Bewegungssteuerung per Schrittmotoren über Sercos

von Prof. Dr. Engels

Lassen sich mit vergleichsweise kostengünstigen Schrittmotoren auch komplexe Motion-Control-Aufgaben über den Automatisierungsbus Sercos erledigen? Dieser Frage ging ein industrienahes Entwicklungsprojekt an der Hochschule Fulda nach.

Bildquelle: © Hochschule Fulda / Robert Gross

Schrittmotoren kleiner Leistung eignen sich besonders dort, wo preisgünstigere Lösungen realisiert werden sollen, als sie mit Servomotoren möglich wären. Gegenüber letzteren haben Schrittmotoren jedoch auch verschiedene Nachteile wie beispielsweise unerwünschte Rastmomente oder ein bei höheren Drehzahlen stark abfallendes Drehmoment. Mittels Inkrementalgebern und einer hochauflösenden Mikroschrittsteuerung lassen sich allerdings sowohl etwaige Schrittverluste vermeiden als auch die Drehmomentwelligkeit stark reduzieren, so dass die angesprochenen Nachteile im Kontext des erforderlichen Kosten-Nutzen-Verhältnisses akzeptierbar sind.

Sofern nun aber mit Schrittmotoren komplexe Motion-Control-Aufgaben wie beispielsweise synchrone oder koordinierte Achsbewegungen unter Nutzung des dafür prädestinierten Automatisierungsbusses Sercos durchgeführt werden sollen, ist es hilfreich, wenn die Programmiermethoden und Antriebsfunktionalitäten von modernen Servoantrieben vorhanden sind. Vor diesem Hintergrund wurde an der Hochschule Fulda ein Projekt initiiert mit dem Ziel, die Möglichkeit der Kommandierung der Motoren an Schrittmotorantrieben mit Matlab unter Nutzung des Motion-Kernels im Motion-Controller zu untersuchen. Ein weiteres

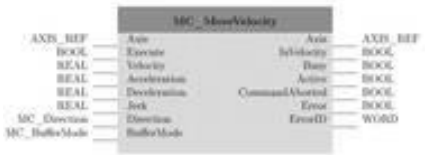
Teilziel der Untersuchung war die Entwicklung einer Testumgebung, die aus Gründen der Entwicklungseffizienz automatisierte und reproduzierbare Testszenarien ermöglicht.

Bildquelle: © Elmar Engels

Bild 1: Der Versuchsaufbau an der Hochschule Fulda.

Grundsätzlich bietet der Markt für solche Anwendungen nur eine sehr geringe Auswahl an Antriebsreglern. Für die Projektdurchführung entschieden sich die Verantwortlichen letztlich für die Kombination eines Sercos-Schrittmotorantriebs der Firma Cannon Automata (SMC3) mit einem Motion-Controller vom Typ IndraMotion MLC von Bosch Rexroth unter Verwendung der Rapid-Control-Prototyping-Software von Mathworks (**Bild 1**). Beim IndraMotion MLC handelt es sich um einen Motion-Controller mit Sercos-Master-Funktionalität, der die Berechnung der Antriebsbewegungen in der Steuerung übernehmen kann. Beispiele für komplexe Bewegungsprozesse sind die elektronische Getriebesynchronisation oder Kurvenscheibenbewegungen. Der Schrittmotorregler SMC3 ist ein relativ junges Produkt, das die Sercos-Parameter des ‚sercos function specific profile drive' (FSP-Drive) unterstützt.

Rapid-Control-Prototyping in der Automatisierungstechnik



Bildquelle: © Hochschule Fulda

Bild 2: Die Kommandierung einer Geschwindigkeitsregelung kann nach PLCopen über einen Funktionsbaustein unter Angabe der Bewegungssollgeschwindigkeit und der Bewegungsgrenzwerte für die Beschleunigungs- und Verzögerungsrampen erfolgen.

Bildquelle: © Hochschule Fulda

Bild 3: Matlab-Funktion zur Kommandierung einer Geschwindigkeitsregelung.

Der verwendete Motion Controller unterstützt alle Schnittstellen des sogenannten Open Core Engineering von Bosch Rexroth und kann neben der klassischen IEC-Programmierung unter anderem mit den Programmiersprachen C/C++, Java oder Lua programmiert werden. Für die Rapid-Control-Prototyping-Engineering-Umgebung Matlab, die sich insbesondere für eine schnelle und flexible Software-Entwicklung im Rahmen des Prototypenbaus oder für Versuchsaufbauten anbietet, gibt es von Bosch Rexroth das Softwarepaket mlpi4Matlab. Die hierin enthaltene Motion-Bibliothek beinhaltet über 50 Funktionen zum Lesen und Schreiben von Konfigurationseinstellungen sowie über 30 Funktionen zur Ausführung von Bewegungskommandos. Die Bewegungskommandos lehnen sich dabei an die Definition der Funktionsbausteine der PLCopen an. Als Beispiel sei der Funktionsbaustein MC_MoveVelocity der PLCopen für die Geschwindigkeitsregelung einer einzelnen Achse genannt (**siehe Bild 2**). **Bild 3** zeigt die Syntax der entsprechenden Matlab-Funktion m4MMotionMoveVelocity. Diese Funktion besitzt ähnliche Aufrufparameter und liefert unter anderem einen Funktionshandle als Rückgabewert, mit dem Informationen über die Abarbeitung der Geschwindigkeitsregelung abgefragt werden können.

Die in der Zielsetzung angesprochenen Funktionalitäten der mlpi4-Matlab-Software umfassen Befehle sowohl für das Kommandieren von Einzelachsen als auch für das Kommandieren synchroner Achsbewegungen. Für diese Funktionalitäten benötigt das System mindestens zwei SMC3-Antriebsregler. Aufgrund des Bereichs der elektrischen Motorsignale – 19 bis 48 V(DC) – wurden kostengünstige Motoren in der Baugröße Nema 17 mit 24-V-Motorspannung verwendet: Am ersten SMC3 wurde ein Motor ohne Geber angeschlossen, am zweiten Regler wurde ein Motor mit einem Inkrementalgeber mit einer Auflösung von 1000 Inkrementen pro Umdrehung genutzt.

Beide Antriebsregler verfügen über eine weitere Schnittstelle, die es ermöglicht, digitale Signale (DIO) zu lesen und zu schreiben. Über diese DIO lassen sich Referenzmarken, Lage-End-Schalter oder einfache Sensoren und Aktoren ansprechen. Für das System wurden LED-Leuchtmelder mit Tastfunktion gewählt, da damit Signale sehr einfach beziehungsweise schnell visualisierbar sind. Um das Motion-Control-System über eine Softwareschnittstelle aus Matlab ansprechen zu können, kam schließlich das Software-Entwicklungswerkzeug MLPI SDK 1.0 zum Einsatz, welches ebenfalls Bestandteil des Open Core Engineering von Bosch Rexroth ist.

Die Projektierung der Testanlage

Bildquelle: © Hochschule Fulda

Bild 4: Projektierungssicht des Testaufbaus in IndraWorks.

Die Projektierung der Testanlage ist bei der Entwicklung einer Motion-Control-Applikation immer der erste Schritt, dem im zweiten Schritt die Softwareprogrammierung folgt (Bild 4). Die Projektierung der Testkonfiguration ist denkbar einfach. Da die SMC3-Antriebe von IndraWorks über die Funktion ‚Buskonfiguration scannen' entsprechend als Sercos-Drives erkannt werden, lassen sie sich als solche in das Projekt einfügen. Weitere Einstellungen, wie beispielsweise die Einstellung der Mikroschritte, des Phasenstroms oder der Motorschrittauflösung können einerseits über IndraWorks erfolgen, alternativ dazu aber auch sehr einfach und programmatisch aus Matlab heraus.

Die Definition der Testfälle orientiert sich an den Funktionalitäten für Motion-Control, die die mlpi4Matlab-Toolbox zur Verfügung stellt. Diese gliedern sich in Konfigurations- und Bewegungsfunktionen. Letztere waren der zentrale Gegenstand der Untersuchung. Von diesen über 30 Funktionen sind einige nicht relevant für den SMC3, da dieser Antrieb beispielsweise den drehmomentgeregelten Betrieb nicht unterstützt; wohl aber die Positions- und Geschwindigkeitsregelung sowie die Übertragung von zyklischen Geschwindigkeits- oder Lage-Sollwerten. Neben den Konfigurationsfunktionen sind für den Einzelachs- und synchronen Multi-Achsbetrieb die folgenden Funktionen am wichtigsten:

- m4MMotionFlexProfile, m4MMotionSynchronOut, m4MMotionGetAxisValues,
- m4MMotionGearIn, m4MMotionGearInPos,
- m4MMotionMoveAbsolute, m4MMotionMoveAdditive, m4MMotionMoveRelative,
- m4MMotionMoveVelocity, m4MMotionPower, m4MMotionStop,
- m4MMotionWriteCyclicPosition, m4MMotionWriteCyclicVelocity.

Das gesamte Testszenario umfasst insgesamt etwa 60 Einzeltests, die ihrerseits abgeschlossene Testfälle für beide Achsen umfassen.

Im Allgemeinen sind für den Anwender die Funktionen mit selbstredenden Funktionsnamen die einfachste und schnellste Möglichkeit der Erstellung von Programmen in Matlab. Die Antriebsregler umfassen allerdings zusätzlich eine Vielzahl von Einstellparametern, die zwar

nicht direkt über dedizierte Funktionsnamen aufrufbar, jedoch über den Zugriff auf Standard-Sercos- (S-) oder produktspezifische (P-) Parameter zugänglich sind.

Für das Testszenario bedeutet dies, dass über die m4MParameter-Funktionen auch der Parameterzugriff getestet werden muss. Dies kann in einem eigenen Test erfolgen; allerdings sind für die Grundeinstellung des SMC3 zur Anpassung an die Motoren bereits diese Funktionen erforderlich. Für fundamentale Funktionen, wie beispielsweise das Phasenschalten, gilt dies ebenfalls. Einige der S- und P-Parameter sind in der Kommunikationsphase 4 (CP4) der Sercos-Kommunikation schreibgeschützt, so dass die Matlab-Funktionen für das Phasenschalten zwischen der Kommunikationsphase 2 (CP2) und CP4 integraler Bestandteil sind.

Entsprechend der Systemkonfiguration wurde schließlich ein Testmanager – der sogenannte mlpi4SMC3TestBuddy – entwickelt, der die Durchführung beliebig ausprogrammierter Einzeltests übernimmt. Gemeinsam ist den Einzeltests lediglich eine Schnittstellendefinition für die Rückgabewerte der Einzeltests und der Aufrufparameter. Ein Beispiel für einen Testaufruf ist:

```
>> m4MMotionFlexProfile_smc3('192.168.178.123','FlexTest.log')
```

In diesem Test wird eine FlexProfile-Kurvenscheibe spezifiziert, dem Motion-Controller zur Prüfung übermittelt und anschließend ausgeführt. Für die Ausführung sind eine Master-Achse und eine Slave-Achse zu definieren und mit der Funktion m4MMotionFlexProfile zu synchronisieren. Anschließend wird die Master-Achse in Positions- oder Geschwindigkeitsregelung kommandiert, der die Slave-Achse bei sinnvoller Kurvenscheibenparametrierung folgen kann. Die Achsen müssen anschließend wieder gestoppt und absynchronisiert werden. Der Vorgang beinhaltet zusätzlich noch das Leistungsschalten und weitere Bewegungskommandos.

Das Wichtige an dem Vorgehen ist, dass die Tests eine möglichst geringe Komplexität aufweisen. Dadurch erfüllen sie nicht nur ihre eigentliche Aufgabe der applikationsnahen Prüfungsdurchführung, sondern sind gleichzeitig ein Programmier-Template für Anwender. Insgesamt wurden für etwa 60 Funktionen Einzeltests programmiert. Mit Hilfe dieser final - erfolgreich durchgeführten Tests ließ sich schließlich nachweisen, dass zahlreiche Bewegungsfunktionen des Motion-Controllers mit dem verwendeten Schrittmotorantriebsregler realisierbar sind.

Aus Matlab heraus automatisiert C-Code erzeugen

Aufgrund der Tatsache, dass Matlab auf einem Betriebssystem ausgeführt wird, das keine Echtzeit-Eigenschaften im Sinne von Motion-Controllern aufweist, sind die programmierten Einzeltests auf den ersten Blick nicht echtzeitfähig. Dennoch lassen sich synchrone Achsbewegungen programmieren, wenn Kurvenscheibenfunktionen oder elektronische Getriebe genutzt werden. Auch der Positionierbetrieb, in dem der Motion-Controller für eine Bewegung in Echtzeit sorgt, ist leicht testbar. Daneben ist die mlpi4Matlab-Software in großen Teilen Coder-fähig. Das heißt: Innerhalb eines Einzeltests können Code-Generierungsprozeduren eingebaut werden, die automatisiert aus Matlab-Code einen C-Code generieren, diesen compilieren, auf das Target übertragen und dort ausführen. Das kann automatisiert und ohne Eingriffe des Anwenders erfolgen und ist zudem frei programmierbar. Als weitere Alternative kann aus Matlab heraus ein PLC-Coder aufgerufen werden, der dann entsprechenden IEC-Quellcode für die SPS der MLC generiert.

Zusammenfassend lässt sich festhalten: Insbesondere wenn Anwendungen nur kleine Motoren benötigen, die Kostenanforderungen weit unterhalb von typischen Servomotoren haben und gleichzeitig nicht deren Dynamik erfordern, sind Antriebsregler mit Schrittmotoren am Automatisierungsbus Sercos eine überlegenswerte Alternative für Motion-Control-Anwendungen. Zwar werden solche Applikationen im Allgemeinen in den Programmiersprachen der IEC 61131-3 programmiert. Prototypische Anwendungen oder Aufgaben, die geringere Echtzeit-Anforderungen haben, lassen sich per Rapid Control Prototyping jedoch deutlich schneller und effizienter programmieren.

Die Ursachen liegen ganz einfach darin, dass bei klassischer IEC-Programmierung bei Programmänderungen diese üblicherweise neu übersetzt und auf den Motion-Controller übertragen werden müssen. Die Tatsache, dass die Funktionalität des Online-Change deutlich schneller ausgeführt werden kann als ein kompletter Download, ändert daran nur wenig. Ein weiterer Effizienzvorteil sind die Programm-Manipulationsmöglichkeiten beim Debugging von Software. Zwar lassen sich IEC-Programme ebenfalls mit gängigen Methoden debuggen, also Haltepunkte setzen, Variablenwerte analysieren oder modifizieren: das Einfügen oder Ausführen von Quellcode in Form von Funktionsaufrufen oder Skripten oder Typänderungen von Variablen während des Debuggens sind so aber nicht möglich. All diese Punkte gehören zur Vorgehensweise beim explorativen Programmieren und sind insbesondere beim Prototyping von Software oftmals von Vorteil.

In Vorentwicklungen wurden bereits Roboteranwendungen mit der beschriebenen Systemkonfiguration programmiert, so dass die Erweiterung auf die Robotik ein naheliegender nächster Schritt ist.

Autor:
Prof. Elmar Engels verantwortet das Lehrgebiet Automatisierungstechnik und Systemtechnik an der Hochschule Fulda.