MICRODESIGN

MCStep/Unipos API

Nutzung des Windows-Programmierschnittstelle zur Kommunikation mit MCStep und Unipos 110 Steuerungen

Übersicht der verwendeten Systemressourcen

Stand dieses Dokuments: Februar 2006

MCStep/Unipos API

Nutzung des Windows-Programmierschnittstelle zur Kommunikation mit MCStep und Unipos 110 Steuerungen

Übersicht der verwendeten Systemressourcen

Copyright © 2000 MICRO DESIGN Industrieelektronik GmbH.
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

Raum für Ihre Notizen

Kapitel 1 Einführung

Diese Dokumentation wendet sich an Software-Entwickler, die ein MCStep, StarStep oder Unipos 110 System von einem PC aus programmieren oder kontrollieren möchten. Das Dokument gliedert sich hierbei in zwei wesentliche Abschnitte:

- **Programmierung mit der MCStep API**
  Für Computer, die unter einem beliebigen Microsoft Windows Betriebssystem arbeiten, dürfte die Verwendung der MCStep API zumeist die einfachste und zugleich komfortabelste Lösung sein. Im ersten Teil der Dokumentation wird die Schnittstelle zu dieser API aus der MCStep.DLL beschrieben,
- **Direkte Programmierung ohne API**
  Wer seine Anwendung unter einem anderen Betriebssystem nutzen möchte oder aus anderen Gründen keine DLL verwenden will oder kann, hat die Möglichkeit direkt auf die verwendeten Systemressourcen zuzugreifen. Diese Variante ist wesentlich komplexer und erfordert ein umfassendes Hintergrundwissen sowie entsprechend erweiterte Kenntnisse in der Software-Entwicklung. Greifen Sie nur dann zu dieser Variante, wenn Sie mit der API definitiv nicht ans Ziel kommen.

Was ist MCStep? Was ist Unipos? Was ist Starstep?

MCStep ist eine Programmieroberfläche für MC100 Steuerungssysteme aus dem Hause MICRODESIGN. Dieses System wird als OEM-Produkt von einer Anzahl von Herstellern unter eigenem Namen vertrieben. Die am meist verbreitenden Varianten sind

- MCStep,
- Unipos 110 (mit der Software UniED) und
- StarStep.

Im Weiteren werden wir in dieser Dokumentation stets die Bezeichnung MCStep verwenden. Sämtliche Hinweise beziehen sich aber stets auf alle kompatiblen Steuerungssystem.

Das MCStep System selbst ist ein Anwendungsprogramm, entwickelt in der MC-1A Sprache für MC100 Systeme. Dieses Anwendungsprogramm überwacht und interpretiert die MCStep Ablaufprogramme als Befehlsinterpreter und Benuzterinterface.

Wie werden MCStep Programme gespeichert?

Das Programm selbst ist in 24-Bit Variablen codiert. Die Codierung der Befehle erfolgt entweder durch das Anwendungsprogramm in der MC100 Steuerung, die PC-Software MCStep Studio oder über die zugrundeliegende MCStep.DLL.

Gibt es weiterführende Literatur?

Die MC-1A Programmierdokumentation geht auf die Programmierung der MC100 Steuerungen in der MC-1A Sprache ein. In dieser Sprache wurde auch das MCStep System enwickelt.

Raum für Ihre Notizen

Kapitel 2 MCStep API

Die MCStep API liegt in Form einer DLL mit frei zugänglichen und dokumentieren Funktionen vor, die Ihnen die einfache und komfortable Nutzung der MCStep Steuerung ermöglicht. Folgende Grundvoraussetzungen müssen jedoch erfüllt sein, damit auf einem Endkundenrechner die API genutzt werden kann:

- Betriebssystem Microsoft Windows 95, Microsoft Windows 98, Microsoft Windows ME, Microsoft Windows NT 4.0, Microsoft Windows 2000, Microsoft Windows XP oder Microsoft Windows Server 2003 und
- eines der Softwarepakete MCStep Developer Studio, UniED/X2 oder VMC Workbench X2 installiert auf dem Zielcomputer.

MCStep API ohne installiertes Softwarepaket

Ohne eines der oben angeführten Softwarepakete kann die MCStep API nicht verwendet werden. Es reicht auch nicht aus, Dateien wie die MCStep.DLL auf den Zielcomputer zu kopieren: während der Installation der oben angeführten Softwarepakete wird eine grössere Menge Treiber, Systemdateien und –dienste sowie Basiseinstellungen auf dem Rechner eingerichtet. Ohne diese funktioniert die MCStep API nicht.

Bitte beachten Sie, dass einige der Softwarepakete, wie z.B. UniED, nicht direkt über MICRODESIGN bezogen werden können. Wenden Sie sich bitte in jedem Fall zunächst an Ihren Systemlieferanten.

2.1 Einbinden der API

Die API kann in jede beliebige Programmiersprache eingebunden werden, die das Nachladen von dynamischen Laufzeitbibliotheken (Dynamic Link Library, kurz DLL) ermöglicht. Für die Programmiersprachen C und Delphi finden sich, je nach installiertem Softwarepaket, in dem Unterverzeichnis

```
Win32\MCStep API\
```

respektive

```
API\
```

die folgenden Dateien:

- MCStepAPI.h für C/C++,
- MCStepAPI.Pas für Delphi,
- MCStepAPI.Pdf (dieses Dokument) sowie
- MCStep.DLL.

Die DLL-Datei wurde bereits bei der Installation des entsprechenden Softwarepakets in das Windows-Systemverzeichnis des Zielrechners kopiert. Sie ist lediglich noch einmal zusätzlich in diesem Unterverzeichnis gespeichert, falls Sie für Testzwecke die Datei auf einem anderen Rechner benötigen.

Vorgehensweise

- Binden Sie die Header-Datei (z.B. MCStepAPI.h für C/C++) in Ihr Projekt ein und erstellen Sie in Ihrem Quellcode einen Verweis (bei C/C++ über **`#include`**, bei Delphi über **`uses`**).
- Kompilieren Sie Ihr Projekt neu. Die eingebundenen Header-Dateien dürfen nicht zu Fehlern führen.
- Laden Sie Ihrem Projekt an geeigneter Stelle die MCStep.DLL für die weitere Verwendung. In C++ sähe das z.B. wie folgt aus:

```
HMODULE hDLL = LoadLibrary("MCStep.DLL");
if (!hDLL)
  // Fehlerbehandlung
```

- Suchen Sie sich die benötigten Funktionen aus der API zusammen und binden Sie Ihr Programm mit der dynamisch geladenen DLL, z.B. wie folgt in C++:

```
MCStep_Online = GetProcAddress(hDLL, "MCStep_Online");
MCStep_ProgramWrite = GetProcAddress(hDLL, "MCStep_ProgramWrite");
```

- Anschliessend können Sie die so dynamisch eingebundenen Funktionen in Ihrem Programm nutzen:

```
MCStep_Online();    // System online schalten
```

Andere Programmiersysteme

Lediglich die Programmsysteme C/C++ und Delphi werden direkt unterstützt. Für alle anderen Systeme können Sie sich jedoch auf Basis dieser Dokumentation und der vorhandenen Header-Dateien vergleichsweise einfach eine eigene Implementation zusammenstellen. Uns sind Referenzimplementationen in eine Vielzahl von Systemen bekannt, unter anderem z.B. in **`LapView`**. Leider können wir aus urheberrechtlichen Gründen diese Dokumente Ihnen hier nicht zugänglich machen.

2.2 Funktionsübersicht

Im Folgenden finden Sie eine Übersicht aller Funktionen der MCStep API.

Programme verwalten und übertragen

Am häufigsten benötigt werden in der Praxis die Befehle zum Übertragen und Auslesen von Programmen. Bitte beachten Sie hierzu auch das Beispiel in Kapitel 2.5.

LONG MCStep_ProgramRead(ULONG dwProgram, PCHAR szCmd)

Liest ein Programm aus der MCStep Steuerung. **dwProgram** enthält den zu lesenden Programmspeicherplatz, **szCmd** zeigt auf einen initialisierten Puffer. Dieser Puffer sollte mindestens 64kB gross sein.

LONG MCStep_ProgramWrite(ULONG dwProgram, PCHAR szCmd)

Compiliert ein Programm und übertragt es an die MCStep Steuerung. **dwProgram** enthält die Nummer des zu schreibenden Programms, **szCmd** den gesamten Quellcode dieses Programms. Zeilenenden sind jeweils durch die Kombination CR/LF (ASCII 13/10) zu kennzeichnen, das Ende des Programms nach C-Stringkonventionen mit einem Nullbyte.

LONG MCStep_ProgramActiveGet(VOID)

Liest die Nummer des aktuell laufenden Programms in der MCStep Steuerung.

LONG MCStep_ProgramCountGet(VOID)

Gibt die Anzahl der verfügbaren MCStep Programme in der Steuerung zurück.

LONG MCStep_ProgramCountSet(ULONG dwNewValue)

Setzt die Anzahl der verfügbaren MCStep Programme in der Steuerung. **dwNewValue** enthält die Anzahl der gewünschten Programme. Bitte beachten Sie, dass durch Ausführung dieser Funktion alle in der Steuerung hinterlegen MCStep Programme gelöscht werden und neu übertragen werden müssen! Verwenden Sie diese Funktion nur, wenn Sie sich über die Konsequenzen bewusst sind. Ein veränderter Wert für die Anzahl Programme ändert automatisch auch die maximale Zeilenanzahl.

LONG MCStep_ProgramSizeGet(VOID)

Ermittelt die maximale Anzahl der Zeilen für ein MCStep Programm.

LONG MCStep_ProgramSizeSet(ULONG dwNewValue)

Setzt die maximale Anzahl der Zeilen für ein MCStep Programm. **dwNewValue** enthält die Anzahl der gewünschten Programme. Bitte beachten Sie, dass durch Ausführung dieser Funktion alle in der Steuerung hinterlegen MCStep Programme gelöscht werden und neu übertragen werden müssen! Verwenden Sie diese Funktion nur, wenn Sie sich über die Konsequenzen bewusst sind. Ein veränderter Wert für die maximale Anzahl der Zeilen ändert automatisch die verfügbare Programmzahl.

Online und Offline

Bevor Sie mit der MCStep Steuerung kommunizieren, müssen Sie die API online schalten.

LONG MCStep_Online(VOID)

Schaltet die API online. Gegebenenfalls wird nach angeschlossenen Steuerungen gesucht.

LONG MCStep_Offline(VOID)

Schaltet die API offline.

Nachkommastellen / Positioniergenauigkeit

MCStep speichert intern alle Positionen als ganzzahlige Werte. Um je nach gewählter Auflösung auch nicht ganzzahlige Positionswerte anfahren zu können, kann die Anzahl der Nachkommastellen für jede Position bestimmt werden. Praktisch wird aus dem Zahlenwert 12345 je nach Anzahl der Kommastellen die Position

- Ohne Kommastellen: 12345
- Mit einer Kommastelle: 1234,5
- Mit zwei Kommastellen: 123,45
- Mit drei Kommastellen: 12,345

Diese Auswahl hat also direkten Einfluss auf die Abarbeitung der Programme. Stimmt die eingestellte Anzahl der Nachkommastellen nicht mit der Anzahl zusammen, die im Programm verwendet wurde, führt dies zwangsläufig zu falschen Positionen.

LONG MCStep_AxisResolutionGet(ULONG dwAxis)

Als Parameter wird die Achsnummer übergeben, zurückgeliefert wird die einstellte Anzahl der Nachkommastellen.

LONG MCStep_AxisResolutionSet(ULONG dwAxis, ULONG dwNewValue)

Als Parameter wird die Achsnummer sowie die gewünschte Anzahl der Nachkommastellen von 0 bis 3 übergeben. Der Rückgabewert ist **MCSTEP_ERR_NONE** im Erfolgsfall.

Einzelne Befehle codieren und übertragen

Mit der MCStep API können Sie direkt einzelne Befehle in den intern verwendeten Binärcode übersetzen und dabei die Syntax überprüfen oder auch diese Befehle gezielt in eine bestimmte Zeile des ausgeführten Programms der Steuerung speichern. Diese Funktionen sind eher für technisch versierte, sehr spezialisierte Anwendungen vorgesehen. Der übliche Funktionsumfang wird durch die Funktionen zur Verwaltung und Übertragung von Programmen gedeckt.

LONG MCStep_CommandRead(ULONG dwProgram, ULONG dwLine, ULONG dwLineIdx, PCHAR szCmd)

Liest einen Befehl aus einer angegebenen Programmzeile der Steuerung. dwProgram ist der Speicherplatz, dwLine die Programmzeile und dwLineIdx die Subzeile zwischen 1 und 4. Der Parameter szCmd zeigt auf einen initialisierten Puffer, in welchen der Befehl im Klartext geschrieben wird. Dieser Puffer muss mindestens 256 Byte gross sein.

LONG MCStep_CommandWrite(ULONG dwProgram, ULONG dwLine, ULONG dwLineIdx, PCHAR szCmd)

Schreibt einen MCStep Befehl in die angegebene Programmzeile der Steuerung. **dwProgram** ist der Speicherplatz, **dwLine** die Programmzeile und **dwLineIdx** die Subzeile zwischen 1 und 4. Der Parameter **szCmd** zeigt auf den entsprechenden MCStep Befehl.

LONG MCStep_EncodeCommand(PCHAR szCmd)

Gibt den compilierten Binärcode für eine einzelne Programmzeile zurück: **szCmd** zeigt auf den zu compilierenden Befehl, der Rückgabewert ist die Programmzeile. Bei gesetztem Bit 31 ist der Rückgabewert kein Befehlscode, sondern ein Fehlercode wie in Kapitel 2.3 beschrieben.

LONG MCStep_DecodeCommand(PCHAR szCmd, ULONG dwCode)

Übersetzt einen binären Befehlscode zurück in lesbaren Text. **szCmd** zeigt auf einen initialisierten Puffer mit mindestens 256 Byte Länge für den lesbaren Text, **dwCode** enthält den zu übersetzenden Binärcode.

Weitere Funktionen

Einige weitere Funktionen können je nach Anwendungsfall für Sie von Bedeutung sein.

LONG MCStep_Reset(VOID)

Setzt die MCStep Steuerung zurück. Entspricht dem Aus- und wieder Einschalten des Geräts.

LONG MCStep_SingleStep(PCHAR szCmd)

Übermittelt einen einzelnen Befehl, der sofort ausgeführt wird. Bitte beachten Sie hierzu die Erläuterungen im Kapitel 2.4.

LONG MCStep_ExpertMode(LONG liStatus)

Aktiviert oder deaktiviert den Expertenmodus. Zur Aktivierung enthält `liStatus` einen Wert ungleich Null. Bei aktivem Expertenmodus sind einige logische Überwachungen ausgeschaltet. So wird dann nicht mehr geprüft, ob vor der Übertragung eines Programms die Steuerung korrekterweise den Automatikmodus verlässt. Verwenden Sie diese Funktion auf eigene Gefahr!

LONG MCStep_IsAutomatic(VOID)

Überprüft, ob die Steuerung sind gegenwärtig im Automatikmodus befindet. Liefert einen Wert ungleich Null zurück, wenn der Automatikmodus aktiviert ist.

LONG MCStep_MemoryInfoGet(PULONG pdwPrograms, PULONG pdwLines, PULONG pdwTotal)

Liefert erweiterte Informationen über die parametrierte Speicheraufteilung der MCStep Steuerung. Alle Parameter zeigen auf eine initialisierte 32-Bit Integervariable. In `pdwPrograms` wird die Anzahl der verfügbaren Programme geschrieben, in `pdwLines` die maximale Anzahl Zeilen und in `pdwTotal` die verfügbare Anzahl Programmvariablen.

2.3 Fehlercodes und Rückmeldungen

Alle möglichen Fehlercodes sind ebenfalls in den bereits zu Beginn dieses Kapitels näher beschriebenen Header-Dateien abgelegt. Hier eine Liste mit entsprechender Erläuterung:

Code	Konstante	Bedeutung
0	MCSTEP_ERR_NONE	Keine Fehler, der letzte Befehl wurde erfolgreich ausgeführt. Diese Rückmeldung kennzeichnet die fehlerfreie Ausführung.
-1	MCSTEP_ERR_ILLEGAL_AXIS	Die angegebene Achsnummer ist nicht zulässig. Lediglich Achsen von 1-4 (X, Y, Z, A) werden unterstützt.
-2	MCSTEP_ERR_ILLEGAL_PARAMETER	Der angegebene Parameter ist ungültig oder liegt ausserhalb des definierten Wertebereichs. Bitte vergleichen Sie die Hinweise in Ihrer Programmierdokumentation.
-3	MCSTEP_ERR_NO_COMMAND	Es wurde kein MCStep Befehl angegeben.
-4	MCSTEP_ERR_UNKNOWN_COMMAND	Der übergebene MCStep Befehl ist unbekannt oder ungültig.
-5	MCSTEP_ERR_MISSING_PARAMETERS	Der angegebene Befehl erwartet mehr Parameter als angegeben wurden.
-6	MCSTEP_ERR_TOO_SMALL	Der angegebene Parameter liegt unterhalb des zulässigen Wertebereichs für diesen Befehl.
-7	MCSTEP_ERR_TOO_LARGE	Der angegebene Parameter liegt oberhalb des zulässigen Wertebereichs für diesen Befehl.
-8	MCSTEP_ERR_VMC_DLL	Die MCStep API konnte keine Verbindung zur benötigten VMC Compatibility Layer aufbauen. Bitte installieren Sie das Softwarepaket neu.
-9	MCSTEP_ERR_NO_DEVICE	Die Funktion kann nur mit angeschlossener MCStep Steuerung ausgeführt werden, das System ist aber nicht online verbunden.
-10	MCSTEP_ERR_COMM	Bei der Ausführung der Funktion trat ein Kommunikationsfehler auf.
-11	MCSTEP_ERR_ILLEGAL_PROGRAM	Die angegebene Programmnummer ist ungültig.
-12	MCSTEP_ERR_ILLEGAL_LINE	Die angegebene Zeilennummer ist ungültig.
-13	MCSTEP_ERR_OS_VERSION	Das Betriebssystem der Steuerung ist für eine Verwendung dieser API zu alt.
-14	MCSTEP_ERR_AUTOMATIC	Für die gewählte Funktion darf die Steuerung nicht im Automatik Modus laufen.
-15	MCSTEP_ERR_NO_SINGLESTEP	Ungültige Aufforderung zum Einzelfehl (Kapitel 2.4).
-16	MCSTEP_ERR_PREV_SINGLESTEP	Der letzte Einzelschritt ist noch nicht beendet.
-17	MCSTEP_ERR_NO_SINGLESTEP_PRG	Das Einzelschrittprogramm ist nicht installiert.
-18	MCSTEP_ERR_NO_MC100	Die angeschlossene Steuerung ist vom falschen Typ.
-19	MCSTEP_ERR_NO_MCSTEP	Das MCStep SPS-Programm ist nicht installiert.
-20	MCSTEP_ERR_INVALID_DATA	Ungültige Daten (allgemeiner Fehler).

Tabelle 1 – Fehlercodes und Rückmeldungen

2.4 Einzelschritt und dynamische Positionen

Im Verzeichnis

```
Vorlagen\
```

der Installation findet sich eine Datei mit dem Namen **SingleCommand.MSS**. Wenn diese Datei im Programmspeicherplatz 1 Ihrer MCStep Steuerung abgelegt ist, dann können Sie die Funktion **Einzelbefehl ausführen** nutzen. Dies bedeutet, dass Sie einen Befehl zur Laufzeit vom PC zur Steuerung senden können und ihn dort direkt ausführen können, ohne den Automatikablauf der Steuerung zu unterbrechen.

Es gibt keine prinzipielle Limitierung der Befehle, wobei allerdings verständlich sein sollte, dass Label- und Sprungbefehle sowie andere Anweisungen, die sich auf die Programmstruktur auswirken, hier nichts verloren haben.

Nutzung des Einzelschritts

- Stellen Sie sicher, dass die Datei **SingleCommand.MSS** im Speicherplatz 1 der Steuerung abgelegt ist. Übertragen Sie ggf. bei jedem Start Ihres Anwendungsprogramms mit Hilfe der Funktion **MCStep_ProgramWrite()** die Datei in den Speicherplatz 1.
- Stellen Sie sicher, dass die Steuerung sich im Automatikbetrieb befindet. Sie können hierzu die Funktion **MCStep_IsAutomatic()** verwenden.
- Codieren Sie den auszuführenden Befehl als String und rufen Sie die Funktion **MCStep_SingleStep()** auf. Angenommen, Sie möchten die Y-Achse dynamisch auf eine Position verfahren, die Sie im Programmablauf ermitteln, kann das unter C/C++ wie folgt aussehen:

```
CHAR szCmd[256];
sprintf(szCmd, „ABS Y %d“, liZielPosition);
MCStep_SingleStep(szCmd);
```

- Die Funktion kehrt sofort zurück, wenn der Befehl erfolgreich an die MCStep Steuerung übermittelt wurde. Die API wartet also nicht darauf, dass die Steuerung den Befehl ausführt, also z.B. die Achse Y die angegebene Zielposition erreicht.
- Der nächste Einzelschritt kann erst ausgeführt werden, wenn der aktuelle beendet wurde. Sonst kehrt die Funktion **MCStep_SingleStep()** mit dem Fehler **MCSTEP_ERR_PREV_SINGLESTEP** zurück.

2.5 Beispiel

Nachfolgend finden Sie den Quellcode eines kleinen C++-Fragments, welches ein MCStep Programm von der Festplatte liest und an die angeschlossene MCStep Steuerung überträgt.

```
BOOL TransferFile(ULONG dwProgram, PCHAR szFileName)
{
HMODULE hDLL = LoadLibrary(„MCStep.DLL“);
If (!hDLL)
    Return FALSE;

MCStep_Online = (ULONG (WINAPI *)(VOID)) GetProcAddress(hDLL, "MCStep_Online");
MCStep_ProgramWrite = (ULONG (WINAPI *)(ULONG, PCHAR))
    GetProcAddress(hDLL, "MCStep_Online");

If (MCStep_Online() != MCSTEP_ERR_NONE)
{
    FreeLibrary(hDLL);
    Return FALSE;
}

HANDLE hFile = CreateFile(szFileName, GENERIC_READ, FILE_SHARE_READ, NULL,
    OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
If (hFile == INVALID_HANDLE_VALUE)
{
    FreeLibrary(hDLL);
    Return FALSE;
}

ULONG dwSize = GetFileSize(hFile, NULL);
PCHAR szBuffer = new CHAR[dwSize + 1];
ZeroMemory(szBuffer, dwSize + 1);

ULONG dwReadSize = 0;
If (!ReadFile(hFile, szBuffer, dwSize, &dwReadSize, NULL))
{
    delete szBuffer;
    CloseHandle(hFile);
    FreeLibrary(hDLL);
    Return FALSE;
}
CloseHandle(hFile);

If (MCStep_ProgramWrite(dwProgram, szBuffer) != MCSTEP_ERR_NONE)
{
    delete szBuffer;
    FreeLibrary(hDLL);
    Return FALSE;
}

delete szBuffer;
FreeLibrary(hDLL);
Return TRUE;
```


Kapitel 3 Direkte Programmierung

Sollten Sie die in Kapitel 2 beschriebene MCStep API nicht nutzen können oder nicht nutzen wollen, so können Sie selbstverständlich auch direkt auf das System zugreifen. Bitte beachten Sie jedoch folgende Hinweise:

Erfahrung mit Protokollen und NC-System vorausgesetzt!

Es ist nicht wirklich trivial, direkt mit einem komplexen, in MC-1A Sprache codierten Interpretersystem zu kommunizieren. Sie sollten in der Vergangenheit vergleichbare Erfahrungen gesammelt haben.

Keine direkte Kundenunterstützung!

Die einzige direkte und kostenfreie Unterstützung, die MICRDESIGN Ihnen in dieser Sache liefert, ist vorliegende Dokumentation sowie die anderen, nachfolgend erwähnten Dokumente. Falls Sie Unterstützung bei Ihrem Projekt benötigen, wenden Sie sich bitte zuvor direkt an MICRODESIGN.

Die MC-1 SPS-Sprache

Falls Sie noch keine Erfahrung mit der MC-1 SPS-Sprache gesammelt haben, beachten Sie bitte unbedingt das Programmierhandbuch **`MC-1A Dokumentation`** im Downloadbereich von www.microdesign.de.

Weiterführende Themen

Falls Sie mit einem Windows-basiertem System arbeiten, können Sie die Kommunikation mit der Steuerung über unser VMC Server System, die VMC OCX Control oder die VMC Compatibility Layer DLL abwickeln. Bitte beachten Sie die dem jeweiligen Produkt beiliegende Dokumentation. Wenn Sie ein anderes Betriebssystem verwenden, müssen Sie auch die Kommunikation mit dem System selbst kontrollieren. Beachten Sie hierzu das Dokument **`MC-1 Protokolle`** im Downloadbereich von www.microdesign.de.

Wichtiger Hinweis!

Lücken innerhalb der nachfolgenden Ressourcenlisten bedeuten nicht, dass die entsprechenden Merker und Variablen nicht verwendet werden. Vielmehr handelt es sich hierbei vermutlich um interne Systemdaten der MC100 Steuerung, die für den Ablauf des Anwendungsprogramms zwingend benötigt werden und nicht verändert werden dürfen.

3.1 Verwendete Merker

Nummer	Name	Funktion
1	M_SYSTEM_OK	Es liegt keine Störmeldung vor
14	M_SPRACHE	Sprachumschaltung aktiv
15	M_E3FUN	E3 Sonderfunktion aktiv
16	M_E4E8FUN	E4/E8 Sonderfunktion aktiv
17	M_E_STEP	Impulsauswertung E3 aktiv
20	M_SERVICE	Servicebetrieb über Display angewählt
21	M_PARAM	Parametereingabe über Display angewählt
22	M_PROGR	Programmeingabe über Display angewählt
23	M_AUTOMA	Automatikbetrieb über Display angewählt
24	M_HANDBE	Handbetrieb über Display angewählt
25	M_HANDSV	Handbetrieb für Servomotoren über Display angewählt.
26	M_PROG_AKT	MCStep Programm läuft
32	M_TEACH_AKT	Teach-In Modus aktiv
33	M_FMAX_HAND	Handbetrieb für Schrittmotoren in Fmin/Fmax Funktion
34	M_TEACH_OK	Freigabe für Teach-Werte
37	M_INTERPO	Interpolation aktiv
40	M_JUMP_RES	Verknüpfungsergebnis bedingte Sprünge
56	M_AUTO_STOP	Stop des Automatikbetriebs gewünscht
57	M_AUTO_PC	Automatikbetrieb wird durch PC angefragt
70	M_E1E2FUN	E1/E2 Sonderfunktion aktiv
100	M_RUNDA_SMA1	Rundachsenanwahl Schrittmotorachse 1
101	M_RUNDA_SMA2	Rundachsenanwahl Schrittmotorachse 2
102	M_RUNDA_SMA3	Rundachsenanwahl Schrittmotorachse 3
103	M_RUNDA_SMA4	Rundachsenanwahl Schrittmotorachse 4
104	M_RUNDA_SVA1	Rundachsenanwahl Servomotorachse 1
105	M_RUNDA_SVA2	Rundachsenanwahl Servomotorachse 2
106	M_RUNDA_SVA3	Rundachsenanwahl Servomotorachse 3
107	M_RUNDA_SVA4	Rundachsenanwahl Servomotorachse 4
111	M_REF_OK_SMA1	Referenzfahrt ok Schrittmotorachse 1
112	M_REF_OK_SMA2	Referenzfahrt ok Schrittmotorachse 2
113	M_REF_OK_SMA3	Referenzfahrt ok Schrittmotorachse 3
114	M_REF_OK_SMA4	Referenzfahrt ok Schrittmotorachse 4
115	M_LAST_DIR_SMA1	Letzte Richtung Schrittmotorachse 1 (aus = positive Richtung, ein = negative Richtung)
116	M_LAST_DIR_SMA2	Letzte Richtung Schrittmotorachse 2 (aus = positive Richtung, ein = negative Richtung)

Nummer	Name	Funktion
117	M_LAST_DIR_SMA3	Letzte Richtung Schrittmotorachse 3 (aus = positive Richtung, ein = negative Richtung)
118	M_LAST_DIR_SMA4	Letzte Richtung Schrittmotorachse 4 (aus = positive Richtung, ein = negative Richtung)
119	M_STOP_AKT_SMA1	Stop aktiv Schrittmotorachse 1
120	M_STOP_AKT_SMA2	Stop aktiv Schrittmotorachse 2
121	M_STOP_AKT_SMA3	Stop aktiv Schrittmotorachse 3
122	M_STOP_AKT_SMA4	Stop aktiv Schrittmotorachse 4
123	M_STOP_AKT_SVA1	Stop aktiv Servomotorachse 1
124	M_STOP_AKT_SVA2	Stop aktiv Servomotorachse 2
125	M_STOP_AKT_SVA3	Stop aktiv Servomotorachse 3
126	M_STOP_AKT_SVA4	Stop aktiv Servomotorachse 4
127	M_REFAKT_A1	Referenzfahrt Achse 1 läuft
128	M_REFAKT_A2	Referenzfahrt Achse 2 läuft
129	M_REFAKT_A3	Referenzfahrt Achse 3 läuft
130	M_REFAKT_A4	Referenzfahrt Achse 4 läuft
131	M_REFMI_A1	Fahrtrichtung bei Referenzfahrt Achse 1 (aus = negative Richtung, ein = positive Richtung)
132	M_REFMI_A2	Fahrtrichtung bei Referenzfahrt Achse 2 (aus = negative Richtung, ein = positive Richtung)
133	M_REFMI_A3	Fahrtrichtung bei Referenzfahrt Achse 3 (aus = negative Richtung, ein = positive Richtung)
134	M_REFMI_A4	Fahrtrichtung bei Referenzfahrt Achse 4 (aus = negative Richtung, ein = positive Richtung)
135	M_RUNDA_A1	Rundachsenanwahl Achse 1
136	M_RUNDA_A2	Rundachsenanwahl Achse 2
137	M_RUNDA_A3	Rundachsenanwahl Achse 3
138	M_RUNDA_A4	Rundachsenanwahl Achse 4
140	M_ENDLOS_A1	Achse 1 Endlosbetrieb
141	M_ENDLOS_A2	Achse 2 Endlosbetrieb
142	M_ENDLOS_A3	Achse 3 Endlosbetrieb
143	M_ENDLOS_A4	Achse 4 Endlosbetrieb
144	M_ENDLOS_SMA1	Schrittmotorachse 1 Endlosbetrieb
145	M_ENDLOS_SMA2	Schrittmotorachse 2 Endlosbetrieb
146	M_ENDLOS_SMA3	Schrittmotorachse 3 Endlosbetrieb
147	M_ENDLOS_SMA4	Schrittmotorachse 4 Endlosbetrieb
148	M_ENDLOS_SVA1	Servomotorachse 1 Endlosbetrieb
149	M_ENDLOS_SVA2	Servomotorachse 2 Endlosbetrieb
150	M_ENDLOS_SVA3	Servomotorachse 3 Endlosbetrieb

Nummer	Name	Funktion
151	M_ENDLOS_SVA4	Servomotorachse 4 Endlosbetrieb
152	M_LAST_DIR_SVA1	Letzte Richtung Servomotorachse 1 (aus = positive Richtung, ein = negative Richtung)
153	M_LAST_DIR_SVA2	Letzte Richtung Servomotorachse 2 (aus = positive Richtung, ein = negative Richtung)
154	M_LAST_DIR_SVA3	Letzte Richtung Servomotorachse 3 (aus = positive Richtung, ein = negative Richtung)
155	M_LAST_DIR_SVA4	Letzte Richtung Servomotorachse 4 (aus = positive Richtung, ein = negative Richtung)
159	M_1ACHSEN	MCStep läuft als 1-Achs System
160	M_2ACHSEN	MCStep läuft als 2-Achs System
161	M_3ACHSEN	MCStep läuft als 3-Achs System
162	M_4ACHSEN	MCStep läuft als 4-Achs System
163	M_A1_ESOEF	Konfiguration Endschalter Achse 1: aus = Endschalter sind Öffner, ein = Endschalter sind Schließer
164	M_A2_ESOEF	Konfiguration Endschalter Achse 2: aus = Endschalter sind Öffner, ein = Endschalter sind Schließer
165	M_A3_ESOEF	Konfiguration Endschalter Achse 3: aus = Endschalter sind Öffner, ein = Endschalter sind Schließer
166	M_A4_ESOEF	Konfiguration Endschalter Achse 4: aus = Endschalter sind Öffner, ein = Endschalter sind Schließer
167	M_A1_ESNO	Konfiguration Endschalter Achse 1: aus = Endschalter sind vorhanden, ein = Endschalter sind nicht vorhanden
168	M_A2_ESNO	Konfiguration Endschalter Achse 2: aus = Endschalter sind vorhanden, ein = Endschalter sind nicht vorhanden
169	M_A3_ESNO	Konfiguration Endschalter Achse 3: aus = Endschalter sind vorhanden, ein = Endschalter sind nicht vorhanden
170	M_A4_ESNO	Konfiguration Endschalter Achse 4: aus = Endschalter sind vorhanden, ein = Endschalter sind nicht vorhanden
171	M_A1_ESINV	Konfiguration Endschalter Achse 1: aus = Endschalter normal verwenden, ein = Endschalter sind invertiert
172	M_A2_ESINV	Konfiguration Endschalter Achse 2: aus = Endschalter normal verwenden, ein = Endschalter sind invertiert
173	M_A3_ESINV	Konfiguration Endschalter Achse 3: aus = Endschalter normal verwenden, ein = Endschalter sind invertiert
174	M_A4_ESINV	Konfiguration Endschalter Achse 4: aus = Endschalter normal verwenden, ein = Endschalter sind invertiert
201	M_STOE_SYST	Störungsmerker: eine Störung ist aufgetreten
202	M_STOE_SLA1	Schleppfehler Achse 1
203	M_STOE_EPA1	Endschalter Plus Achse 1
204	M_STOE_EMA1	Endschalter Minus Achse 1
205	M_STOE_RBA1	Regler bereit Achse 1

Nummer	Name	Funktion
206	M_STOE_SLA2	Schleppfehler Achse 2
207	M_STOE_EPA2	Endschalter Plus Achse 2
208	M_STOE_EMA2	Endschalter Minus Achse 2
209	M_STOE_RBA2	Regler bereit Achse 2
210	M_STOE_SLA3	Schleppfehler Achse 3
211	M_STOE_EPA3	Endschalter Plus Achse 3
212	M_STOE_EMA3	Endschalter Minus Achse 3
213	M_STOE_RBA3	Regler bereit Achse 3
214	M_STOE_SLA4	Schleppfehler Achse 4
215	M_STOE_EPA4	Endschalter Plus Achse 4
216	M_STOE_EMA4	Endschalter Minus Achse 4
217	M_STOE_RBA4	Regler bereit Achse 4
305	M_EPL_A1	Endschalter Plus bestromt für Achse 1
306	M_EMI_A1	Endschalter Minus bestromt für Achse 1
307	M_ERB_A1	Regler bereit bestromt für Achse 1
308	M_EVZ_A1	Sondereingang bestromt für Achse 1
309	M_EPL_A2	Endschalter Plus bestromt für Achse 2
310	M_EMI_A2	Endschalter Minus bestromt für Achse 2
311	M_ERB_A2	Regler bereit bestromt für Achse 2
312	M_EVZ_A2	Sondereingang bestromt für Achse 2
313	M_EPL_A3	Endschalter Plus bestromt für Achse 3
314	M_EMI_A3	Endschalter Minus bestromt für Achse 3
315	M_ERB_A3	Regler bereit bestromt für Achse 3
316	M_EVZ_A3	Sondereingang bestromt für Achse 3
317	M_EPL_A4	Endschalter Plus bestromt für Achse 4
318	M_EMI_A4	Endschalter Minus bestromt für Achse 4
319	M_ERB_A4	Regler bereit bestromt für Achse 4
320	M_EVZ_A4	Sondereingang bestromt für Achse 4
413	M_E3_BRK	Abbruch mit E3
511	M_TIMER1	Timer 1 läuft
512	M_TIMER2	Timer 2 läuft
513	M_TIMER3	Timer 3 läuft
514	M_TIMER4	Timer 4 läuft
550	M_RESET	Einschalten für vollständigen Steuerungs-Reset
571	M_STOE_COMM	Störung in der RS485 Kommunikation aufgetreten

Tabelle 2 - Verwendete Merker

3.2 Verwendete Variablen

Nummer	Name	Funktion
40	V_STOE	Kennzahl aktuelle Störung
41	V_STOE_OLD	Kennzahl vorherige Störung
42	V_STOE_TXT	Text-Nummer aktuelle Störung
50	V_VORSCH_A1	Vorschub Achse 1
51	V_VORSCH_A2	Vorschub Achse 2
52	V_VORSCH_A3	Vorschub Achse 1
53	V_VORSCH_A4	Vorschub Achse 2
54	V_SOLLP_A1	Sollposition Achse 1
55	V_SOLLP_A2	Sollposition Achse 2
56	V_SOLLP_A3	Sollposition Achse 3
57	V_SOLLP_A4	Sollposition Achse 4
58	V_ISTP_A1	Istposition Achse 1
59	V_ISTP_A2	Istposition Achse 2
60	V_ISTP_A3	Istposition Achse 3
61	V_ISTP_A4	Istposition Achse 4
86	V_VE_A1	Multiplikator Verfahreinheiten Achse 1 (Basis: µm)
87	V_VE_A2	Multiplikator Verfahreinheiten Achse 2 (Basis: µm)
88	V_VE_A3	Multiplikator Verfahreinheiten Achse 3 (Basis: µm)
89	V_VE_A4	Multiplikator Verfahreinheiten Achse 4 (Basis: µm)
90	V_A1_VMAX	Maximale Geschwindigkeit Achse 1
91	V_A2_VMAX	Maximale Geschwindigkeit Achse 2
92	V_A3_VMAX	Maximale Geschwindigkeit Achse 3
93	V_A4_VMAX	Maximale Geschwindigkeit Achse 4
120	V_AUTO_CNT1	Aktueller Wert für Zähler 1
121	V_AUTO_CNT2	Aktueller Wert für Zähler 2
122	V_AUTO_CNT3	Aktueller Wert für Zähler 3
123	V_AUTO_CNT4	Aktueller Wert für Zähler 4
124	V_AUTO_CNT5	Aktueller Wert für Zähler 5
125	V_AUTO_CNT6	Aktueller Wert für Zähler 6
126	V_AUTO_CNT7	Aktueller Wert für Zähler 7
127	V_AUTO_CNT8	Aktueller Wert für Zähler 8
130	V_OVERRD	Aktueller Wert des Achsen-Overrides
132	V_BRKPOS_A1	Halteposition Achse 1
133	V_BRKPOS_A2	Halteposition Achse 2
134	V_BRKPOS_A3	Halteposition Achse 3

Nummer	Name	Funktion
135	V_BRKPOS_A4	Halteposition Achse 4
190	V_ENDS_WEG_SMA1	Endschalter-Überfahrweg für Schrittmotorachse 1
191	V_ENDS_WEG_SMA2	Endschalter-Überfahrweg für Schrittmotorachse 2
192	V_ENDS_WEG_SMA3	Endschalter-Überfahrweg für Schrittmotorachse 3
193	V_ENDS_WEG_SMA4	Endschalter-Überfahrweg für Schrittmotorachse 4
196	V_RAMP_A1	Programmierte Rampe Achse 1
197	V_RAMP_A2	Programmierte Rampe Achse 2
198	V_RAMP_A3	Programmierte Rampe Achse 3
199	V_RAMP_A4	Programmierte Rampe Achse 4
201	V_TIMER1	Aktueller Wert des Timer 1 in 100ms Intervallen
202	V_TIMER2	Aktueller Wert des Timer 2 in 100ms Intervallen
203	V_TIMER3	Aktueller Wert des Timer 3 in 100ms Intervallen
204	V_TIMER4	Aktueller Wert des Timer 4 in 100ms Intervallen
242	V_ANZOFFS	Textoffset für Sprachumschaltung
681	V_PRG_LEN	Länge eines MCStep-Programms in Zeilen
690	V_PRG_NR	Aktuell gewähltes MCStep-Programm
691	V_PRG_LEN2	Anzahl benötigter Variablen für das aktuelle MCStep-Programm
698	V_PRG_MAX	Maximale Anzahl MCStep-Programme
865	V_FREQ_A1	Geschwindigkeit/Frequenz Achse 1
866	V_FREQ_A2	Geschwindigkeit/Frequenz Achse 2
867	V_FREQ_A3	Geschwindigkeit/Frequenz Achse 3
868	V_FREQ_A4	Geschwindigkeit/Frequenz Achse 4
990	V_SWSPL_SMA1	Software-Endschalter Plus Schrittmotorachse 1
991	V_SWSMI_SMA1	Software-Endschalter Minus Schrittmotorachse 1
992	V_SWSPL_SMA2	Software-Endschalter Plus Schrittmotorachse 2
993	V_SWSMI_SMA2	Software-Endschalter Minus Schrittmotorachse 2
994	V_SWSPL_SMA3	Software-Endschalter Plus Schrittmotorachse 3
995	V_SWSMI_SMA3	Software-Endschalter Minus Schrittmotorachse 3
996	V_SWSPL_SMA4	Software-Endschalter Plus Schrittmotorachse 4
997	V_SWSMI_SMA4	Software-Endschalter Minus Schrittmotorachse 4
998	V_SWSPL_SVA1	Software-Endschalter Plus Servomotorachse 1
999	V_SWSMI_SVA1	Software-Endschalter Minus Servomotorachse 1
1000	V_SWSPL_SVA2	Software-Endschalter Plus Servomotorachse 2
1001	V_SWSMI_SVA2	Software-Endschalter Minus Servomotorachse 2
1002	V_SWSPL_SVA3	Software-Endschalter Plus Servomotorachse 3
1003	V_SWSMI_SVA3	Software-Endschalter Minus Servomotorachse 3

Nummer	Name	Funktion
1004	V_SWSPL_SVA4	Software-Endschalter Plus Servomotorachse 4
1005	V_SWSMI_SVA4	Software-Endschalter Minus Servomotorachse 4
1110	V_MAX_VARIABLE	Maximale Anzahl Variablen, die für MCStep-Programme zur Verfügung stehen (ergibt /2 die maximale Anzahl Programmzeilen)
1111	V_MCSTEP_KENNUNG	Enthält immer den Wert "0x3f3f3f" (hexadezimal)
1112	V_MCS_PRGBEG	Erste Variable der MCStep-Programme (normalerweise 1200)
1113	V_MCSTEP_VERSION	Versionsnummer des MCStep-Anwendungsprogramms. Gültig ab Version 600 (6.0).
1200	V_PROG_BEG	Erste Zeile des ersten MCStep-Programms (Achtung! Kann sich ändern! Bitte verwenden Sie den Inhalt der Variable 1112).

Tabelle 3 - Verwendete Variablen

Raum für Ihre Notizen