MC-1 Protokolle

Referenz der Kommunikationsprotokolle zu Steuerungen der Typen MC100 und MC200

Stand dieses Dokuments: 09.11.1999 11:51

Fehler! Formatvorlage nicht definiert.

Referenz der Kommunikationsprotokolle zu **Steuerungen der Typen MC100 und MC200**

Waldweg 55, 88690 Uhldingen, Deutschland

Telefon +49-7556-9218-0, Telefax +49-7556-9218-50

E-Mail: technik@microdesign.de

http://www.microdesign.de

We like to move it!

Inhaltsverzeichnis

Kapitel 1 Einführung

Diese Dokumentation wendet sich an fortgeschrittene Software-Entwickler, die eine Anbindung der MC100 oder MC200 Steuerungen an sonstige Systeme realisieren möchten.

n Welche Protokolle gibt es?

Grundsätzlich sind zwei verschiedene Arten von Protokollen verfügbar:

- das ASCII-Protokoll zur einfachen Kommunikation über ein Standard-Terminalprogramm und
- das Binär-Protokoll zum schnellen, vollständigen Zugriff auf die Steuerung.

Das ASCII-Protokoll ist standardmäßig nur in Steuerungen der MC100 Familie integriert. Für eine Anbindung an andere Steuerungen muß also stets das Binärprotokoll verwendet werden, bei MC100 haben Sie die Wahl zwischen Binär- oder ASCII-Protokoll.

n Wann benötige ich das Binärprotokoll?

Den direkten Eingriff auf der Binärprotokollebene sollten Sie aufgrund der Komplexität der Materie nur dann vornehmen, wenn es tatsächlich notwendig ist. Für alle Anwendungen, die auf einer der folgenden Hardwareplattformen und einem der erwähnten Betriebssysteme realisiert werden, sind von MICRO DESIGN komplette Treiberbibliotheken erhältlich. Die direkte Verwendung des Binärprotokolls ist in diesem Zusammenhang nicht empfohlen:

Betriebssystem	MC100 Steuerungen	MC200 Steuerungen
MS-DOS ab 3.11	Treiber verfügbar als DLL oder C-Quellcode	Kein Treiber verfügbar, MC100 Treiber kann begrenzt eingesetzt werden.
Windows 3.11	Verwenden Sie den MS-DOS Treiber	Verwenden Sie den MS-DOS Treiber
Windows 95/Windows 98	Nativer Treiber verfügbar (VMC Application Layer DLL/Gerätetreiber)	Nativer Treiber verfügbar (VMC Application Layer DLL/Gerätetreiber)
Windows NT 4/Windows NT 5	Nativer Treiber verfügbar (VMC Application Layer DLL/Gerätetreiber)	Nativer Treiber verfügbar (VMC Application Layer DLL/Gerätetreiber)

n Tabelle 1 – Übersicht Treiberbibliotheken

n Welche Steuerungen sind kompatibel?

Alle CPU-Module der Baureihen MC100 und MC200 arbeiten mit einem grundsätzlich identischem Protokoll, dem MC-1 Binärprotokoll. Dieses Protokoll bietet eine sehr schnelle Kommunikation bei gleichzeitig größtmöglicher Datensicherheit.

Das MC-1 Binärprotokoll ist abwärtskompatibel, d.h. eine MC200 Steuerung kommt problemlos mit dem für MC100 Steuerungen gültigen Protokolldefinitionen klar. Andersherum gilt dies natürlich nicht, denn die Protokollvariante, die bei MC200 Steuerungen zum Einsatz kommt, ist wesentlich erweitert und ergänzt worden. Eine MC100 Steuerung wird also nicht alle Prokotokollerweiterungen verstehen, die mit der MC200 eingeführt wurden.

1.1 Über diese Dokumentation

Die hier vorliegende Dokumentation ist logisch in folgende Kapitel gegliedert:

Kapitel 1 - Einführung

Dieses Kapitel lesen Sie gerade. Es gibt Ihnen einen kurzen Überblick über die grundlegene Funktion der Protokolls sowie Hinweise zur den Hardwarevoraussetzungen. Außerdem finden Sie hier wichtige Hinweise, wie Sie am besten mit dem Handbuch umgehen, damit Sie die gewünschten Information auch schnell auffinden können.

Kapitel 2 – Protokolldefinition

Damit Sie selbst auf der Binärprotokollebene arbeiten können, müssen Sie natürlich einiges über den grundlegenden Aufbau des MC-1 Protokolls wissen. Dieses Kapitel beschäftigt sich genau damit: wie wird ein Telegramm codiert? Wie erfolgt die Checksummensicherung? Wann bekomme ich eine Antwort von der Steuerung? Lesen Sie dieses Kapitel auf jeden Fall aufmerksam, bevor Sie mit der Implementation des Binärprotkolls in Ihr System beginnen.

Kapitel 3 – Telegrammreferenz

Das wohl wichtigste Kapitel dieser Dokumentation ist die Telegrammreferenz. Alle innerhalb des MC-1 Protokolls unterstützten Telegramme werden hier detailliert beschrieben. Bitte beachten Sie, daß bei jedem Telegramm die Information enthalten ist, für welche Steuerungen der jeweilige Befehl zur Verfügung steht.

Kapitel 4 – Speicherlayout

Bei einigen Telegrammen ist es notwendig, daß Sie sich mit der internen Speicherstruktur der Steuerung auskennen. Entsprechende Informationen finden Sie in Kapitel 4.

Kapitel 5 – How to...

In diesem Kapitel geht es in die Praxis! Hier wird beschrieben, wie Sie konkrete Aufgaben – z.B. das Lesen oder Schreiben von Merkern – mit dem Binärprotokoll realisieren können. Zudem finden sich in diesem Kapitel auch Beispiele zur Programmierung in C.

n Struktur und Nummerierung

Zur besseren Übersicht haben wir darauf verzichtet, jede Überschrift mit einer Kapitelnummer zu versehen. Lediglich die wichtigsten Abschnitte sind mit einer Kapitelnummer gekennzeichnet.

n Formatierung in dieser Dokumentation

Damit Sie sich in dieser Dokumentation schnell zurechtfinden können, werden spezielle Informationen stets durch eine besondere Formatierung gekennzeichnet. Wenn Sie sich mit dieser Formatierung vertraut machen, werden Sie sich wesentlich einfacher innerhalb dieser Dokumentation zurechtfinden können.

Wichtige Hinweise

Besonders wichtige Informationen, werden stets durch **Fettdruck** und eine Kennzeichnung am linken Rand hervorgehoben, z.B.:

Bitte probieren Sie das nicht zuhause!

1.2 Hardware und Verdrahtung

n Schnittstelleneinstellung

Die Steuerung kann an jede beliebige RS232-Schnittstelle, die vom PC unterstützt wird, angeschlossen werden. Folgende Anschlußeinstellungen sind notwendig:

Bezeichnung	MC100	MC200
Baudrate	9600 Baud	9600 – 115200 Baud
Parity	Keine	Keine
Datenbits	8	8
Stopbits	1	1

n Tabelle 2 - Schnittstellenparameter

n Kabel MC100

Als Verbindungskabel wird ein mindestens 3-adriges Kabel benötigt. Die Leitungen für RxD und TxD sind hierbei zu kreuzen. Mindestens die Ground-Leitung muß zudem durchverbunden werden.

n Kabel MC200

Als Verbindungskabel ist ein normales 9-adriges, 1:1 durchverbundenes Kabel benötigt. Es müssen alle 9 Leitungen angeschlossen sein.

n Vorbereitung der Schnittstelle (nur MC200)

Die MC200 Familie verfügt auf dem Netzteilmodul über eine galvanisch getrennte Schnittstelle, um elektrischen Schäden am PC zu vermeiden. Aus diesem Grund erfolgt die Spannungsversorgung der RS232-Treiber auf dem Netzteilmodul durch die Handshakeleitungen des PC, die entsprechend gesetzt werden müssen.

Bevor eine Kommunikation mit der Steuerung aufgebaut wird, stellen Sie sicher, daß

- DTR auf Dauer-High und
- RTS auf Dauer-Low liegt.

Wenn diese Voraussetzungen nicht erfüllt sind, kann keine Kommunikation erfolgen, da dann die RS232-Treiber der Steuerung nicht mit Spannung versorgt werden!

n Raum für Ihre Notizen

Kapitel 2 Protokolldefinition

In diesem Kapitel finden Sie die Definition des MC-1 Protokollaufbaus. Das Kapitel 2 ist unterteilt in zwei wesentliche Abschnitte:

n Binärprotokoll

Das Binärprotokoll ist für alle Steuerungen der MC100 und MC200 Familie verfügbar, wenngleich mit einigen Unterschieden zwischen den Protokollvarianten. Es bietet die umfassenste Möglichkeit einer Anbindung an andere Geräte, ist jedoch aufgrund seiner Komplexität nicht unbedingt einfach zu implementieren. Hier wird u.a. besprochen,

- wie ein Sendetelegramm aussehen muß (Kapitel 2.1),
- wie Empfangstelegramme aussehen können (Kapitel 2.2),
- wie Telegrammfehler behandelt werden (Kapitel 2.3),
- welche Protokollvarianten gegenwärtig existieren (Kapitel 2.4) und
- welche speziellen Maßnahmen bei einer Anbindung an MC100 System notwendig sind (Kapitel 2.5).

Bitte lesen Sie die Informationen in diesem Kapitel aufmerksam, bevor Sie mit einer Einbindung des MC-1 Binärprotokolls in Ihr System beginnen.

n ASCII Protokoll

Dieses Protokoll, welches ausschließlich für die MC100 Familie verfügbar ist, erlaubt das einfache Abfragen von Basis-Informationen der Steuerung. Hier wird beschrieben, wie Daten über die Schnittstelle gesendet werden, und welche Daten abgefragt werden können (Kapitel 2.6).

2.1 Das Sendetelegramm

Alle MC100/MC200-Protokolle verfügen über das gleiche Basis-Format:

Byte	Bedeutung
01	Opcode
02	Checksumme
03 – n	Nutzdaten

n Tabelle 3 – Protokollaufbau Sendetelegramm

n Der Befehls-Opcode

Über das erste Byte des Telegramms, den Befehls-Opcode, wird entschieden, welche Aktion durch das Telegramm ausgelöst wird. Je nach Opcode können dies Leseoperationen, Schreiboperationen oder spezielle Funktionen sein.

n Die Telegrammlänge

Mit Ausnahme einiger spezieller Telegramme, die nur im Zusammenhang mit den erweiterten Protokollen verfügbar sind (siehe Kapitel 2.4), läßt sich die Gesamtlänge des Telegramms mit Hilfe einer binären Und-Verknüpfung wie folgt aus dem Befehls-Opcode errechnen:

```
Länge = (Befehls-Opcode) & 0x07 + 1
```

n Die Checksumme

Die Checksumme berechnet sich als 8-Bit Addition aus allen Bytes des Telegramms, mit Ausnahme des Checksummenbytes selbst. Angenommen, das Telegramm ist 4 Byte lang, so ergibt sich die Checksumme aus

```
Checksum = Byte1 + Byte3 + Byte4
```

Das zweite Byte des Telegramms wird hierbei ausgelassen, weil es die Checksumme selbst enthält (siehe auch Tabelle 2). Ein Programmbeispiel für die Berechnung der Checksumme finden Sie in Kapitel 4.

n Die Nutzdaten

Die Nutzdaten unterscheiden sich je nach Befehlsopcode und haben demnach eine variable Länge, bzw. können u.U. ganz ausfallen. Näheres entnehmen Sie bitte der Beschreibung der einzelnen Telegramme.

2.2 Das Empfangstelegramm

Auf jedes Telegramm wird von der Steuerung ein Antworttelegramm geschickt. Je nach Typ des Sendetelegramms kann es sich hierbei um die gewünschten Daten oder lediglich um eine Empfangsbestätigung handeln.

Bitte beachten Sie: Jedes Sendetelegramm gibt automatisch vor, mit welchem Opcode die Empfangsdaten beginnen müssen. Alle empfangenen Daten, die nicht mit genau diesem Opcode beginnen, sind automatisch ungültig.

n Variante 1: Nur Empfangsbestätigung

Erfordert das Sendetelegramm keine detaillierte Antwort, so sendet die Steuerung lediglich ein Acknowledge zurück. Acknowledge ist definiert als 80h (dezimal 128).

n Variante 2: Antworttelegramm

Das Antworttelegramm ist exakt gleich aufgebaut wie das unter 2.1 beschriebende Sendetelegramm:

Byte	Bedeutung
01	Opcode
02	Checksumme
03 – n	Nutzdaten

n Tabelle 4 - Protokollaufbau Antworttelegramm

Für eine Beschreibung der einzelnen Daten in diesem Telegramm vergleichen Sie bitte Kapitel 2.1.

2.3 Fehlerbehandlung

Grundsätzlich sind folgende Situationen als fehlerhafte Übertragungen zu verstehen:

- Es war nicht möglich, alle Bytes des Sendetelegramms über die Schnittstelle zu übertragen.
- Die Steuerung antwortete überhaupt nicht.
- Die Steuerung antwortete mit einem "Not Acknowledge" (Code 87h = 135 dezimal).
- Die Steuerung antwortete mit unerwarteten Daten.

In diesen Fällen wird empfohlen, das Telegramm einige Male zu wiederholen. Falls dies nicht zum Erfolg führt, sollten Sie die Kommunikation mit Fehlermeldung abbrechen bzw. unterbrechen.

n Wie stelle ich fest, daß nicht alle Daten übertragen wurden?

Dies ist abhängig von der Hardware-Plattform, mit der Sie die Daten zur Steuerung übertragen. Die meisten System bieten die Möglichkeit, den Erfolg der Datenübertragung über die serielle Schnittstelle mit einer Funktion oder durch direkten Eingriff an den Hardware-Chip festzustellen. Falls das von Ihnen verwendete System diese Möglichkeit nicht bietet, können Sie schlicht immer davon ausgehen, daß die Daten erfolgreich gesendet wurden, und lediglich die Antwort der Steuerung als Fehlerbedingung prüfen.

n Wann antwortet die Steuerung überhaupt nicht?

Die Aussage „die Steuerung antwortet überhaupt nicht" müssen wir relativieren, bzw. etwas präziser formulieren, denn es gibt zwei mögliche Situationen:

- Die Steuerung antwortet tatsächlich überhaupt nicht, oder
- die Steuerung antwortet nicht innerhalb der festgelegten Timeouts (Zeitgrenzen).

Im zweiten Fall ist vermutlich ein Telegramm nicht vollständig übertragen worden, d.h. z.B. das die Steuerung auf 5 Byte Daten wartet, aber nur 4 Byte erhalten hat. Dann wartet die Steuerung für eine festgelegte Zeit auf das fehlende Byte, bevor Sie ein NAK sendet. Innerhalb dieser Zeit könnte jedoch bereits Ihr Timeout abgelaufen sein, und deshalb erscheint es Ihnen so, als würde die Steuerung überhaupt nicht antworten. Erhöhen Sie dann einmal testweise Ihre Timeout-Werte, um das tatsächliche Verhalten der Steuerung zu ergründen.

Falls die Steuerung tatsächlich nicht antwortet, dann prüfen Sie bitte sorgfältig, ob die Steuerung eingeschaltet ist und normal läuft und ob alle Hardware- und Systemvoraussetzungen – wie in Kapitel 1 beschrieben – erfüllt sind. Für MC100 Steuerungen beachten Sie auch Kapitel 2.5.

n Wann sendet die Steuerung ein NAK?

Die Steuerung sendet immer dann ein "Not Acknowledge" (Code 87h = 135 dezimal), wenn

- ein ungültiger Opcode gesendet wurde,
- eine ungültige Checksumme gesendet wurde,
- ein Telegramm eine ungültige Länge hatte,
- ein Telegramm nicht vollständig empfangen werden konnte oder
- ein Telegramm ungültige Parameter hatte.

n Wann können unerwartete Daten eintreffen, und was ist das?

Da mit jedem Sendetelegramm bereits festgelegt wird, wie der Opcode der Antwortdaten aussehen muß, sind alle Empfangsdaten, die nicht mit dem richtigen Opcode eingehen, automatisch unerwartet und damit ungültig. Prüfen Sie in diesem Fall unbedingt die Baudrate und die Schnittstelleneinstellung.

2.4 Verfügbare Protokollvarianten

Das MC-1 Protokoll wird natürlich ständig weiterentwickelt, um den stets neuen Anforderungen gerecht zu werden. Gegenwärtig ist das MC-1 Protokoll in vier verschiedenen Varianten erhältlich:

n MC100 Protokoll V1

Der „Urvater" des MC-1 Protokolls bietet die volle Funktionalität, aber nur einen begrenzten Befehlssatz. Dieses Protokoll kommt in allen älteren MC100MA Karten, mit der MC100PC Karte sowie bei allen MC100 Karten, die eine Schnittstellensyncronisation benötigen (vgl. Kapitel 2.5) zum Einsatz.

n MC100 Protokoll V2

Das erneuerte MC-1 Protokoll für MC100MA Karten bietet zusätzlich zum Protokoll V1 einige Befehle für den direkten Speicherzugriff. Es kommt in MC100MA Karten mit einem Betriebssystem ab der Version 5.26 zum Einsatz, wenn die entsprechende Karte über einen Dallas-Prozessor mit zwei seriellen Schnittstellen verfügt, sprich: wenn die Karte keine Syncronisation der Schnittstelle benötigt.

n MC200 Protokoll V1

Dies ist die erste MC-1 Protokollvariante, die zusammen mit der MC200 eingesetzt wurde – allerdings nur für kurze Zeit. Denn lediglich Betriebssystemversion der MC200CPU bis 1.30 bzw. des MC200PROFI bis Version 1.11 nutzten dieses Protokoll. Alle MC200 Module mit einem neueren Versionsstand verwenden das MC200 Protokoll V2.

n MC200 Protokoll V2

Dies stellt die modernste Variante des MC-1 Protokolls dar; sie bietet eine Anzahl erweiterter Funktionen, so wie z.B. die Möglichkeit der Achsparametrierung oder des Blocktransfers. Wichtigste Neuerung des V2 Protokolls sind jedoch sehr mächtige Zugriffe auf den internen Speicher der Steuerung, die viele Aktionen beschleunigen, wenn auch komplizieren.

2.5 Syncronisation mit MC100MA

Einige Modelle der MC100 Steuerungen verfügen nur über eine serielle Schnittstelle, und schalten diese Schnittstelle periodisch zwischen RS232 (PC-Kommunikation) und RS485 (interne Buskommunikation) um. Dabei handelt es sich um folgende Versionen bzw. Modellreihen:

- Alle MC100MA Karten mit Standardprozessor (nicht Dallas-CPU)
- Alle MC100MA Karten mit integriertem Indexer

Falls Sie sich nicht sicher sind, ob Ihre MC100MA Karte mit oder ohne Schnittstellenumschaltung arbeitet, verwenden Sie bitte die VMC System Information (enthalten im VMC Workbench Paket), um dies festzustellen. Die entsprechende Information erhalten Sie im Unterpunkt „Geräte-MC100-Eigenschaften":

- Dual Fifo bedeutet: keine Schnittstellenumschalten
- Single Fifo mit 0xff-Syncronisation bedeutet: mit Schnittstellenumschaltung

Wenn Sie über ein Modell mit Schnittstellenumschaltung verfügen, müssen Sie jede Kommunikation mit der MC100MA Karte über das Syncronzeichen syncronisieren. Ansonsten können Sie dieses Kapitel überspringen: eine spezielle Syncronisation ist dann nicht erforderlich.

n Ablauf der Syncronisation

- Die MC100MA Karte sendet in definierten Zeitabschnitten (je nach Betriebssystemversion entweder 5ms oder 10ms) als Syncronsignal das Byte 0xff (hex), also 255 dezimal. Da dieser Code als Telegramm-Opcode nicht gültig ist, kann die Gegenstelle davon ausgehen, daß ein empfangenes 0xff (hex) Byte stets das Syncronsignal ist.
- Die Gegenstelle hat jetzt für einen definierten Zeitraum (bei Betriebssystemversionen mit 5ms Syncronisationsintervall: 2ms, bei Versionen mit 10ms Intervall: 4ms) die Gelegenheit, ein Telegramm an die Steuerung zu senden. Falls es der Gegenstelle nicht gelingt, innerhalb dieses Zeitraums ein gültiges Telegramm die Steuerung zu senden, wird die Schnittstelle wieder auf RS485 umgeschaltet. Daten, die jetzt noch an die Steuerung gesendet werden, gehen unweigerlich verloren.
- Trifft innerhalb der o.b. Wartezeit ein gültiges Telegramm ein, wird die Schnittstellenumschaltung von der Steuerung so lange verzögert, bis die Kommunikation für dieses Telegramm beendet wurde.
- Nach Abschluß eines Telegramms – gleichgültig, ob erfolgreich oder nicht – darf nicht automatisch das nächste Telegramm an die Steuerung gesendet werden, weil diese nach jedem Telegramm stets die Schnittstelle umschaltet. Sie müssen dann also erneut auf ein Syncronzeichen warten, bevor weitere Telegramme geschrieben werden dürfen.

n Mögliche Fehlerquellen

Der Ablauf der Syncronisation ist durchaus knifflig, und macht die Programmierung des Protokolls nicht gerade einfacher. Falls möglich, probieren Sie Ihre grundsätzliche Protokollimplementation erst einmal mit einer Steuerung ohne Schnittstellenumschaltung aus, bevor Sie das Syncronisationsprotokoll implementieren. Typische Probleme bei der Syncronisation sind:

- Die Gegenstelle antwortet nicht innerhalb des festgelegten Zeitrasters (2ms bzw. 4ms)
- Das Syncronisationssignal, daß die Gegenstelle erkennt, ist nicht das zuletzt gesendete, sondern lag noch im Empfangspuffer der Gegenstelle. Achten Sie darauf, daß Sie vor jeder Abfrage des Syncronsignals den Empfangspuffer der seriellen Schnittstelle leeren!

2.6 ASCII-Protokoll

Um die MC100 Steuerung im ASCII-Protokoll zu betreiben, benötigen Sie lediglich ein Standard-Terminalprogramm: das ASCII-Protokoll versteht Klartexteingaben. Achten Sie beim Öffnen der Schnittstelle darauf, daß keinerlei Flußkontrolle (X-On/X-Off, Hardware-Handshake) aktiviert ist.

n Variablen lesen/schreiben

Zum Lesen einer Variable geben Sie ein:

```
Vxxx=? [Enter-Taste]
```

wobei hier „xxx" für die Nummer der Variable steht. Die Steuerung sendet den Wert der Variable als Text, gefolgt von CR (0x0d) zurück. Zum Schreiben einer Variable geben Sie ein:

```
Vxxx=yyy [Enter-Taste]
```

wobei hier „xxx" für die Nummer der Variable, und „yyy" für den neuen Wert steht. Bei erfolgreicher Ausführung sendet die Steuerung „OK", gefolgt von CR (0x0d) zurück.

n Merker lesen/schreiben

Zum Lesen eines Merkers geben Sie ein:

```
Mxxx=? [Enter-Taste]
```

wobei hier „xxx" für die Nummer des Merkers steht. Die Steuerung sendet „1", wenn der Merker eingeschaltet ist, oder „0", wenn er ausgeschaltet ist. Die Antwort wird durch ein CR (0x0d) abgeschlossen. Zum Schreiben eines Merkers geben Sie ein:

```
Mxxx=y [Enter-Taste]
```

wobei hier „xxx" für die Nummer des Merkers, und „y" für den neuen Wert steht. Ein Wert von „0" bedeutet hierbei „Ausschalten", ein Wert von „1" dementsprechend „Einschalten". Bei erfolgreicher Ausführung sendet die Steuerung „OK", gefolgt von CR (0x0d) zurück.

n Ausgang lesen/schreiben

Zum Lesen eines Ausgangs geben Sie ein:

```
Axxx=? [Enter-Taste]
```

wobei hier „xxx" für die Nummer des Ausgangs steht. Die Steuerung sendet „1", wenn der Ausgang eingeschaltet ist, oder „0", wenn er ausgeschaltet ist. Die Antwort wird durch ein CR (0x0d) abgeschlossen. Zum Schreiben eines Ausgangs geben Sie ein:

```
Axxx=y [Enter-Taste]
```

wobei hier „xxx" für die Nummer des Ausgangs, und „y" für den neuen Wert steht. Ein Wert von „0" bedeutet hierbei „Ausschalten", ein Wert von „1" dementsprechend „Einschalten". Bei erfolgreicher Ausführung sendet die Steuerung „OK", gefolgt von CR (0x0d) zurück.

n Eingang lesen/schreiben

Zum Lesen eines Eingangs geben Sie ein:

```
Exxx=? [Enter-Taste]
```

wobei hier „xxx" für die Nummer des Eingangs steht. Die Steuerung sendet „1", wenn der Eingang eingeschaltet ist, oder „0", wenn er ausgeschaltet ist.

n Raum für Ihre Notizen

Kapitel 3 Telegramme

Im Folgenden findet sich eine Übersicht aller Telegramme innerhalb des MC-1 Protokolls:

Befehl	Opcode Senden	Opcode Antwort	Funktion
Alle MC-1 Protokolle			
CMD_RDVAR	0xC3	0x04 (MC100) 0x05 (MC200)	Liest eine SPS-Variable aus der Steuerung
CMD_WRVAR	0xCE (MC100) 0xCF (MC200)	0x80	Schreibt den Inhalt einer SPS-Variable in die Steuerung
CMD_RDBYTE_INT	0x22	0x02	Liest den Inhalt eines Bytes im internen RAM (benötigt für interne Merker, Eingänge und Ausgänge)
CMD_WRBYTE_INT	0x2B	0x80	Schreibt den Inhalt eines Bytes im internen RAM
CMD_WRBIT_INT	0x3B	0x80	Schreibt den Inhalt eines Bits im internen RAM (benötigt für interne Merker, Eingänge und Ausgänge)
CMD_RDBYTE_EXT	0xA3	0x02	Liest den Inhalt eines Bytes im externen RAM (benötigt für externe Merker)
CMD_WRBYTE_EXT	0xAC	0x80	Schreibt den Inhalt eines Bytes im externen RAM
CMD_WRBIT_EXT	0xBB	0x80	Schreibt den Inhalt eines Bits im externen RAM (benötigt für externe Merker)
Nur MC100 Protokoll V2			
CMD_RDBLOCK	0xD5	0xD7	Liest Daten aus dem Speicher der MC100 Steuerung
Nur MC200 Protokoll V1 und MC200 Protokoll V2			
CMD_RDSTAT	0xE3	0xE5	Liest einen Achs- oder Systemparameter
CMD_WRSTAT	0xEF	0x80	Schreibt einen Achs- oder Systemparameter
CMD_RDTABLE	0xE5	0xE3	Liest die logische Adresstabelle
CMD_RDMEM6	0xD5	0xD7	Liest 6 Byte Daten aus dem Speicher
Nur MC200 Protokoll V2			
CMD_RDMEM	0xD6	Variabel	Liest Speicherinhalte variabler Länge
CMD_WRMEM	0xDE	0x80	Schreibt Speicherinhalte variabler Länge
CMD_SYS_DATA0	0x92	Variabel	Systembefehl ohne zusätzliche Daten
CMD_SYS_DATA1	0x93	Variabel	Systembefehl mit einem Byte Daten
CMD_SYS_DATA4	0x96	Variabel	Systembefehl mit vier Byte Daten
CMD_AXIS_NODATA	0x9b	0x80	Achsbefehl ohne zusätzliche Daten
CMD_AXIS_DATA	0x9f	0x80	Achsbefehl mit vier Byte Daten

n Tabelle 5 – MC-1 Telegramme

3.1 CMD_RDVAR

Ü MC100 (Protokoll V1) Ü MC100 (Protokoll V2) Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Lesen einer SPS-Variable aus der Steuerung. Als Parameter wird die Nummer der zu lesenden Variable übergeben. Bitte beachten Sie: die Steuerung kontrolliert nicht, ob die angegebene Variable eine gültige Variable ist, sondern liefert stets einen Wert zurück. Fragen Sie deshalb nur gültige Variablen ab.

Bitte beachten Sie, daß SPS-Variablen bei MC100 und MC200 unterschiedliche Längen haben: Variablen in der MC100 sind 3 Byte (= 24 Bit), in der MC200 4 Byte (= 32 Bit) lang. Deshalb wird zwar sowohl für MC100 als auch MC200 das gleiche Sendetelegramm, aber ein unterschiedliches Antworttelegramm verwendet.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_RDVAR: 0xC3
2	Checksumme (siehe 2.1)
3	Nummer der Variable (untere 8 Bit)
4	Nummer der Variable (obere 8 Bit)

n Tabelle 6 - Sendetelegramm CMD_RDVAR

n Antworttelegramm MC100

Byte	Inhalt
1	Opcode ANS_RDVAR24: 0x04
2	Checksumme (siehe Kapitel 2.1)
3	Variablenwert (Bits 0-7)
4	Variablenwert (Bits 8-15)
5	Variablenwert (Bits 16-23)

n Tabelle 7 - Antworttelegramm CMD_RDVAR mit MC100

n Antworttelegramm MC200

Byte	Inhalt
1	Opcode ANS_RDVAR32: 0x05
2	Checksumme (siehe Kapitel 2.1)
3	Variablenwert (Bits 0-7)
4	Variablenwert (Bits 8-15)
5	Variablenwert (Bits 16-23)
6	Variablenwert (Bits 24-31)

n Tabelle 8 - Antworttelegramm CMD_RDVAR mit MC200

3.2 CMD_WRVAR

Ü MC100 (Protokoll V1) Ü MC100 (Protokoll V2) Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Schreiben einer SPS-Variable. Alle SPS-Variablen sind bei der MC200 4 Byte (= 32 Bit), bei der MC100 3 Byte (= 24 Bit) lang. Seitens der Steuerung erfolgt keine Kontrolle, ob es sich um eine gültige SPS-Variable handelt.

n Sendetelegramm (MC100)

Byte	Inhalt
1	Opcode CMD_RDVAR: 0xCE
2	Checksumme (siehe 2.1)
3	Nummer der Variable (untere 8 Bit)
4	Nummer der Variable (obere 8 Bit)
5	Neuer Variablenwert (Bits 0-7)
6	Neuer Variablenwert (Bits 8-15)
7	Neuer Variablenwert (Bits 16-23)

n Tabelle 9 – Sendetelegramm CMD_WRVAR (MC100)

n Sendetelegramm (MC200)

Byte	Inhalt
1	Opcode CMD_RDVAR: 0xCF
2	Checksumme (siehe 2.1)
3	Nummer der Variable (untere 8 Bit)
4	Nummer der Variable (obere 8 Bit)
5	Neuer Variablenwert (Bits 0-7)
6	Neuer Variablenwert (Bits 8-15)
7	Neuer Variablenwert (Bits 16-23)
8	Neuer Variablenwert (Bits 24-31)

n Tabelle 10 – Sendetelegramm CMD_WRVAR (MC200)

n Antworttelegramm

Byte	Inhalt
1	Acknowledge: 0x80

n Tabelle 11 – Antworttelegramm CMD_WRVAR

3.3 CMD_RDBYTE_INT

Ü MC100 (Protokoll V1) Ü MC100 (Protokoll V2) Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Lesen eines Bytes im internen RAM der Steuerung. Das Lesen der Zustände von internen Merkern, Ein- und Ausgängen wird über diese Funktion abgewickelt. Bei Verwendung eines V2 Protokolls wird jedoch empfohlen, stattdessen einen Befehl für den direkten Speicherzugriff zu wählen.

Entsprechende Tabellen für die Zuordnung der internen Resourcen finden Sie in Kapitel 4.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_RDBYTE_INT: 0x22
2	Checksumme (siehe 2.1)
3	Byteadresse

n Tabelle 12 – Sendetelegramm CMD_RDBYTE_INT

n Antworttelegramm

Byte	Inhalt
1	Opcode ANS_RDBYTE_INT: 0x02
2	Checksumme (siehe 2.1)
3	Binärer Wert der Speicherzelle

n Tabelle 13 - Antworttelegramm CMD_RDBYTE_INT

3.4 CMD_RDBIT_INT

Ü MC100 (Protokoll V1) Ü MC100 (Protokoll V2) Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Lesen eines Bits im internen RAM der Steuerung. Dieser Befehl ist grundsätzlich eine Implementation des CMD_RDBYTE_INT Kommandos, nur das zusätzlich eine Bitmaske übergeben gibt.

Da wir nicht gewährleisten können, daß dieser Befehl auch in zukünftigen Versionen des MC-1 Protokolls noch implementiert sein wird, verwenden Sie bitte zum Lesen interner Daten ausschließlich den Befehl CMD_RDBYTE_INT.

3.5 CMD_WRBIT_INT

Ü MC100 (Protokoll V1) Ü MC100 (Protokoll V2) Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Schreiben eines Bits im internen RAM der Steuerung. Das Schreiben der Zustände von internen Merkern und Ausgängen wird über diese Funktion abgewickelt. Entsprechende Zuordnungstabellen finden Sie in Kapitel 4.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_WRBIT_INT: 0x3B
2	Checksumme (siehe 2.1)
3	Byteadresse
4	Flag (siehe unten)

n Tabelle 14 – Sendetelegramm CMD_WRBIT_INT

Auswertung des Flags

- Soll das entsprechende Bit gesetzt werden, ist der anzugebende Wert 2 ^ Bitnummer, also z.B. 01h für Bit 0, 02h für Bit 1 usw.
- Soll das Bit gelöscht werden, ist der anzugebende Wert NOT (2 ^ Bitnummer), also z.B. FEh für Bit 0, FDh für Bit 1 usw.

n Antworttelegramm

Byte	Inhalt
1	Acknowledge: 0x80

n Tabelle 15 – Antworttelegramm CMD_WRBIT_INT

3.6 CMD_RDBYTE_EXT

Ü MC100 (Protokoll V1) Ü MC100 (Protokoll V2) Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Lesen eines Bytes im externen RAM der Steuerung. Das Lesen der Zustände von externen Merkern wird über diese Funktion abgewickelt. Bei Verwendung eines V2 Protokolls wird jedoch empfohlen, stattdessen einen Befehl für den direkten Speicherzugriff zu wählen.

Entsprechende Tabellen für die Zuordnung der internen Resourcen finden Sie in Kapitel 4.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_RDBYTE_EXT: 0xA3
2	Checksumme (siehe 2.1)
3	Byteadresse
4	Pageadresse

n Tabelle 16 – Sendetelegramm CMD_RDBYTE_EXT

n Antworttelegramm

Byte	Inhalt
1	Opcode ANS_RDBYTE_EXT: 0x02
2	Checksumme (siehe 2.1)
3	Binärer Wert der Speicherzelle

n Tabelle 17 – Antworttelegramm CMD_RDBYTE_EXT

3.7 CMD_RDBIT_EXT

Ü MC100 (Protokoll V1) Ü MC100 (Protokoll V2) Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Lesen eines Bits im externen RAM der Steuerung. Dieser Befehl ist grundsätzlich eine Implementation des CMD_RDBYTE_EXT Kommandos, nur das zusätzlich eine Bitmaske übergeben gibt.

Da wir nicht gewährleisten können, daß dieser Befehl auch in zukünftigen Versionen des MC-1 Protokolls noch implementiert sein wird, verwenden Sie bitte zum Lesen interner Daten ausschließlich den Befehl CMD_RDBYTE_INT.

3.8 CMD_WRBIT_EXT

Ü MC100 (Protokoll V1) Ü MC100 (Protokoll V2) Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Schreiben eines Bits im externen RAM der Steuerung. Ausschließlich das Schreiben der Zustände von externen Merkern wird über diese Funktion abgewickelt, deshalb verwendet die Steuerung auch stets und ausschließlich die Pageadresse der externen Merkerbereiche. Entsprechende Tabellen finden Sie in Kapitel 4.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_WRBIT_EXT: 0xBB
2	Checksumme (siehe 2.1)
3	Byteadresse
4	Flag (siehe unten)

n Tabelle 18 – Sendetelegramm CMD_WRBIT_EXT

Flagauswertung

- Soll das entsprechende Bit gesetzt werden, ist der anzugebende Wert 2 ^ Bitnummer, also z.B. 01h für Bit 0, 02h für Bit 1 usw.
- Soll das Bit gelöscht werden, ist der anzugebende Wert NOT (2 ^ Bitnummer), also z.B. FEh für Bit 0, FDh für Bit 1 usw.

n Antworttelegramm

Byte	Inhalt
1	Acknowledge: 0x80

n Tabelle 19 – Antworttelegramm CMD_WRBIT_EXT

3.9 CMD_RDBLOCK

Ü MC100 (Protokoll V2)

Liest einen Datenblock aus dem Speicher der MC100MA Karte. Es kann sowohl das interne als auch das externe RAM angesprochen werden. Bitte beachten Sie, daß für die Verwendung dieser Funktion eine genaue Kenntnis des Speicherlayouts innerhalb der MC100MA Karte notwendig ist. Es können maximal 56 Byte in einem Telegramm gelesen werden.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_RDBLOCK: 0xD5
2	Checksumme (siehe 2.1)
3	Byteadresse
4	Pageadresse
5	0=internes RAM, 1=externes RAM
6	Länge der zu lesenden Daten (max. 56 Byte)

n Tabelle 20 – Sendetelegramm CMD_RDBLOCK

n Antworttelegramm

Byte	Inhalt
1	Opcode ANS_RDBLOCK: 0xD7
2	Checksumme (siehe 2.1)
3-n	Daten, Länge wie angefordert

n Tabelle 21 – Antworttelegramm CMD_RDBLOCK

3.10 CMD_RDSTAT

Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Lesen eines System- oder Achsparameters. Die Parameter entnehmen Sie bitte der MC-1B Programmierdokumentation oder den Hilfe- und Zusatzdateien der VMC Workbench.

Die Codierung des zu lesenden Parameters erfolgt wie in der MC-1B Programmierdokumentation beschrieben.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_RDSTAT: 0xE3
2	Checksumme (siehe 2.1)
3	Parameternummer, Bits 0-7
4	Parameternummer, Bits 8-15

n Tabelle 22 – Sendetelegramm CMD_RDSTAT

n Antworttelegramm

Byte	Inhalt
1	Opcode ANS_RDSTAT: 0xE5
2	Checksumme (siehe 2.1)
3	Wert des Parameters, Bits 0-7
4	Wert des Parameters, Bits 8-15
5	Wert des Parameters, Bits 16-23
6	Wert des Parameters, Bits 24-31

n Tabelle 23 – Antworttelegramm CMD_RDSTAT

3.11 CMD_WRSTAT

Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Lesen eines System- oder Achsparameters. Die Parameter entnehmen Sie bitte der MC-1B Programmierdokumentation oder den Hilfe- und Zusatzdateien der VMC Workbench.

Die Codierung des zu lesenden Parameters erfolgt wie in der MC-1B Programmierdokumentation beschrieben.

Achtung! Das Beschreiben ungültiger Parameter kann unvorhersehbare Folgen haben. Zudem lösen bestimmte Parameter direkt Funktionen innerhalb der Steuerung aus! Verwenden Sie das Telegramm CMD_WRSTAT mit äußerster Vorsicht!

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_WRSTAT: 0xEF
2	Checksumme (siehe 2.1)
3	Parameternummer, Bits 0-7
4	Parameternummer, Bits 8-15
5	Neuer Wert des Parameters, Bits 0-7
6	Neuer Wert des Parameters, Bits 8-15
7	Neuer Wert des Parameters, Bits 16-23
8	Neuer Wert des Parameters, Bits 24-31

n Tabelle 24 – Sendetelegramm CMD_WRSTAT

n Antworttelegramm

Byte	Inhalt
1	Acknowledge: 0x80

n Tabelle 25 – Antworttelegramm CMD_WRSTAT

3.12 CMD_RDTABLE

Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Mit diesem Befehl kann die logisch-physikalische Adresszuordnungstabelle der angeschlossenen Modulen ausgelesen werden. Dies ermöglicht eine Kontrolle der Zuordnung logischer Modulnummern zu deren physikalischen Codierung.

Grundsätzlich ist dieses Protokoll zum Lesen der Adresszuordnungstabellen für alle unterstützten MC200-Module vorgesehen, gegenwärtig wird jedoch nur das Lesen der Zuordnungstabellen von Achsmodulen unterstützt.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_RDTABLE: 0xE5
2	Checksumme (siehe 2.1)
3	Reserviert, immer 0x00
4	Reserviert, immer 0x00
5	Logische Addresse der zu lesenden Achse
6	Reserviert, immer 0x00

n Tabelle 26 – Sendetelegramm CMD_RDTABLE

n Antworttelegramm

Byte	Inhalt
1	Opcode ANS_RDTABLE: 0xE3
2	Checksumme (siehe 2.1)
3	Physikalische Codierung der Achse (0-7)
4	Interner Typcode des Achsmoduls

n Tabelle 27 – Antworttelegramm CMD_RDTABLE

3.13 CMD_RDMEM6

Ü MC200 (Protokoll V1) Ü MC200 (Protokoll V2)

Lesen eines 6 Byte langen Speicherblocks aus der MC200 Steuerung. Die Speicheradresse wird linear angegeben, die Länge der gelesenen Daten ist konstant auf 6 Byte festgelegt.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_RDMEM6: 0xD5
2	Checksumme (siehe 2.1)
3	Zu lesende Speicheradresse, Bits 0-7
4	Zu lesende Speicheradresse, Bits 8-15

n Tabelle 28 – Sendetelegramm CMD_RDMEM6

n Antworttelegramm

Byte	Inhalt
1	Opcode ANS_RDMEM6: 0xD7
2	Checksumme (siehe 2.1)
3-8	6 Byte Daten

n Tabelle 29 – Antworttelegramm CMD_RDMEM6

3.14 CMD_RDMEM

ü MC200 (Protokoll V2)

Lesen eines Speicherblocks variabler Länge aus der MC200 Steuerung. Die Speicheradresse wird linear angegeben, die Länge der gelesenen Daten darf 125 Byte nicht übersteigen.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_RDMEM: 0xD6
2	Checksumme (siehe 2.1)
3	Zu lesende Speicheradresse, Bits 0-7
4	Zu lesende Speicheradresse, Bits 8-15
5	Länge der zu lesenden Daten (max. 125 Byte)

n Tabelle 30 – Sendetelegramm CMD_RDMEM

n Antworttelegramm

Byte	Inhalt
1	Opcode ANS_RDMEM: 0xD0
2	Checksumme (siehe 2.1)
3-n	Nutzdaten

n Tabelle 31 – Antworttelegramm CMD_RDMEM

3.15 CMD_WRMEM

Ü MC200 (Protokoll V2)

Schreiben eines Speicherblocks variabler Länge in die MC200 Steuerung. Die Speicheradresse wird linear angegeben, die Länge der zu schreibenden Daten darf 121 Byte nicht übersteigen.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_WRMEM: 0xDE
2	Checksumme (siehe 2.1)
3	Zu schreibende Speicheradresse, Bits 0-7
4	Zu schreibende Speicheradresse, Bits 8-15
5	Länge der folgenden Nutzdaten (max. 121 Byte)
6-n	Nutzdaten

n Tabelle 32 – Sendetelegramm CMD_WRMEM

n Antworttelegramm

Byte	Inhalt
1	Acknowledge: 0x80

n Tabelle 33 – Antworttelegramm CMD_WRMEM

3.16 CMD_SYSDATA

Ü MC200 (Protokoll V2)

Auslösen einer internen Steuerungsfunktion. Je nach auszulösendem Befehl werden entweder keine, 1 Byte oder 4 Byte Parameterdaten übertragen. Das zu erwartende Antworttelegramm hängt ebenfalls von dem ausgelösten Befehl aus.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_SYSDATA0: 0x92 Opcode CMD_SYSDATA1: 0x93 Opcode CMD_SYSDATA4: 0x96
2	Checksumme (siehe 2.1)
3	Systemkommando, ggf. von Parameterdaten gefolgt (je nach gewähltem Telegramm)

n Tabelle 34 – Sendetelegramm CMD_SYSDATA

n Systemkommandos

Wert	Bedeutung	Opcode Senden	Opcode Antwort
0x00	Liest den aktuellen SPS Status	0x92	0x94
0x01	Startet das SPS-Programm	0x92	0x80
0x02	Stoppt das SPS-Programm	0x92	0x80
0x03	Führt einen Einzelschritt im SPS-Programm aus	0x92	0x80
0x04	Löscht das SPS-Programm	0x92	0x80
0x05	Führt einen Software-Reset der Steuerung durch Parameterbyte: Bit 0 gesetzt = SPS-Programm nach Reset nicht neu starten. Bit 1 gesetzt = Hardware-Reset mit Urlöschung durchführen	0x93	0x80
0x06	Bereitet die Übertragung eines neuen SPS-Programms vor (aktuelles SPS-Programm wird angehalten, Achsen werden gestoppt, I/O Verarbeitung wird angehalten). Parameterbyte: reserviert, muß immer 0 sein.	0x93	0x80
0x07	Beendet die Übertragung des SPS-Programms in den Speicher der Steuerung. Das System wird jetzt neu gestartet (Software-Reset).	0x92	0x80
0x08	Ermittelt Anfangsadresse des UDB	0x92	0x95
0x09	Löst eine Display-Funktion aus (siehe Tabelle 36). Die vier Byte Parameterdaten haben folgende Funktion: Byte 1: Auswahl des Displays (0 = 1. Display) Byte 2-4: Funktion und Parameter (siehe Tabelle 36)	0x96	0x80
0x0a	Löst Funktionen im Zusammenhang mit dem Flash aus. Als Parameter wird die Funktion übergeben (Tabelle 37).	0x93	0x80

n Tabelle 35 – Gültige Systemkommandos für CMD_SYSDATA0

n Display Funktionscodes

Funktion	Bedeutung	Parameter 1	Parameter 2
0x01	Display-Buffer aus der Steuerung ans Display übertragen (auffrischen)	Speicheroffset, ab dem übertragen werden soll (erstes Zeichen im Display = 0)	Länge der zu übertrangenden Daten
0x02	Display-Buffer aus der Steuerung ans Display übertragen (auffrischen) und in großer Schrift darstellen	Speicheroffset, ab dem übertragen werden soll (erstes Zeichen = 0)	Länge der zu übertrangenden Daten
0x03	Funktionen im Zusammenhang mit dem eingebauten Lautsprecher	1 = Tastenklick einschalten 2 = Tastenklick ausschalten 3 = Dauerton ausgeben 4 = Ton 125ms 5 = Ton 250ms 6 = Ton 500ms 7 = Ton 1s 8 = Ton 2s 9 = Dauerton abschalten	Nicht verwendet
0x0b	Gespeichertes Bitmap anzeigen	Nummer des darzustellenden Bitmaps	Nicht verwendet
0x0c	Gespeichertes Bitmap mit dem aktuellen Inhalt des Displays binär verordern und darstellen	Nummer des dazustellenden Bitmaps	Nicht verwendet
0x0d	Display-Seite auswählen	Nummer der darzustellenden Displayseite	Nicht verwendet

n Tabelle 36 – Display Funktionscodes mit CMD_SYSDATA

n Flash Funktionscodes

Wert	Bedeutung
0x01	SPS-Programm im Flash speichern
0x02	SPS-Programm aus dem Flash laden

n Tabelle 37 – Flash Funktionscodes mit CMD_SYSDATA

n Antworttelegramme

Je nach gewählter Funktion unterscheiden sich die Antworttelegramme. Bitte lesen Sie in der Tabelle 35 das entsprechende Antworttelegramm zu jeder Funktion ab, und finden Sie hier die zugehörige Tabelle.

Nur Acknowledge (0x80)

Byte	Inhalt
1	Acknowledge: 0x80

n Tabelle 38 – Antworttelegramme CMD_SYSDATA

ANS_SYS_STATUS (0x94)

Byte	Inhalt
1	Opcode ANS_SYS_STATUS: 0x94
2	Checksumme (siehe Kapitel 2.1)
3	Flags Bit 0 = SPS-Programm vorhanden Bit 1 = SPS läuft Bit 2 = Fehler: Ungültiger Opcode Bit 3 = Fehler: Stack Überlauf Bit 4 = Haltepunkt-Liste wurde gelöscht Bit 5 = Task-Umschaltung deaktiviert Bit 6 = Aktuelles Bitergebnis Bit 7 = SPS-Programm steht auf Haltepunkt
4	Aktueller Programmzähler (Bits 0-7)
5	Aktueller Programmzähler (Bits 8-15)

n Tabelle 39 – Antworttelegramm CMD_SYSDATA auf Statusanfrage

ANS_SYS_UDB (0x95)

Byte	Inhalt
1	Opcode ANS_SYS_UDB: 0x95
2	Checksumme (siehe Kapitel 2.1)
3	UDB Adresse (Bits 0-7)
4	UDB Adresse (Bits 8-15)
5	UDB Adresse (Bits 16-23)
6	UDB Adresse (Bits 24-31)

n Tabelle 40 – Antworttelegramm CMD_SYSDATA auf UDB-Anfrage

3.17 CMD_AXIS_NODATA

Ü MC200 (Protokoll V2)

Löst einen Achsbefehl innerhalb der Steuerung aus. Parameter werden keine übergeben.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_AXIS_NODATA: 0x9b
2	Checksumme (siehe 2.1)
3	Logische Nummer der Achse
4	Achsbefehl (siehe Tabelle 42)

n Tabelle 41 – Sendetelegramm CMD_AXIS_NODATA

n Achsbefehle

Wert	Bedeutung
0x06	Referenzfahrt durchführen
0x07	Start auf vorgeladenen Wert
0x08	Stop mit programmierter Rampe
0x09	Stop mit maximaler Rampe
0x11	Achsfehler quittieren und löschen

n Tabelle 42 – Achsbefehle für CMD_AXIS_NODATA

n Antworttelegramm

Byte	Inhalt
1	Acknowledge: 0x80

n Tabelle 43 – Antworttelegramm CMD_AXIS_NODATA

3.18 CMD_AXIS_DATA

Ü MC200 (Protokoll V2)

Löst einen Achsbefehl innerhalb der Steuerung aus. Hierbei werden 4 Byte Parameterdaten übergeben.

n Sendetelegramm

Byte	Inhalt
1	Opcode CMD_AXIS_NODATA: 0x9f
2	Checksumme (siehe 2.1)
3	Logische Nummer der Achse
4	Achsbefehl (siehe Tabelle 45)
5-8	Parameter

n Tabelle 44 – Sendetelegramm CMD_AXIS_DATA

n Achsbefehle

Wert	Bedeutung	Parameter
0x00	Sonderfunktion auslösen	Nummer der Sonderfunktion
0x01	Relative Position vorladen	Position
0x02	Absolute Position vorladen	Position
0x03	Start auf absolute Position	Position
0x04	Start auf relative Position	Position
0x05	Start kontinuierlich	1 = Plusrichtung, 0 = Minusrichtung
0x0a	Override setzen	Neuer Override-Wert
0x0b	Geschwindigkeitswechsel an Position	Position für Geschwindigkeitswechsel
0x0c	Geschwindigkeit setzen	Geschwindigkeit
0x0d	Beschleunigung (Rampe) setzen	Rampenwert
0x0e	Sofortiger Geschwindigkeitswechsel	Geschwindigkeit
0x10	Leitungsteil ein-/ausschalten	0 = LT ausschalten, 1 = LT einschalten
0x12	Starte absolute interpolierte Bewegung	Position
0x13	Start relative interpolierte Bewegung	Position

n Tabelle 45 – Achsbefehle für CMD_AXIS_DATA

n Antworttelegramm

Byte	Inhalt
1	Acknowledge: 0x80

n Tabelle 46 – Antworttelegramm CMD_AXIS_DATA

n Raum für Ihre Notizen

Kapitel 4 Speicherauslegung

Einige der Funktionen, die im vorhergenden Kapitel beschrieben wurden, müssen eine Speicheradresse innerhalb der Steuerung als Parameter angeben. Da dies unter anderem sehr wichtige Funktionsbereiche, wie z.B. das Lesen und Schreiben von SPS-Merkern, betrifft, finden sich in diesem Kapitel entsprechende Informationen zur Speicherauslegung innerhalb der Steuerung.

n MC100/MC200

Bitte beachten Sie, daß die Steuerungen der Reihe MC100 und MC200 ein grundlegend unterschiedliches Speicherlayout haben. Während in der MC100 streng zwischen internem und externem RAM getrennt wird, und zudem einige Daten aufgrund von Speichermangel recht ungleichmäßig innerhalb des Speichers verteilt sind, ist die MC200 streng linear adressiert. Die Unterscheidung zwischen internem und externem RAM ist hier zwar grundsätzlich auch gegeben, jedoch werden die entsprechenden Telegramme nur aus Kompatibilitätsgründen weiterhin wie bei der MC100 verarbeitet.

Falls Sie eine Anbindung alleine für MC200 Steuerungen entwickeln, könnte es für Sie interessant sein, aus Gründen der besseren Performance ausschließlich über die direkten Speicherlesebefehle zu arbeiten, statt die (kompatiblen) RD_BYTE_INT und RD_BYTE_EXT zu verwenden.

n Warnung: Schreibbefehle

Innerhalb der Steuerung werden keinerlei Gültigkeitsprüfungen durchgeführt, ob es sich bei dem zu schreibenden Wert nun tatsächlich um einen Merker oder Ausgang, oder aber um interne Programmbereiche handelt. Der Hauptgrund hierfür ist, daß die gleichen Telegramme auch von System- und Diagnosewerkzeugen – wie z.B. der VMC Workbench – verwendet werden, um interne Datenbereiche der Steuerung zu verändern.

Dies bedeutet für Sie, daß Sie Schreibbefehle mit äußerster Vorsicht programmieren sollten. Wenn Sie Datenbereiche außerhalb der für Veränderungen zugelassenen Bereiche für Merker und Ausgänge verändern, kann dies zu nicht vorhersagbaren Ergebnissen im Verhalten der Steuerung führen.

4.1 MC100 Steuerungen

Wie bereits erwähnt, wird innerhalb der MC100 streng zwischen internem und externem RAM getrennt. Dies ist nicht weiter kompliziert; nicht ganz so einfach ist hingegen das Lesen und Schreiben von internen Systemmerkern mit den RD_BYTE_INT und WR_BIT_INT Befehlen, denn diese Merkerbereiche sind innerhalb der Steuerung nicht linear adressiert.

n Interne Merker

Um einen internen Merker zu lesen oder zu schreiben, gehen Sie bitte wie folgt vor:

- Suchen Sie den entsprechen Merker in der nachfolgenden Tabelle.
- Lesen Sie mit RD_BYTE_INT die entsprechende Speicherzelle aus.
- Maskieren Sie das Ergebnis mit der angegebene Bitmaske.
- Ist das Ergebnis 0, ist der Merker ausgeschaltet. In allen anderen Fällen ist er eingeschaltet.

Merker	Speicheradresse	Bitmaske
M500	0x2B	0x01
M501	0x2B	0x02
M502	0x2B	0x04
M503	0x2B	0x08
M504	0x2B	0x10
M505	0x2B	0x20
M506	0x2B	0x40
M507	0x2B	0x80
M508	0x21	0x20
M509	0x21	0x40
M510	0x09	0x01
M511	0x2C	0x01
M512	0x2C	0x02
M513	0x2C	0x04
M514	0x2C	0x08
M515	0x2C	0x10
M516	0x2C	0x20
M517	0x2C	0x40
M518	0x2C	0x80
M519	0x65	0x01
M520	0x65	0x02
M521	0x65	0x04
M522	0x65	0x08
M523	0x65	0x10
M524	0x65	0x20
M525	0x65	0x40
M526	0x65	0x80
M527	0x66	0x01
M528	0x66	0x02
M529	0x66	0x04
M530	0x66	0x08
M531	0x66	0x10
M532	0x66	0x20
M533	0x66	0x40
M534	0x66	0x80
M535	0x67	0x01
M536	0x67	0x02
M537	0x67	0x04

Merker	Speicheradresse	Bitmaske
M538	0x67	0x08
M539	0x67	0x10
M540	0x67	0x20
M541	0x67	0x40
M542	0x67	0x80
M550	0x23	0x10
M551	0xe1	0x01
M552	0xe1	0x02
M553	0xe1	0x04
M554	0xe1	0x08
M555	0xe1	0x10
M556	0xe1	0x20
M557	0xe1	0x40
M558	0xe1	0x80
M559	0x23	0x02
M560	0x23	0x04
M561	0x22	0x01
M562	0x22	0x08
M563	0x22	0x10
M564	0x22	0x40
M565	0x22	0x80
M566	0x23	0x01
M570	0x23	0x08
M571	0x23	0x80
M572	0x28	0x40
M573	0x21	0x01
M574	0x20	0x80
M575	0xe0	0x01
M576	0xe0	0x02
M580	0x21	0x02
M588	0x26	0x80
M589	0x26	0x40
M590	0x24	0x10
M591	0x24	0x20
M592	0x24	0x40
M593	0x24	0x80
M594	0x25	0x01
M595	0x25	0x02

Merker	Speicheradresse	Bitmaske
M596	0x25	0x10
M597	0x25	0x20
M598	0x25	0x40
M599	0x25	0x80
M601	0x58	0x01
M602	0x58	0x02
M603	0x58	0x04
M604	0x58	0x08
M605	0x58	0x10
M606	0x58	0x20
M607	0x58	0x40
M608	0x58	0x80
M609	0x59	0x01
M610	0x59	0x02
M611	0x59	0x04
M612	0x59	0x08
M613	0x59	0x10
M614	0x59	0x20
M615	0x59	0x40
M616	0x59	0x80
M617	0x5A	0x01
M618	0x5A	0x02
M619	0x5A	0x04
M620	0x5A	0x08
M621	0x5A	0x10
M622	0x5A	0x20
M623	0x5A	0x40
M624	0x5A	0x80
M625	0x5B	0x01
M626	0x5B	0x02
M627	0x5B	0x04
M628	0x5B	0x08
M629	0x5B	0x10
M630	0x5B	0x20
M631	0x5B	0x40
M632	0x5B	0x80
M633	0x5C	0x01
M634	0x5C	0x02
M635	0x5C	0x04
M636	0x5C	0x08
M637	0x5C	0x10
M638	0x5C	0x20
M639	0x5C	0x40
M640	0x5C	0x80
M641	0x5D	0x01
M642	0x5D	0x02
M643	0x5D	0x04
M644	0x5D	0x08
M645	0x5D	0x10
M646	0x5D	0x20
M647	0x5D	0x40
M648	0x5D	0x80
M649	0x5E	0x01
M650	0x5E	0x02
M651	0x5E	0x04

Merker	Speicheradresse	Bitmaske
M652	0x5E	0x08
M653	0x5E	0x10
M654	0x5E	0x20
M655	0x5E	0x40
M656	0x5E	0x80
M723		
M801	0xB8	0x01
M802	0xB8	0x02
M803	0xB8	0x04
M804	0xB8	0x08
M805	0xB8	0x10
M806	0xB8	0x20
M807	0xB8	0x40
M808	0xB8	0x80
M809	0xB9	0x01
M810	0xB9	0x02
M811	0xB9	0x04
M812	0xB9	0x08
M813	0xB9	0x10
M814	0xB9	0x20
M815	0xB9	0x40
M816	0xB9	0x80
M817	0xBA	0x01
M818	0xBA	0x02
M819	0xBA	0x04
M820	0xBA	0x08
M821	0xBA	0x10
M822	0xBA	0x20
M823	0xBA	0x40
M824	0xBA	0x80
M825	0xBB	0x01
M826	0xBB	0x02
M827	0xBB	0x04
M828	0xBB	0x08
M829	0xBB	0x10
M830	0xBB	0x20
M831	0xBB	0x40
M832	0xBB	0x80
M833	0xBC	0x01
M834	0xBC	0x02
M835	0xBC	0x04
M836	0xBC	0x08
M837	0xBC	0x10
M838	0xBC	0x20
M839	0xBC	0x40
M840	0xBC	0x80
M841	0xBD	0x01
M842	0xBD	0x02
M843	0xBD	0x04
M844	0xBD	0x08
M845	0xBD	0x10
M846	0xBD	0x20
M847	0xBD	0x40
M848	0xBD	0x80
M849	0xBE	0x01

Merker	Speicheradresse	Bitmaske
M850	0xBE	0x02
M851	0xBE	0x04
M852	0xBE	0x08
M853	0xBE	0x10
M854	0xBE	0x20
M855	0xBE	0x40
M856	0xBE	0x80
M857	0xBF	0x01
M858	0xBF	0x02
M859	0xBF	0x04
M860	0xBF	0x08
M861	0xBF	0x10
M862	0xBF	0x20
M863	0xBF	0x40
M864	0xBF	0x80
M865	0xC0	0x01
M866	0xC0	0x02
M867	0xC0	0x04
M868	0xC0	0x08
M869	0xC0	0x10
M870	0xC0	0x20
M871	0xC0	0x40
M872	0xC0	0x80
M873	0xC1	0x01
M874	0xC1	0x02
M875	0xC1	0x04
M876	0xC1	0x08
M877	0xC1	0x10
M878	0xC1	0x20
M879	0xC1	0x40
M880	0xC1	0x80
M881	0xC2	0x01
M882	0xC2	0x02
M883	0xC2	0x04
M884	0xC2	0x08
M885	0xC2	0x10
M886	0xC2	0x20
M887	0xC2	0x40
M888	0xC2	0x80
M889	0xC3	0x01
M890	0xC3	0x02
M891	0xC3	0x04
M892	0xC3	0x08
M893	0xC3	0x10
M894	0xC3	0x20
M895	0xC3	0x40
M896	0xC3	0x80
M897	0xC4	0x01
M898	0xC4	0x02
M899	0xC4	0x04
M900	0xC4	0x08
M901	0xC4	0x10
M902	0xC4	0x20
M903	0xC4	0x40
M904	0xC4	0x80

Merker	Speicheradresse	Bitmaske
M905	0xC5	0x01
M906	0xC5	0x02
M907	0xC5	0x04
M908	0xC5	0x08
M909	0xC5	0x10
M910	0xC5	0x20
M911	0xC5	0x40
M912	0xC5	0x80
M913	0xC6	0x01
M914	0xC6	0x02
M915	0xC6	0x04
M916	0xC6	0x08
M917	0xC6	0x10
M918	0xC6	0x20
M919	0xC6	0x40
M920	0xC6	0x80
M921	0xC7	0x01
M922	0xC7	0x02
M923	0xC7	0x04
M924	0xC7	0x08
M925	0xC7	0x10
M926	0xC7	0x20
M927	0xC7	0x40
M928	0xC7	0x80
M929	0xCC	0x01
M930	0xCC	0x02
M931	0xCC	0x04
M932	0xCC	0x08
M933	0xCC	0x10
M934	0xCC	0x20
M935	0xCC	0x40
M936	0xCC	0x80
M937	0xC8	0x01
M938	0xC8	0x02
M939	0xC8	0x04
M940	0xC8	0x08
M941	0xC8	0x10
M942	0xC8	0x20
M943	0xC8	0x40
M944	0xC8	0x80
M945	0xC9	0x01
M946	0xC9	0x02
M947	0xC9	0x04
M948	0xC9	0x08
M949	0xC9	0x10
M950	0xC9	0x20
M951	0xC9	0x40
M952	0xC9	0x80
M953	0xCD	0x01
M954	0xCD	0x02
M955	0xCD	0x04
M956	0xCD	0x08
M957	0xCD	0x10
M958	0xCD	0x20
M959	0xCD	0x40

Merker	Speicheradresse	Bitmaske
M960	0xCD	0x80
M1001	0x60	0x01
M1002	0x60	0x02
M1003	0x60	0x04
M1004	0x60	0x08
M1005	0x60	0x10
M1006	0x60	0x20
M1007	0x60	0x40
M1008	0x60	0x80
M1009	0x61	0x01
M1010	0x61	0x02
M1011	0x61	0x04
M1012	0x61	0x08
M1013	0x61	0x10
M1014	0x61	0x20
M1015	0x61	0x40
M1016	0x61	0x80
M1017	0x62	0x01
M1018	0x62	0x02
M1019	0x62	0x04
M1020	0x62	0x08
M1021	0x62	0x10
M1022	0x62	0x20
M1023	0x62	0x40
M1024	0x62	0x80
M1025	0xD0	0x01
M1026	0xD0	0x02
M1027	0xD0	0x04
M1028	0xD0	0x08
M1029	0xD0	0x10
M1030	0xD0	0x20
M1031	0xD0	0x40
M1032	0xD0	0x80
M1033	0xD1	0x01
M1034	0xD1	0x02
M1035	0xD1	0x04
M1036	0xD1	0x08
M1037	0xD1	0x10
M1038	0xD1	0x20
M1039	0xD1	0x40
M1040	0xD1	0x80
M1041	0xD2	0x01
M1042	0xD2	0x02
M1043	0xD2	0x04
M1044	0xD2	0x08
M1045	0xD2	0x10
M1046	0xD2	0x20
M1047	0xD2	0x40
M1048	0xD2	0x80
M1049	0xD3	0x01
M1050	0xD3	0x02
M1051	0xD3	0x04
M1052	0xD3	0x08
M1053	0xD3	0x10
M1054	0xD3	0x20
M1055	0xD3	0x40
M1056	0xD3	0x80
M1057	0xD4	0x01
M1058	0xD4	0x02
M1059	0xD4	0x04
M1060	0xD4	0x08
M1061	0xD4	0x10
M1062	0xD4	0x20
M1063	0xD4	0x40
M1064	0xD4	0x80
M1065	0xD5	0x01
M1066	0xD5	0x02
M1067	0xD5	0x04
M1068	0xD5	0x08
M1069	0xD5	0x10
M1070	0xD5	0x20
M1071	0xD5	0x40
M1072	0xD5	0x80
M1073	0xD6	0x01
M1074	0xD6	0x02
M1075	0xD6	0x04
M1076	0xD6	0x08
M1077	0xD6	0x10
M1078	0xD6	0x20
M1079	0xD6	0x40
M1080	0xD6	0x80
M1081	0xD7	0x01
M1082	0xD7	0x02
M1083	0xD7	0x04
M1084	0xD7	0x08
M1085	0xD7	0x10
M1086	0xD7	0x20
M1087	0xD7	0x40
M1088	0xD7	0x80

n Tabelle 47 – Interne Merker MC100

Hinweis: Lücken innerhalb dieser Tabelle bedeuten, daß die entsprechenden Merker tatsächlich nicht vorhanden sind und dementsprechend auch weder gelesen noch geschrieben werden können.

n Externe Merker

Alle externen Merker sind innerhalb des externen RAM der MC100 Steuerung linear abgelegt und können mit dem Befehl CMD_RDBYTE_EXT gelesen bzw. mit dem Befehl CMD_WRBIT_EXT geschrieben werden.

⇒ Erster externer Merker (Bereich 1): 1

⇒ Letzter externer Merker (Bereich 1): 499

⇒ Erster externer Merker (Bereich 2): 2000

⇒ Letzter externer Merker (Bereich 2): 3548

⇒ Speicherseite (Page) im externen RAM: 0x7f

Die Berechnungsformel zum Ermitteln der Anfangsadresse und des Bitoffset lautet wie folgt:

Page = 0x7f
Wenn Merker < 500: Adresse = Int((Merkernummer – 1) / 8)
Wenn Merker > 1999: Adresse = Int((Merkernummer – 1501) / 8)
Bit = 2 ^ (Merkernummer – 1) % 8

n SPS-Ausgänge

Alle digitalen SPS-Ausgänge sind innerhalb des internen RAM der MC100 Steuerung linear abgelegt und können mit dem Befehl CMD_RDBYTE_INT gelesen bzw. mit dem Befehl CMD_WRBIT_INT geschrieben werden.

⇒ Erster digitaler Ausgang: 1

⇒ Letzter digitaler Ausgang: 128

⇒ Speicherseite (Page) im internen RAM: 0x90

Die Berechnungsformel zum Ermitteln der Anfangsadresse und des Bitoffset lautet wie folgt:

Adresse = Int((Ausgangsnummer – 1) / 8) + 0x90
Bit = 2 ^ (Ausgangsnummer – 1) % 8

Ein Beispiel für diese Berechnung finden Sie in Kapitel 5.

n SPS-Eingänge

Alle digitalen SPS-Eingänge sind innerhalb des internen RAM der MC100 Steuerung linear abgelegt und können mit dem Befehl CMD_RDBYTE_INT gelesen werden.

⇒ Erster digitaler Eingang: 1

⇒ Letzter digitaler Eingang: 196

⇒ Speicherseite (Page) im internen RAM: 0xA0

Die Berechnungsformel zum Ermitteln der Anfangsadresse und des Bitoffset lautet wie folgt:

Adresse = Int((Eingangsnummer – 1) / 8) + 0xA0
Bit = 2 ^ (Eingangsnummer – 1) % 8

Ein Beispiel für diese Berechnung finden Sie in Kapitel 5.

n Sondereingänge

Innerhalb der MC100 sind auch einige Sondereingänge definiert, die vom SPS-Programm wie normale Eingänge behandelt werden können, jedoch innerhalb des MC-1 Protokolls wie ein interner Merker abgefragt werden müssen. Um die entsprechen Eingänge (von Eingang 500 bis 579) zu lesen, verwenden Sie bitte – wie auch bei den internen Merkern – die nachfolgende Tabelle.

Eingang	Speicheradresse	Bitmaske
E500	0xB8	0x01
E501	0xB8	0x02
E502	0xB8	0x04
E503	0xB8	0x08
E504	0xB9	0x01
E505	0xB9	0x02
E506	0xB9	0x04
E507	0xB9	0x08
E508	0xBA	0x01
E509	0xBA	0x02
E510	0xBA	0x04
E511	0xBA	0x08
E512	0xBB	0x01
E513	0xBB	0x02
E514	0xBB	0x04
E515	0xBB	0x08
E516	0xBC	0x01
E517	0xBC	0x02
E518	0xBC	0x04
E519	0xBC	0x08
E520	0xBD	0x01
E521	0xBD	0x02
E522	0xBD	0x04
E523	0xBD	0x08
E524	0xBE	0x01
E525	0xBE	0x02
E526	0xBE	0x04
E527	0xBE	0x08
E528	0xBF	0x01
E529	0xBF	0x02
E530	0xBF	0x04
E531	0xBF	0x08
E532	0xC0	0x01
E533	0xC0	0x02
E534	0xC0	0x04
E535	0xC0	0x08
E536	0xC0	0x10
E537	0xC1	0x01
E538	0xC1	0x02
E539	0xC1	0x04
E540	0xC1	0x08
E541	0xC1	0x10
E542	0xC2	0x01
E543	0xC2	0x02
E544	0xC2	0x04
E545	0xC2	0x08
E546	0xC2	0x10
E547	0xC3	0x01
E548	0xC3	0x02
E549	0xC3	0x04
E550	0xC3	0x08
E551	0xC3	0x10
E552	0xC4	0x01
E553	0xC4	0x02
E554	0xC4	0x04
E555	0xC4	0x08
E556	0xC4	0x10
E557	0xC5	0x01
E558	0xC5	0x02
E559	0xC5	0x04
E560	0xC5	0x08
E561	0xC5	0x10
E562	0xC6	0x01
E563	0xC6	0x02
E564	0xC6	0x04
E565	0xC6	0x08
E566	0xC6	0x10
E567	0xC7	0x01
E568	0xC7	0x02
E569	0xC7	0x04
E570	0xC7	0x08
E571	0xC7	0x10
E572	0xC8	0x01
E573	0xC8	0x02
E574	0xC8	0x04
E575	0xC8	0x08
E576	0xC9	0x01
E577	0xC9	0x02
E578	0xC9	0x04
E579	0xC9	0x08

n Tabelle 48 – Sondereingänge MC100

4.2 MC200 Steuerungen

Der Speicher in MC200 Steuerungen in vollständig linear adressiert. Aus diesem Grund können alle Daten anhand einfacher Berechnungen ausgelesen werden.

n Interne Merker

Alle internen Merker sind innerhalb des internen RAM der MC200 Steuerung linear abgelegt und können mit dem Befehl CMD_RDBYTE_INT gelesen bzw. mit dem Befehl CMD_WRBIT_INT geschrieben werden.

⇒ Erster interner Merker: 2049

⇒ Letzter interner Merker: 3548

⇒ Speicherseite (Page) im internen RAM: 0x00

Die Berechnungsformel zum Ermitteln der Anfangsadresse und des Bitoffset lautet wie folgt:

Adresse = Int((Merkernummer – 1) / 8) + 0x00
Bit = 2 ^ (Merkernummer – 1) % 8

Ein Beispiel für diese Berechnung finden Sie in Kapitel 5.

n Externe Merker

Alle externen Merker sind innerhalb des externen RAM der MC200 Steuerung linear abgelegt und können mit dem Befehl CMD_RDBYTE_EXT gelesen bzw. mit dem Befehl CMD_WRBIT_EXT geschrieben werden.

⇒ Erster externer Merker: 1

⇒ Letzter externer Merker: 2048

⇒ Speicherseite (Page) im externen RAM: 0x00

Die Berechnungsformel zum Ermitteln der Anfangsadresse und des Bitoffset lautet wie folgt:

Page = 0x00
Adresse = Int((Merkernummer – 1) / 8)
Bit = 2 ^ (Merkernummer – 1) % 8

n SPS-Ausgänge

Alle digitalen SPS-Ausgänge sind innerhalb des internen RAM der MC200 Steuerung linear abgelegt und können mit dem Befehl CMD_RDBYTE_INT gelesen bzw. mit dem Befehl CMD_WRBIT_INT geschrieben werden.

⇒ Erster digitaler Ausgang: 1

⇒ Letzter digitaler Ausgang: 256

⇒ Speicherseite (Page) im internen RAM: 0xC0

Die Berechnungsformel zum Ermitteln der Anfangsadresse und des Bitoffset lautet wie folgt:

Adresse = Int((Ausgangsnummer – 1) / 8) + 0xC0
Bit = 2 ^ (Ausgangsnummer – 1) % 8

Ein Beispiel für diese Berechnung finden Sie in Kapitel 5.

n SPS-Eingänge

Alle digitalen SPS-Eingänge sind innerhalb des internen RAM der MC200 Steuerung linear abgelegt und können mit dem Befehl CMD_RDBYTE_INT gelesen werden.

⇒ Erster digitaler Eingang: 1

⇒ Letzter digitaler Eingang: 256

⇒ Speicherseite (Page) im internen RAM: 0xE0

Die Berechnungsformel zum Ermitteln der Anfangsadresse und des Bitoffset lautet wie folgt:

Adresse = Int((Eingangsnummer – 1) / 8) + 0xE0
Bit = 2 ^ (Eingangsnummer – 1) % 8

Ein Beispiel für diese Berechnung finden Sie in Kapitel 5.

n Raum für Ihre Notizen

Kapitel 5 How to...

In diesem Kapitel wird das theoretische Wissen aus den vorherigen Kapiteln noch ein wenig präzisiert. Sie finden hier eine genaue Anleitung, wie Sie mit Hilfe der verfügbaren Telegramme sowohl interne als auch externe Merker, Eingänge und Ausgänge lesen und schreiben können.

Die Beispiele in diesem Kapitel beziehen sich stets auf das MC200 System, sind jedoch nicht abhängig von einem der erweiteren Protokolle. Im die Beispiele zusammen mit einem MC100 System zu verwenden, müssen Sie lediglich die verwendeten Speicheradressen durch die in Kapitel 4 aufgeführten, für MC100 gültigen Speicheradressen ersetzen.

In Kapitel 5.5 finden Sie zudem einige Beispiele für die Programmierung dieses Protokolls in C.

5.1 Interne Merker

Dieses Beispiel bezieht sich auf die MC200 Steuerung. Mit einer MC100 Steuerung können interne Merker nur mit Hilfe der in Kapitel 4 aufgeführten Speichertabellen gelesen werden.

Die internen Merker sind byteweise im internen RAM ab der Adresse 00h angeordnet. Der erste gültige interne Merker ist 2049, der letzte ist 3584. Der Byteoffset eines internen Merkers ergibt sich also aus der Formel:

```
ByteOffset = Int((MerkerNummer – 2049) / 8)
```

Der Bitoffset des jeweiligen Merkers ergibt sich aus der Formel

```
BitOffset = (MerkerNummer – 2049) mod 8
```

n Beispiel: Lesen eines internen Merkers

Um z.B. den internen Merker 2711 auszulesen, gehen Sie wie folgt vor:

- Offset auf ersten internen Merker berechnen:

  ```
  MerkerOffset = 2711 – 2049 = 662
  ```

- Byteoffset im internen RAM berechnen:

  ```
  ByteOffset = Int(662 / 8) = 82
  ```

- Lesen des Byte 82 mit dem Befehl CMD_RDBYTE_INT, Ergebnis in Data
- Berechnen des Bitoffsets:

  ```
  BitOffset = 662 mod 8 = 6
  ```

- Isolieren des entsprechenden Bits:

  ```
  Result = (Data AND (2 ^ BitOffset)) <> 0
  ```

Result enthält nun den boolschen Wert, ob der Merker gesetzt ist.

n Beispiel: Schreiben eines internen Merkers

Um z.B. den internen Merker 2327 zu schreiben, gehen Sie wie folgt vor:

- Offset auf ersten internen Merker berechnen:

  ```
  MerkerOffset = 2327 – 2049 = 278
  ```

- Byteoffset im internen RAM berechnen:

  ```
  ByteOffset = Int(278 / 8) = 34
  ```

- Berechnen des Bitoffsets:

  ```
  BitOffset = 278 mod 8 = 6
  ```

- Berechnen der Bitmaske. Wenn der Merker gesetzt werden soll, wird die Bitmaske so errechnet:

  ```
  BitMask = (2 ^ BitOffset) = 40h = 64
  ```

- Soll der Merker zurückgesetzt (gelöscht) werden, ist die Bitmaske zu intertieren:

  ```
  Bitmask = NOT (2 ^ BitOffset) = BFh = 191
  ```

- Schreiben des Byte 34 mit dem Befehl CMD_WRBIT_INT und der errechneten Bitmaske

5.2 Externe Merker

Dieses Beispiel bezieht sich auf die MC200 Steuerung. Mit einer MC100 Steuerung ist die Codierung zwar ähnlich, verwendet aber andere Speicheradressen. Vergleichen Sie hierzu bitte Kapitel 4.

Die externen Merker sind byteweise im externen RAM ab der Adresse 00h bei der Pageadresse 0000h angeordnet. Der erste gültige externe Merker ist 1, der letzte ist 2048. Der Byteoffset eines externen Merkers ergibt sich also aus der Formel:

```
ByteOffset = Int((MerkerNummer – 1) / 8)
```

Der Bitoffset des jeweiligen Merkers ergibt sich aus der Formel

```
BitOffset = (MerkerNummer – 1) mod 8
```

n Beispiel: Lesen eines externen Merkers

Um z.B. den externen Merker 841 auszulesen, gehen Sie wie folgt vor:

- Offset auf ersten externen Merker berechnen:

  ```
  MerkerOffset = 841 – 1 = 840
  ```

- Byteoffset im internen RAM berechnen:

  ```
  ByteOffset = Int(840 / 8) = 105
  ```

- Lesen des Byte 105 von der PageAdresse 00h mit dem Befehl CMD_RDBYTE_EXT, Ergebnis in Data
- Berechnen des Bitoffsets:

  ```
  BitOffset = 840 mod 8 = 0
  ```

- Isolieren des entsprechenden Bits:

  ```
  Result = (Data AND (2 ^ BitOffset)) <> 0
  ```

Result enthält nun den boolschen Wert, ob der Merker gesetzt ist.

5.3 Digitale Eingänge

Dieses Beispiel bezieht sich auf die MC200 Steuerung. Mit einer MC100 Steuerung ist die Codierung zwar ähnlich, verwendet aber andere Speicheradressen. Vergleichen Sie hierzu bitte Kapitel 4.

Die digitalen Eingänge sind byteweise im internen RAM ab der Adresse E0h angeordnet. Der erste gültige digitale Eingang ist 1, der letzte ist 256. Der Byteoffset eines digitalen Eingangs ergibt sich also aus der Formel

```
ByteOffset = Int((EingangsNummer – 1) / 2) + E0h
```

Der Bitoffset des jeweiligen Eingangs ergibt sich aus der Formel

```
BitOffset = (EingangNummer – 2049) mod 8
```

Um z.B. den internen Eingang 122 auszulesen, gehen Sie wie folgt vor:

- Physikalische Eingangsnummer berechnen:

  ```
  Eingang = 122 – 1 = 121
  ```

- Byteoffset im internen RAM berechnen:

  ```
  ByteOffset = Int(121 / 8) + E0h= EFh
  ```

- Lesen des Byte EFh mit dem Befehl CMD_RDBYTE_INT, Ergebnis in Data
- Berechnen des Bitoffsets:

  ```
  BitOffset = 121 mod 8 = 1
  ```

- Isolieren des entsprechenden Bits:

  ```
  Result = (Data AND (2 ^ BitOffset)) <> 0
  ```

Result enthält nun den boolschen Wert, ob der Eingang gesetzt ist.

5.4 Digitale Ausgänge

Dieses Beispiel bezieht sich auf die MC200 Steuerung. Mit einer MC100 Steuerung ist die Codierung zwar ähnlich, verwendet aber andere Speicheradressen. Vergleichen Sie hierzu bitte Kapitel 4.

Die digitalen Ausgänge sind byteweise im internen RAM ab der Adresse C0h angeordnet. Der erste gültige digitale Ausgang ist 1, der letzte ist 256. Der Byteoffset eines digitalen Ausgangs ergibt sich also aus der Formel

```
ByteOffset = Int((AusgangsNummer – 1) / 2) + C0h
```

Der Bitoffset des jeweiligen Ausgangs ergibt sich aus der Formel

```
BitOffset = (AusgangNummer – 2049) mod 8
```

Um z.B. den internen Ausgang 122 auszulesen, gehen Sie wie folgt vor:

- Physikalische Ausgangsnummer berechnen:

  ```
  Ausgang = 122 – 1 = 121
  ```

- Byteoffset im internen RAM berechnen:

  ```
  ByteOffset = Int(121 / 8) + C0h= CFh
  ```

- Lesen des Byte EFh mit dem Befehl CMD_RDBYTE_INT, Ergebnis in Data
- Berechnen des Bitoffsets:

  ```
  BitOffset = 121 mod 8 = 1
  ```

- Isolieren des entsprechenden Bits:

  ```
  Result = (Data AND (2 ^ BitOffset)) <> 0
  ```

Result enthält nun den boolschen Wert, ob der Ausgang gesetzt ist.

5.5 Beispiele in C

Im Folgenden einige kurze Beispiele für die Einbindung in C.

n Checksummenberechnung

Berechnet für das im Pointer "Daten" enthaltene Telegramm der Länge "Length" die Checksumme und gibt diese als unsigned integer zurück.

```
unsigned char CalcChecksum(char *Daten, int Length)
{
          unsigned char Checksum = *Daten;    // Vorbesetzen mit Opcode
          Length -= 2;                        // Opcode und Checksumme überspringen
          Daten += 2;                         // Pointer ebenfalls erhöhen

          while (Length--)                    // für die Länge des Telegramms...
          Checksum += *Daten++;               // Checksumme berechnen

          return (Checksum & 0xff);           // 8-Bit Checksumme zurückgeben
}
```

n Internen Merker lesen

Liest den als "MerkerNum" angebenen internen Merker. Die Funktionen "WriteSerial" und "ReadSerial", die in diesem Beispiel aufgerufen werde, lesen von und schreiben auf die serielle Schnittstelle. In diesem Beispiel erfolgt keine Prüfung, ob es sich um einen gültigen Merker handelt.

```
int GetMarker(int MerkerNummer);
{
          int ByteOffset, BitOffset;
          char SendString[8], ReadString[8];

          Merker -= 2049;             // Erster physikalischer interner Merker
          ByteOffset = Merker/8;      // Byteoffset errechnen
          BitOffset = Merker % 8;     // Bitoffset errechnen

          SendString[0] = 0x22;       // CMD_RDBYTE_INT
          SendString[2] = ByteOffset; // Zu lesendes Byte
          SendString[1] = CalcChecksum(SendString, 3);    // siehe Beispiel oben

          WriteSerial(SendString, 3); // Telegramm abschicken
          ReadSerial(ReadString, 3);  // Antwort ist 3 Byte lang

// An dieser Stelle sollte die Antwort (inkl. Checksumme) geprüft werden

          if ((ReadString[2] & (2 ^ BitOffset)) == 0) return (FALSE);

          return (TRUE);
}
```


n Raum für Ihre Notizen

n Raum für Ihre Notizen

Anhang A Tabellenverzeichnis

n Raum für Ihre Notizen

n Raum für Ihre Notizen

