Model-Based-Design vereint Regelung und Mechatronik

Entwickeln, Simulieren und Implementieren von Automatisierungssystemen. In der Automatisierungstechnik werden Regelalgorithmen normalerweise implementiert und getestet, indem sie auf einer speicherprogrammierbaren Steuerung (SPS) ausgeführt werden, die mit der Maschine verbunden ist, für die der Algorithmus entwickelt wird. Ein großer Nachteil dieser Methode besteht darin, dass es kostspielig, schwierig oder gefährlich sein kann, die Regelstrategie gleich im ersten Durchlauf an der Maschine zu testen. Es ist effizienter, die Regelung zunächst mithilfe eines Maschinenmodells und einer entsprechenden Simulation zu entwickeln und zu testen.

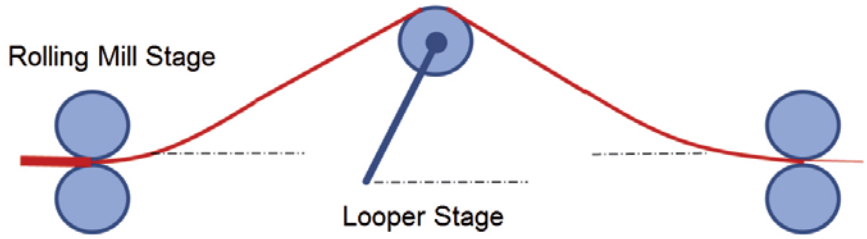

Bild 1: Schematische Darstellung einer Walzstraße

Bill Chou

In der Simulation von Maschinenmodellen können Probleme in einem frühen Stadium des Entwicklungsprozesses erkannt und die Zahl der Fehler reduziert werden, die zu einem späteren Zeitpunkt im Prozess auftreten können und deren Behebung sich dann als kostspielig und schwierig erweist. Wenn die SPS anschließend zu Test- und Validierungszwecken mit der Maschine verbunden wird, können sich die Entwickler eher darauf verlassen, dass das System erwartungsgemäß reagiert.

Model-Based-Design

Aufgrund der Einschränkungen, die mit herkömmlichen Ansätzen bei der Entwicklung von Regelungen verbunden sind, verwenden viele Entwickler das Model-Based-Design, um Modelle auf Systemebene zu entwickeln und zu simulieren, die die Dynamik eines Regelungssystems abbilden. Das Simulationsmodell stellt eine Verbindung zwischen der Regelstrategie und den mechanischen, elektrischen und hydraulischen Komponenten einer Maschine her. Mithilfe der Simulation können Entwickler Testfälle ausführen und Entwicklungs- und Integrationsfehler frühzeitig während des Entwicklungsprozesses erkennen. Wenn das Projekt von der Entwicklungs- in die Implementierungsphase übergeht, können Sie durch manuelles Programmieren entstehende Fehler reduzieren, indem Sie direkt aus dem Regelungsmodell heraus strukturierten Text erstellen und diesen als Add-On-Anweisung (AOI, Add-on-Instruction) in RSLogix importieren. Die Nutzung des Model-Based-Designs erleichtert die Änderung und Aktualisierung der Regelstrategie, da das Modell so schneller aktualisiert, Tests erneut ausgeführt, strukturierter Text neu generiert und die aktualisierte AOI in RSLogix importiert werden kann.

Modellieren einer Stahlwalzstraße

Um zu erläutern, wie das Model-Based-Design in der Praxis angewendet wird, wird im Folgenden ein Projekt erläutert, in dem eine Regelstrategie für eine Stahlwalzstraße entwickelt werden soll. In diesem Stahlwalzwerk wird aus einer Stahlbramme ein Stahlblech von gleichmäßiger Dicke hergestellt. Die Walzstraße besteht in der Regel aus mehreren Walzgerüsten, in denen die Walzen den Stahl komprimieren, der die Walzen passiert. Zwischen den Walzgerüsten befinden sich Schlingenheber, um die Blechspannung aufrechtzuerhalten und um Abreißen und Durchhängen zu vermeiden.

Modellieren einer einstufigen Walzstraße

Um ein vollständiges mehrstufiges System zu modellieren, kann man zunächst damit beginnen, einstufige Walzstraßen zu modellieren, die aus einem Walzgerüst und einem

KONTAKT

MathWorks
Adalperostraße 45
85737 Ismaning
Tel.: +49 89 45235-6700
Fax: +49 89 45235-6710
www.mathworks.de

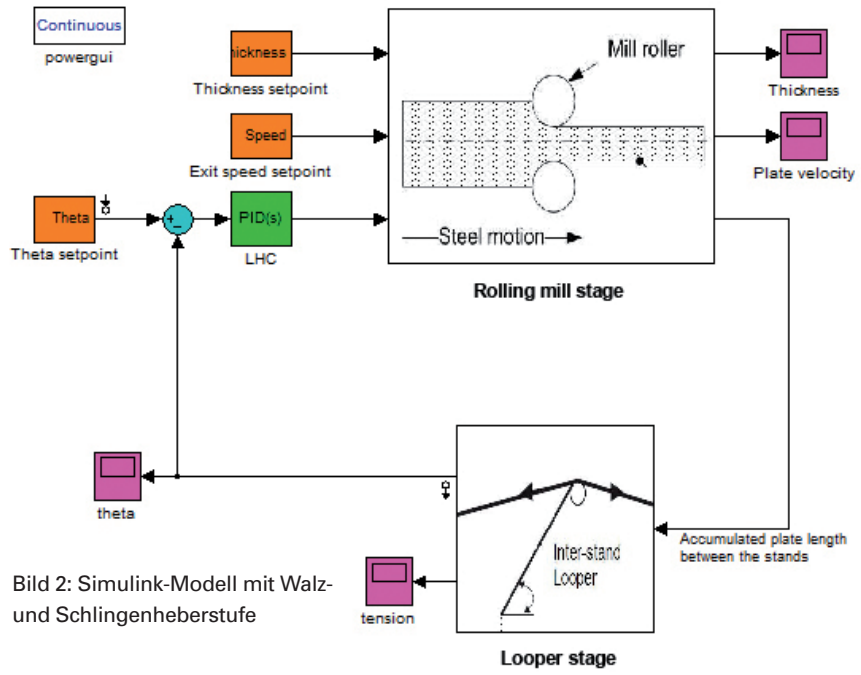

Bild 2: Simulink-Modell mit Walz- und Schlingenheberstufe

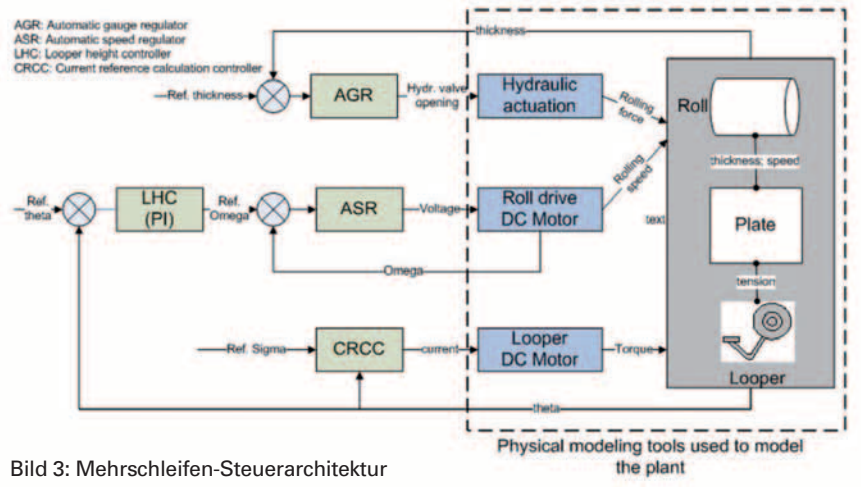

Bild 3: Mehrschleifen-Steuerarchitektur

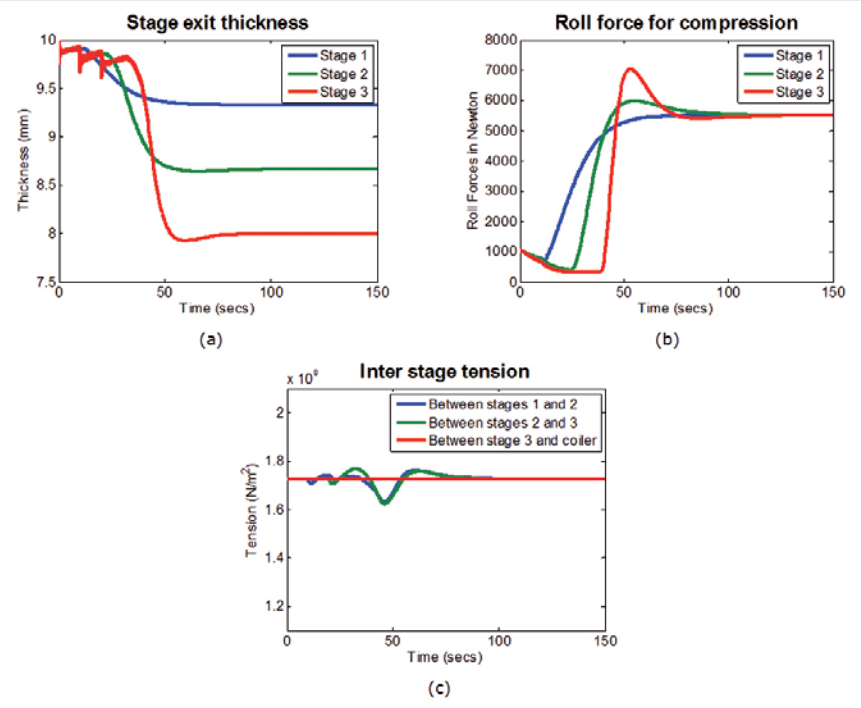

Bild 4: Simulationsergebnisse für die Prozessvariablen (a) Das Ziel, die Gesamtdicke zu reduzieren, wurde am Ende von Stufe 3 erreicht. (b) Das Ziel, die Gesamtdicke zu reduzieren, wurde gleichmäßig auf alle drei Stufen verteilt. (c) Abweichungen der Blechspannung wurden ausgeglichen.

Schlingenheber bestehen. In dem Walzgerüst wird ein Hydraulikaktuator verwendet, um eine Walzdruckkraft zu erzeugen, die die Stahlbramme komprimiert. Das Walzdrehmoment, das von einem Gleichstrommotor erzeugt wird, regelt die Walzgeschwindigkeit. Die mechanischen, elektrischen und hydraulischen Komponenten der Walze lassen sich in Simulink modellieren. Der Schlingenheber, das Stahlblech vor dem Schlingenheber und das Stahlblech nach dem Schlingenheber werden als drei Module modelliert, die durch Konnektoren miteinander verbunden sind.

Modellieren der Regelung für eine einstufige Walzstraße

Als nächstes werden die Regelung für eine einstufige Walzstraße entwickelt und modelliert. Bild 3 zeigt einen typischen Aufbau mit vier Kompensatoren. Der automatische Messregler/Kompensator (AGR, Automatic Gauge Regulator) weist die Öffnung der Hydraulikventile an, die Walzdruckkraft zu erzeugen, die zur Regelung der Blechdicke dient. Der automatische Geschwindigkeitsregler/Kompensator (ASR, Automatic Speed Regulator) ermittelt die Spannung für den Gleichstrommotor, der das Walzdrehmoment erzeugt und somit die Blechgeschwindigkeit regelt. Der Kompensator zur Regelung der Höhe des Schlingenhebers (LHC, Looper Height Control) erzeugt die Drehzahlreferenz für die Walze, um die gewünschte Materialspannung zu erhalten. Als letztes Element stellt der Kompensator zur Regelung der Stromreferenzberechnung (CRCC, Current Reference Calculation Controller) den Strom zum Gleichstrommotor des Schlingenhebers, um diesen so auszurichten, dass die gewünschte Materialspannung beibehalten wird. Alle Regelkreise sind miteinander gekoppelt; der Hydraulikaktuator, der vom AGR-Kompensator geregelt wird, steuert sowohl die Blechdicke als auch die -geschwindigkeit, während die LHC- und ASR-Kompensatoren so miteinander verschaltet sind, dass sie die erforderliche Spannung und Geschwindigkeit des Stahlblechs aufrechterhalten.

Im nächsten Schritt werden die AGR- und ASR-Kompensatoren ausgelegt. Zunächst erfolgt die Linearisierung des nichtlinearen Streckenmodells, anschließend die Herleitung der optimalen Kompensatorkoeffizienten. Nachfolgend lassen sich diese Werte überprüfen, indem mit den hergeleiteten Kompensatorkoeffizienten Simulationen der Regelung und nichtlinearen Streckenmodellen der einstufigen Straße ausgeführt werden. In diesem Beispiel ist der LHC-Kompensator als proportionales integrales (PI) Regelungsglied modelliert. Das PID-

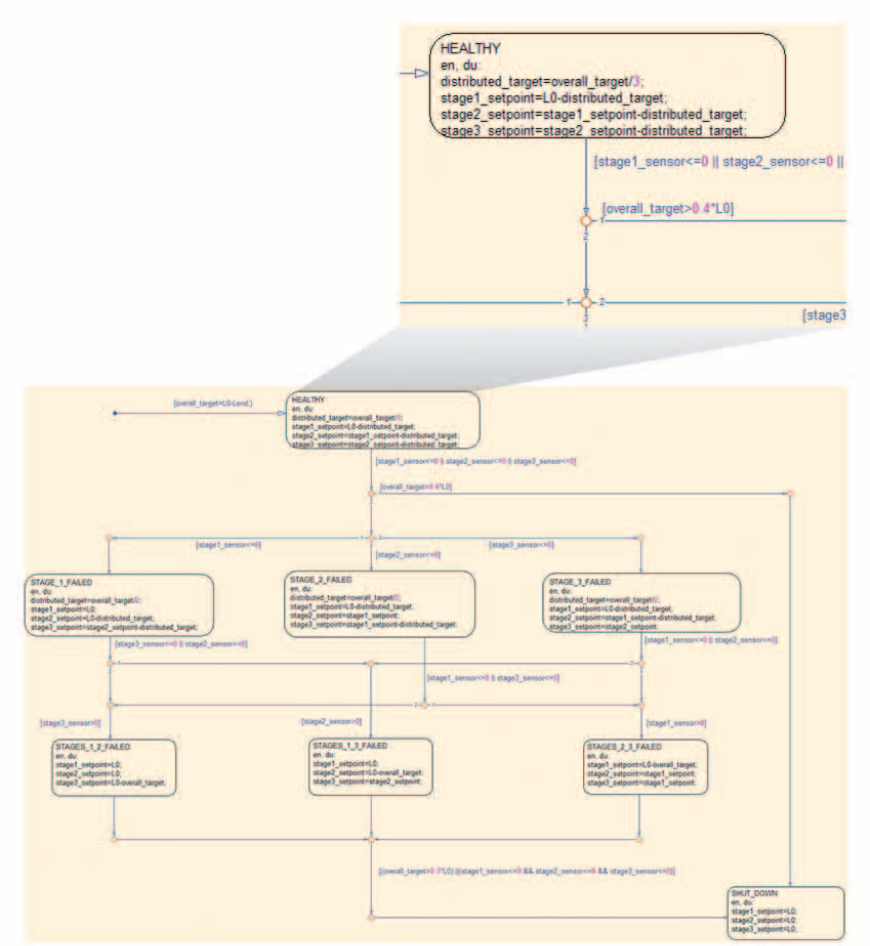

Bild 5: Teil einer fehlertoleranten Sollwertverteilung, implementiert in Stateflow

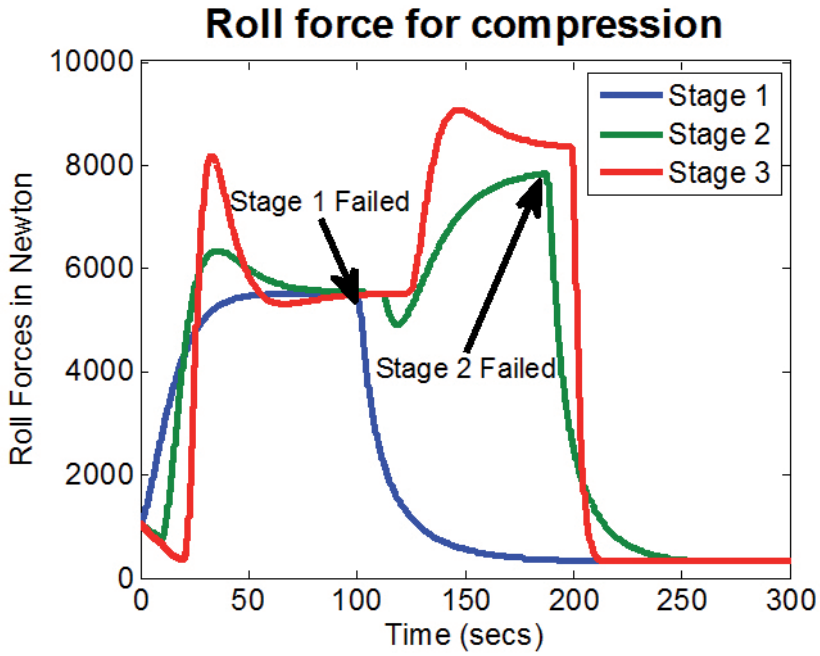

Bild 6: Simulationsergebnisse, die die Systemwiederherstellung ausgehend von Funktionsstörungen in den einzelnen Stufen anzeigen: Stufen 2 und 3 kompensiert, wenn Stufe 1 ausgefallen ist. Wenn Stufe 2 ausfällt, wird das System heruntergefahren, da das Ziel, die Gesamtdicke zu reduzieren, nicht allein durch Stufe 3 erreicht werden konnte.

Tuningtool kann verwendet werden, um die PID-Verstärkung für den Kompensator automatisch basierend auf Zeit- und Frequenzbereichsanforderungen zu berechnen. Abschließend sollte der Gesamtentwurf getestet werden, indem eine nichtlineare Simulation der Walzstraße mit den hergeleiteten Koeffizienten ausgeführt wird.

Modellieren eines mehrstufigen Prozesses

Die Modelle für die einstufige Walzstraße und den Schlingenheber lassen sich auch als Komponenten für das mehrstufige Prozessmodell wieder verwenden. An diesem Punkt können auch zusätzliche Komponenten hinzugefügt werden, wie Massenerhaltung und Transportverzögerungen in verschiedenen Walzstufen. Die Simulation kann die Prozessvariablen in den drei Phasen des Prozesses anzeigen. Die Sollwerte für Dicke und Geschwindigkeit in jeder einzelnen Stufe wurden erreicht, sodass Stahlbleche mit gewünschter Dicke und Durchsatz am Ende der dritten Stufe hergestellt werden können (rote Kurve in Bild 4). Abweichungen hinsichtlich der Blechdicke in den verschiedenen Stufen wurden ebenso erfolgreich ausgeglichen. Zu beachten ist, dass das Werksmodell für zwei Zwecke verwendet wurde. Das Modell wurde zunächst linearisiert und anschließend verwendet, um die Regler einzustellen. Dann wurden Simulationen mit dem vollständigen, nichtlinearen Streckenmodell für verschiedene Testfälle ausgeführt, um die Auswirkungen in verschiedenen Szenarien zu analysieren.

Entwickeln und Prüfen der Logik zur Fehlererkennung

Zusätzlich zur Entwicklung der Kompensatoren muss man die Logik zur Fehlererkennung in der SPS entwickeln, um die Sensoren und Aktoren zu überwachen und zu verwalten. Wenn die SPS einen Fehler erkennt, sollte die Störung auf eine spezifische Komponente zurückzuführen sein, um diese so zu beheben, dass der normale Betrieb nicht unterbrochen wird. Wenn die Störung nicht behoben werden kann, sollte die SPS die Walzstraße sicher herunterfahren. Dieses Beispiel geht auf die Logik zur Fehlererkennung ein, die Störungen an den Hydraulikventilen erkennt und entsprechende Gegenmaßnahmen einleitet (Bild 5). Insbesondere verteilt die Logik das Ziel, die Gesamtdicke zu reduzieren, auf die einzelnen Sollwerte für die Dicke, die in einem mehrstufigen Prozess zu finden sind. Wenn die hydraulische Komprimierung in einer

Stufe ausfällt, werden die Sollwerte zur Reduzierung der Dicke in den anderen beiden Phasen neu berechnet, um das Ziel, die Gesamtdicke zu reduzieren, zu erreichen.

Die Logik lässt sich testen, indem bewusst Fehler in das Modell eingebaut werden. Bild 6 zeigt die Simulationsergebnisse der fehlertoleranten Logik. Wenn eine Stufe ausfällt, prüft die Überwachungs-SPS, ob die Charge auf die restlichen, intakten Phasen verteilt werden kann. Wenn dies der Fall ist, werden die Sollwerte für die Reduzierung der Dicke an die AGRs der einzelnen Phasen gesendet. Andernfalls wird der Prozess heruntergefahren, indem die Blechbewegung gestoppt wird.

Implementieren von strukturiertem Text auf einer SPS

Das automatische Erzeugen von strukturiertem Text ist der nächste logische Schritt nach der Überprüfung des Entwurfs. Diese Funktion reduziert das Auftreten von Fehlern, die durch manuelles Programmieren entstehen können und gewährleistet, dass der geprüfte strukturierte Text zu numerischen Ergebnissen auf der Allen-Bradley-SPS führt, die den Simulationsergebnissen entsprechen. Wenn das gleiche Modell verwendet wird, das auch für die Simulation und den Test Anwendung fand, kann automatisch strukturierter Text für den Steuerungsalgorithmus erzeugt und dieser Text in RSLogix importiert werden, wo es anschließend die Möglichkeit gibt, ihn zu kompilieren und entweder auf RSLogix Emulate oder die Allen-Bradley-SPS herunterzuladen. Bild 7 zeigt strukturierten Text nach IEC 61131 an, der von einer Logik zur Fehlererkennung und -behebung erzeugt wurde. Der generierte strukturierte Text ist ausführlich kommentiert und kann leicht zum Modell zurückverfolgt werden. Simulink PLC Coder kann ebenso eine Testbench generieren, mit der überprüft werden kann, ob die Simulationsergebnisse mit den Ergebnissen in RSLogix Emulate übereinstimmen.

Zusammenfassung

Das Model-Based-Design ermöglicht es Entwicklern, vollständige Systeme zu simulieren und diese in einem frühen Stadium des Entwicklungszyklus zu überprüfen. Das automatische Erzeugen von strukturiertem Text ausgehend vom selben Modell vermeidet solche Fehler, die beim manuellen Programmieren entstehen können. Wenn Entwickler den Entwurf mithilfe einer Simulation prüfen und anschließend die

```
IF ((stage1_sensor <= 0) OR (stage2_sensor <= 0)) OR (stage3_sensor <= 0) THEN
    (* Transition: '<S1>:18' *)
    IF overall_target > (0.4 * L0) THEN
        (* Transition: '<S1>:95' *)
        (* Transition: '<S1>:146' *)
        (* Exit 'HEALTHY': '<S1>:10' *)
        (* Entry 'SHUT_DOWN': '<S1>:33' *)
        is_c1_Setpoint := Setpoint_IN_SHUT_DOWN;
        rtb_stage1_setpoint := L0;
        rtb_stage2_setpoint := L0;
        distributed_target := L0;
    ELSE
        (* Transition: '<S1>:139' *)
        IF stage1_sensor <= 0 THEN
            (* Transition: '<S1>:41' *)
            (* Transition: '<S1>:131' *)
            (* Exit 'HEALTHY': '<S1>:10' *)
            (* Entry 'STAGE_1_FAILED': '<S1>:12' *)
            is_c1_Setpoint := Setpoint_IN_STAGE_1_FAILED;
            distributed_target := overall_target / 2;
            rtb_stage1_setpoint := L0;
            rtb_stage2_setpoint := L0 - distributed_target;
            distributed_target := rtb_stage2_setpoint - distributed_target;
        ELSIF stage3_sensor <= 0 THEN
            (* Transition: '<S1>:42' *)
            (* Transition: '<S1>:129' *)
            (* Exit 'HEALTHY': '<S1>:10' *)
            (* Entry 'STAGE_3_FAILED': '<S1>:23' *)
            is_c1_Setpoint := Setpoint_IN_STAGE_3_FAILED;
            distributed_target := overall_target / 2;
            rtb_stage1_setpoint := L0 - distributed_target;
            rtb_stage2_setpoint := rtb_stage1_setpoint - distributed_target;
            distributed_target := rtb_stage2_setpoint;
        ELSIF stage2_sensor <= 0 THEN
            (* Transition: '<S1>:22' *)
            (* Exit 'HEALTHY': '<S1>:10' *)
            (* Entry 'STAGE_2_FAILED': '<S1>:21' *)
            is_c1_Setpoint := Setpoint_IN_STAGE_2_FAILED;
            distributed_target := overall_target / 2;
            rtb_stage1_setpoint := L0 - distributed_target;
            rtb_stage2_setpoint := rtb_stage1_setpoint;
            distributed_target := rtb_stage1_setpoint - distributed_target;
        ELSE
            guard := TRUE;
        END_IF;
    END_IF;
ELSE
    guard := TRUE;
```

Comments with references to the Stateflow chart

Bild 7: Strukturierter Text nach IEC 61131, übertragen in Add-On-Anweisungen (AOI), mithilfe von Simulink PLC Coder generiert

Implementierung dieses Entwurfs automatisieren, können sie sich darauf verlassen, dass das vollständige System erwartungsgemäß reagiert, wenn die SPS zum ersten Mal an die Maschine angeschlossen wird. *(wp)* ■

www.mathworks.de

Autor:
Bill Chou ist Product Marketing Manager bei MathWorks