. Wird versucht, auf ungültige Adressen zu schreiben, so werden die Daten angenommen. Es wird aber nichts geschrieben. Der Schreibzugriff wird aber mit NAK beendet, und die Fehlerbedingung LIMIT_CHECK wird gesetzt.

**W – x – Adresse – CR – Schreibdaten – CR**
Schreiben Zeichen auf Adresse. Die Adresse ist 16 Bits lang. Sie wird mit vier HEX-Zeichen übertragen. Nach dem ersten CR antwortet die Einrichtung mit ACK. Dann folgen die Schreibdaten. Die Bytes werden direkt (ohne weitere Prüfung) in den Speicher geschrieben. Es ist möglich, in den gesamten RAM zu schreiben. Wird versucht, auf ungültige Adressen zu schreiben, so werden die Daten angenommen. Es wird aber nichts geschrieben. Der Schreibzugriff wird aber mit NAK beendet, und die Fehlerbedingung LIMIT_CHECK wird gesetzt.

**W – Y – Adresse – ein Datenbyte (HEX) – CR**
Ein einzelnes Byte headezimal schreiben. Dem Unterkommandocode folgen vier Hexadezimalzeichen für die Adresse und zwei Hexadezimalzeichen für das Datenbyte.

**W – y – Adresse – ein Datenbyte – CR**
Ein einzelnes Byte schreiben. Dem Unterkommandocode folgen vier Hexadezimalzeichen für die Adresse und das Datenbyte.

**W – Q – Adresse – vier Datenbytes (HEX) – CR**
Vier Bytes hexadezimal schreiben. Dem Unterkommandocode folgen vier Hexadezimalzeichen für die Adresse und acht Hexadezimalzeichen für die vier Datenbytes. Die Datenbytes werden auf aufeinander folgende Adressen geschrieben (Adreßerhöhung jeweils um 1).

**W – q – Adresse – vier Datenbytes – CR**
Vier Bytes schreiben. Dem Unterkommandocode folgen vier Hexadezimalzeichen für die Adresse und vier Datenbytes. Die Datenbytes werden auf aufeinander folgende Adressen geschrieben (Adreßerhöhung jeweils um 1).

**W – O – Objektnummer – CR – Schreibdaten (HEX) – CR**
Schreiben von hexadezimalen Objektdaten. Die Objektnummer ist ein binäres Byte (Wertebereich 0...255). Sie wird mit zwei Hexadezimalzeichen übertragen. Objektadresse und Länge ergeben sich aus der Objekttabelle (Object Reference Table ORT) der Einrichtung. Der Datenbereich des Objekts wird stets von Anfang an überschrieben. Jedes zu schreibende Byte wird mit zwei Hexadezimalzeichen übertragen.

**W – o – Objektnummer – CR – Schreibdaten – CR**
Schreiben von Objektdaten. Die Objektnummer ist ein binäres Byte (Wertebereich 0...255). Sie wird mit zwei Hexadezimalzeichen übertragen. Objektadresse und Länge ergeben sich aus der Objekttabelle (Object Reference Table ORT) der Einrichtung. Der Datenbereich des Objekts wird stets von Anfang an überschrieben. Die Bytes werden direkt (ohne weitere Prüfung) in den Speicher geschrieben.

**W – r – Objektnummer – Objektdeskriptor – CR**
Schreiben Objektdeskriptor. Die Objektnummer ist ein binäres Byte (Wertebereich 0...255). Sie wird mit zwei Hexadezimalzeichen übertragen. Der Objektdeskriptor steht in der Objekttabelle (Object Reference Table ORT) der Einrichtung. Er besteht aus fünf Bytes, die mit zehn Hexadezimalzeichen übertragen werden.

**R – X – Adresse – Byteanzahl – CR**
Lesen hexadezimal von Adresse. Es werden so viele Bytes gelesen, wie als Byteanzahl angegeben. Adresse und Byteanzahl sind jeweils 16 Bits lang. Sie werden mit jeweils vier HEX-Zeichen übertragen. Jedes gelesene Datenbyte wird mit zwei Hexadezimalzeichen zurückgesendet.

**R – x – Adresse – Byteanzahl – CR**
Lesen von Adresse. Es werden so viele Bytes gelesen, wie als Byteanzahl angegeben. Adresse und Byteanzahl sind jeweils 16 Bits lang. Sie werden mit jeweils vier HEX-Zeichen übertragen. Die gelesenen Datenbytes werden direkt zurückgesendet.

**R – Y – Adresse – CR**
Ein einzelnes Byte hexadezimal lesen. Dem Unterkommandocode folgen vier Hexadezimalzeichen für die Adresse. Das gelesene Datenbyte wird mit zwei Hexadezimalzeichen zurückgesendet.

**R – y – Adresse – CR**
Ein einzelnes Byte lesen. Dem Unterkommandocode folgen vier Hexadezimalzeichen für die Adresse. Das gelesene Datenbyte wird direkt zurückgesendet.

**R – Q – Adresse – CR**
Vier aufeinander folgende Bytes hexadezimal lesen. Dem Unterkommandocode folgen vier Hexadezimalzeichen für die Adresse. Die gelesenen Datenbytes werden mit acht Hexadezimalzeichen zurückgesendet.

**R – q – Adresse – CR**
Vier aufeinander folgende Bytes lesen. Dem Unterkommandocode folgen vier Hexadezimalzeichen für die Adresse. Die gelesenen Datenbytes werden direkt zurückgesendet.

**R – O – Objektnummer – CR**
Einen kompletten Objektinhalt hexadezimal lesen. Die Objektnummer ist ein binäres Byte (Wertebereich 0...255). Sie wird mit zwei Hexadezimalzeichen übertragen. Objektadresse und Länge ergeben sich aus der Objekttabelle (Object Reference Table ORT) der Einrichtung. Jedes gelesene Datenbyte wird mit zwei Hexadezimalzeichen zurückgesendet.

**R – o – Objektnummer – CR**
Einen kompletten Objektinhalt lesen. Die Objektnummer ist ein binäres Byte (Wertebereich 0...255). Sie wird mit zwei Hexadezimalzeichen übertragen. Objektadresse und Länge ergeben sich aus der Objekttabelle (Object Reference Table ORT) der Einrichtung. Die gelesenen Datenbytes werden direkt zurückgesendet. Achtung: Nur Textobjekte können so gelesen werden.

**R – r – Objektnummer – CR**
Lesen Objektdeskriptor. Die Objektnummer ist ein binäres Byte (Wertebereich 0...255). Sie wird mit zwei Hexadezimalzeichen übertragen. Der Objektdeskriptor steht in der Objekttabelle (Object Reference Table ORT) der Einrichtung. Er besteht aus fünf Bytes, die mit zehn Hexadezimalzeichen übertragen werden.

**X – Auswahlzeichen – Schreibdaten – CR: Parameter schreiben**
Es können ein ausgewählter Parameter oder alle Parameter geschrieben werden.

- Auswahlzeichen = 0...F (hexadezimal): Parameter 0...15 (ein einzelner Parameter).
- Auswahlzeichen = t: Alle Parameter. Das Kommandogerät unterstützt alle 16 möglichen Parameter.

Die Schreibdaten sind Hexadezimalzeichen. Jedes Byte wird mit zwei Hexadezimalzeichen übertragen. Die Anzahl der Bytes je Parameter entspricht der vorgegebenen Parameterlänge. Das niedrigstwertige Byte kommt jeweils als erstes.

**x – Auswahlzeichen – Schreibdaten – CR: Variable schreiben**
Es können eine ausgewählte Variable oder mehrere Variablen geschrieben werden.

- Auswahlzeichen = 0...F (hexadezimal): Variablen 0...15 (eine einzelne Variable).
- Auswahlzeichen = t: Alle Variablen. Das Kommandogerät unterstützt alle 16 möglichen Variablen.

Die Schreibdaten sind Hexadezimalzeichen. Jedes Byte wird mit zwei Hexadezimalzeichen übertragen. Die Anzahl der Bytes je Variable entspricht der vorgegebenen Variablenlänge. Das niedrigstwertige Byte kommt jeweils als erstes.

**Y – Auswahlzeichen – CR: Parameter lesen**
Es können ein ausgewählter Parameter oder alle Parameter gelesen werden.

- Auswahlzeichen = 0...F (hexadezimal): Parameter 0...15 (ein einzelner Parameter).
- Auswahlzeichen = t: Alle Parameter. Das Kommandogerät unterstützt alle 16 möglichen Parameter.

Die Lesedaten sind Hexadezimalzeichen. Jedes Byte wird mit zwei Hexadezimalzeichen übertragen. Die Anzahl der Bytes je Parameter entspricht der vorgegebenen Parameterlänge. Das niedrigstwertige Byte kommt jeweils als erstes.

**y – Auswahlzeichen – CR: Variable lesen**
Es können eine ausgewählte Variable oder mehrere Variablen gelesen werden.

- Auswahlzeichen = 0...F (hexadezimal): Variablen 0...15 (eine einzelne Variable).
- Auswahlzeichen = t: Alle Variablen. Das Kommandogerät unterstützt alle 16 möglichen Variablen.

Die Lesedaten sind Hexadezimalzeichen. Jedes Byte wird mit zwei Hexadezimalzeichen übertragen. Die Anzahl der Bytes je Variable entspricht der vorgegebenen Variablenlänge. Das niedrigstwertige Byte kommt jeweils als erstes.

**Z – Auswahlzeichen – Schreibdaten – CR: Register schreiben (selektiver Log-In)**
Das Auswahlzeichen wählt ein Register bzw. eine Informationsstruktur der Einrichtung aus, unabhängig von der internen Adresse. Alle Ziffern und Buchstaben des ASCII-Codes sind als Auswahlzeichen zugelassen. Aus dieser Auswahl ergibt sich, wieviele Bytes Schreibdaten nachfolgen müssen. Jedes Byte wird mit zwei Hexadezimalzeichen übertragen. Hinweis: Ein Schreiben in Bereiche, die länger sein können als 255 Bytes, wird in dieser Version nicht unterstützt[2].

**z – Auswahlzeichen – CR: Register lesen (selektiver Log-Out)**
Das Auswahlzeichen wählt ein Register bzw. eine Informationsstruktur der Einrichtung aus, unabhängig von der internen Adresse. Alle Ziffern und Buchstaben des ASCII-Codes sind als Auswahlzeichen zugelassen. Aus dieser Auswahl ergibt sich auch, wieviele Bytes Lesedaten zurückgesendet werden. Jedes Byte wird mit zwei Hexadezimalzeichen übertragen.

**Auswahlzeichen:**

| Zeichen | Bezeichnung | Erläuterung | Hinweise |
|---|---|---|---|
| 0 | EVENTS | Ereignisregister | |
| 1 | REQUEST CODE | Die Ursache der ATN-Anforderung | |
| 2 | USART CHECKS | Die Ursache der Stopanforderung | |
| 3 | SERIAL CHECKS | Fehlerbits der seriellen Übertragung | |
| 4 | INTERNAL CHECKS | Interne Fehlerbits | |
| 5 | ERROR CODE | Fehlercode. Ergänzt manche internen Fehlerbits | |
| 6 | COMMON STATS | Allgemeine Zustandsbits | |
| 7 | MACHINE STATE | Das aktuelle Kommando am Kommandointerface | |
| 8 | EXTERNAL COMMAND CODE | Das aktuelle Kommando am externen Bedieninterface | |
| 9 | SLAVE REPLY | Das aktuelle Anwortbyte vom Slave | |
| A | SLAVE ADRS | Die aktuelle Slaveadresse | |
| B | SERBUFF SAVED | Der gerettete Füllstand des seriellen Puffers (externes Bedieninterface) | |
| C | SLAVEBUFF SAVED | Der gerettete Füllstand des seriellen Puffers (internes Bedieninterface) | |
| D | SERBUFFCOUNT | Der aktuelle Füllstand des seriellen Puffers (externes Bedieninterface) | |
| E | SLAVEBUFFCOUNT | Der aktuelle Füllstand des seriellen Puffers (internes Bedieninterface) | |
| F | LEDSTATE | Die Zustandsbits der LED-Anzeige | |
| G | POINTER | Adreßzeiger in Schreibkommandos | 2 Bytes |
| H | BYTECOUNT | Bytezähler in Schreibkommandos | 2 Bytes |

2: Abhilfe: mit Funktion z die jeweilige Adresse beschaffen und das Schreiben mit W-Kommandos erledigen.

| Zeichen | Bezeichnung | Erläuterung | Hinweise |
|---|---|---|---|
| O | OBJECT TABLE | Objekttabelle | Festwerte no_of_objects * descriptor_length. Kein Schreiben |
| S | STACK AREA | Stackbereich | Festwert stacksize. Kein Schreiben |
| s | SP | Stackpointer | 2 Bytes. Kein Schreiben |
| z | MACHINE IDENT | Maschinenidentifikation | 5 Bytes. Kein Schreiben |

# Die Maschinenidentifikation:

| Byte | Bereich | HEX | Angabe | Name im Programm |
|---|---|---|---|---|
| 1 | 00H...FFH | 2 | Typ der Einrichtung | device_type |
| 3 | 15 | 1 | Parameteranzahl | no_of_parms |
| 4 | 15 | 1 | Variablenanzahl* | no_of_vars |
| 5 | 15 | 1 | Anzahl der Eingangsvariablen* | no_of_ins |
| 6 | 255 | 2 | Anzahl der Objekte | no_of_objects |
| 8 | 255 | 2 | Länge des Maschinenfehlerreports | mc_reportlength |
| 10 | 0...255*4 | 2 | Größe des seriellen Puffers | serbuffsize |
| 12 | 255 | 2 | Wortlänge am Kommandointerface (in Bits) | wordlength |
| 14 | 15 | 1 | Variablenlänge (in Bytes)* | varlength |
| 15 | 15 | 1 | Parameterlänge (in Bytes) | paramlength |
| 16 | 15 | 1 | Deskriptorlänge (in Bytes) | descriptor_length |
| 17 | 0...64k-1 | 4 | die Anfangsadresse der Objekttabelle | object_table |
| 21 | 0...255*4 | 2 | Länge des Stackbereichs | stacksize |
| 23 | 255 | 2 | die Anfangsadresse des RAM | sram_start |
| 25 | 0...64k-1 | 4 | die zugängliche RAM-Speicherkapazität außer dem Stackbereich | ram_arena |

*: Beim Kommandogerät bedeutungslos. Es wird der Wert Null zurückgeliefert.

# 6. Register im Kommandogerät

EVENTS

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EXTERNAL REQUEST | INTERNAL REPLY | | | MACHINE CHECK | DVC ATN | MC REQEST | MC SWITCH |

- MC SW: Schalter betätigt (Schalteranforderung).
- MC REQUEST: Bedienanforderung vom Kommandogerät. Die genaue Ursache steht im Byte REQUEST CODE.
- DVC ATN: ATN-Anforderung von den Einrichtungen.
- MACHINE CHECK. Fehlerbedingung. Näheres in den Fehlerregistern OPCOM CHECKS, INTERNAL CHECKS und ERROR CODE.
- INTERNAL REPLY: Vom ausgewählten Slave ist eine Antwort eingetroffen. Das aktuelle Kommando weiter bearbeiten bzw. zu Ende bringen.
- EXTERNAL REQUEST: Über das externe Bedieninterface wurde ein Kommando übertragen. Ausführen.

REQUEST CODE

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

FFH = Anfangszustand (Initialisierung erforderlich).

USART CHECKS

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | REPLY CHECK | INTERNAL FRAMING ERROR | INTERNAL OVERRUN | | | EXTERNAL FRAMING ERROR | EXTERNAL OVERRUN |

Bits 1, 0: Fehler am externen Bedieninterface.
Bits 6, 5, 4: Fehler am internen Bedieninterface.

- OVERRUN: neues Byte empfangen, aber das vorherige noch nicht abgeholt (Fehlermeldung vom USART).
- FRAMING ERROR: falsches Stopbit (Fehlermeldung vom USART).
- REPLY CHECK: In der Antwort des Slaves ist ein USART-Fehler aufgetreten.

SERIAL CHECKS

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | CR CHECK | DATA CHECK | CMD CHECK | BUFFER EMPTY | BUFFER FULL | BUFFER EMPTY | BUFFER FULL |

Bits 1, 0: Fehler am externen Bedieninterface.
Bits 5, 4: Fehler am internen Bedieninterface.

- BUFFER FULL: CR-Kommando und serieller Puffer voll (Überlauf).
- BUFFER EMPTY: CR-Kommando und serieller Puffer leer.
- CMD CHECK: inkorrekter (primärer) Kommandocode (Kommandos mit CR).
- DATA_CHECK: inkorrekter Füllstand des seriellen Puffers (zu viele oder zu wenige Zeichen).
- CR CHECK: Fehler beim Ausführen eines CR-Kommandos (falsche Codes, falsche Werte usw.).

INTERNAL CHECKS

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| LIMIT CHECK | ADRS CHECK | | | RDY TIMEOUT | COMMAND TIMEOUT CHECK | | SPURIOUS ITRP |

- SPURIOUS ITRP: Unerwartete Unterbrechung.

- RDY Timeout: Zeitkontrollfehler am Kommandointerface.
- ADRS CHECK: Adressierungsfehler beim Schreiben oder Lesen.
- LIMIT CHECK: Bereichsüberschreitung beim Schreiben.

ERROR CODE

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ERROR CODE | | | | | | | |

COMMON STATS

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SERVICE MODE | | ENABLE RDY EDGE DETECT | DIRECT CONTROL | | CHARAC-TER MODE | DATA WRITE | BYTEBUFF FULL |

- BYTEBUFF FULL: Der Bytepuffer zur Wandlung der einlaufenden Daten ist mit einem Hexadezimalzeichen gefüllt.
- DATA WRITE: die einlaufenden Daten werden ins Binäre gewandelt und nicht in den seriellen Puffer, sondern direkt in den Speicher geschrieben.
- CHARACTER MODE: direkte Datenzugriffe (nicht hexadazimal).
- DIRECT CONTROL: Direktsteuerung. Das Kommandogerät wertet nur DC1 unbd CD4 selbst aus. Alle anderen Zeichen werden zum internen Bedieninterface weitergeleitet.
- ENABLE RDY EDGE DETECT: Die Flankenerkennung des RDY-Signals wird aktiviert. Das ist die normale Betriebsweise. Sie muß nach dem Rücksetzen vom Bediengerät aktiviert werden. Ist das Bit nicht gesetzt, wird das RDY-Signal nach Ablauf einer Karenzzeit statisch abgefragt (keine Flankenerkennung). Sonderbetriebsart zu Diagnosezwecken usw.
- SERVICE MODE: Die Einrichtung läuft in der Serviceschleife.

LED STATE

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | FAST BLINK | GREEN BLINK | RED BLINK | | FAST BLINK | GREEN BLINK | RED BLINK |
| RUN/HALT LED | | | | ATN LED | | | |

Steuerbytes und Zählwerte im Kommandogerät:

| Bezeichnung | Offset | Bytes | Inhalt |
|---|---|---|---|
| EVENTS | 0 | 1 | Ereignisse |
| REQUEST CODE | 1 | 1 | Die aktuelle externe Anforderung an das Bediengerät (binär codiert) |
| USART CHECKS | 2 | 1 | Fehlerbits der seriellen Schnittstellen |
| SERIAL CHECKS | 3 | 1 | Fehlerbits der seriellen Kommandoausführung |
| INTERNAL CHECKS | 4 | 1 | Interne Fehlerbis |
| ERROR CODE | 5 | 1 | Näheres zur Fehlerursache (binär codiert) |
| COMMON STATS | 6 | 1 | Interne Zustandsbits |
| MACHINE STATE | 7 | 1 | Der Maschinenzustand der Kommandoschleife |
| EXTERNAL COMMAND CODE | 8 | 1 | Das aktuelle externe Kommando vom Bediengerät |
| SLAVE REPLY | 9 | 1 | Die aktuelle Antwort vom Slave |
| DEVICE ADRS | 10 | 1 | Die aktuelle Slaveadresse |
| SERBUFF SAVED | 11 | 1 | Rettung SERBUFFCOUNT (zwecks Fehleranalyse) |
| SLAVEBUFF SAVED | 12 | 1 | Rettung SLAVEBUFFCOUNT (zwecks Fehleranalyse) |
| SERBUFFCOUNT | 13 | 1 | Füllstand des seriellen Puffers |
| SLAVEBUFFCOUNT | 14 | 1 | Füllstand des seriellen Slave-Pufers |
| LEDSTATE | 15 | 1 | Die Zustände der programmseitig angesteuerten LEDs |
| LEDCOUNT ATN | 16 | 2 | Blinkzeitzähler der Anforderungs-LED |
| LEDCOUNT HR | 18 | 2 | Blinkzeitzähler der Halt-Run-LED |
| BYTEBUFF | 20 | 1 | Puffer für das erste Hexadezimalzeichen beim direkten Schreiben |
| POINTER | 21 | 2 | Adreßzeiger beim Schreiben und Lesen |
| BYTECOUNT | 23 | 2 | Bytezähler beim Schreiben und Lesen |
| MAXLENGTH | 25 | 2 | Bereichslänge beim Schreiben und Lesen |

**Parameter im Kommandogerät:**

| Parameter | Bezeichnung | Inhalt |
|---|---|---|
| P0 | CALCULATION MODE | Steuerung des Rechenablaufs |
| P1 | CYCLE LIMIT | Anzahl der Rechenschritte |
| P2 | LOOP LIMIT | Anzahl der Rechendurchläufe |
| P3 | | |
| P4 | | |
| P5 | | |
| P6 | | |
| P7 | | |
| P8 | | |
| P9 | | |
| PA | | |
| PB | | |
| PC | | |
| PD | | |
| PE | | |
| PF | | |

CALCULATION MODE

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| COUNT | STOP | | | | LOOP | SINGLE CYCLE | SINGLE STEP |

- SINGLE STEP: Nach jedem Rechenschritt anhalten.
- SINGLE CYCLE: Nach jedem Rechendurchlauf anhalten.
- LOOP: nach einer vorgegebenen Anzahl von Rechendurchläufen anhalten (Parameter P2 (LOOP LIMIT)).
- STOP: Das STOP#-Signal von den Einrichtungen bestimmt das Ende des Rechendurchlaufs.
- COUNT: Das Ende des Rechendurchlaufs wird durch Zählen der Rechenzyklen bestimmt. Endwert = Variable CYCLE_LIMIT.

Ist weder STOP noch COUNT noch LOOP gesetzt, ergibt sich eine endlose Folge von Rechendurchläufen. STOP# wird ignoriert, nur ATN# wird beachtet.

**Variable im Kommandogerät:**

| Variable | Bezeichnung | Inhalt |
|---|---|---|
| V0 | CYCLE COUNT | der aktuelle Rechenschritt (Zählwert) |
| V1 | LOOP COUNT | der aktuelle Rechendurchlauf (Zählwert) |
| V2 | | |
| V3 | | |
| V4 | | |
| V5 | | |
| V6 | | |
| V7 | | |
| V8 | | |
| V9 | | |
| VA | | |
| VB | | |
| VC | | |
| VD | | |
| VE | | |
| VF | | |

**Anforderungscodes:**

| REQUEST CODE | Ursache |
|---|---|
| 00H | – |
| 01H | Einzelzyklusbetrieb. Rechenzyklus abgelaufen |
| 02H | Einzelschritt ausgeführt |
| 03H | Die vorgegebene Anzahl an Rechendurchläufen ausgeführt |
| | |
| | |
| | |
| | |
| | |
| | |
| FFH | Anfangszustand (Initialisierung erforderlich) |

**Fehlercodes:**

| ERROR CODE | Bezeichnung | Erläuterung |
|---|---|---|
| 1 | rdy edge timeout check | RDY Zeitüberschreitung Vorderflanke |
| 2 | rdy wait timeout check 1 | RDY Zeitüberschreitung Besetztzustand im Einflankenbetrieb |
| 3 | rdy wait timeout check 2 | RDY Zeitüberschreitung Besetztzustand im Zweiflankenbetrieb |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |