PC-basierte Automatisierung in der Praxis, Teil 3:

Programmierung & Soft-SPS

Die vorliegende Artikelreihe befasst sich mit PC-basierter Automatisierung. In vier Folgen wird der heutige Stand der Automatisierungspraxis mit Industrie-PCs beleuchtet:
Teil 1: Entstehung, Grundlagen und Motivation
Teil 2: Bauformen, Vermarktung und Interoperabilität
Teil 3: Programmierung und Soft-SPS
Teil 4: Leistungsdaten

Die moderne Automatisierung ist ohne SW-Engineering nicht mehr denkbar. Zum einen werden immer mehr Funktionen quasi «gratis» mittels Programmbausteine realisiert (z. B. Regler, Antriebstechnik), zum anderen werden die zu steuernden Anlagen und Maschinen komplexer, was zwangsläufig zu einem erhöhten Programmieraufwand führt. In der Tat ist in den letzten Jahren der Anteil der Software-Entwicklung in einem Automatisierungsprojekt zu Lasten der anderen Engineering-Leistungen wie z. B. dem mechanischen Design kontinuierlich gestiegen. Kein Wunder also, dass die Nachfrage nach immer effizienteren Programmiermethoden wächst.

Programmierung – vom Kontaktplan zur Web-fähigen Automatisierung

Auf dem Gebiet der Programmierung hat eine dynamische Evolution von der reinen Hardware-Verdrahtung über die klassischen SPS-Sprachen wie Kontaktplan, Funktionsplan und Anweisungsliste bis hin zu höheren Programmieransätzen wie Zustandsgraphen, Schrittketten, Technologieplänen und Hochsprache stattgefunden.
Betrachtet man einen typischen Projektablauf, können drei Phasen unterschieden werden: Spezifikation, Entwurf oder Design und Realisierung. In der Spezifikationsphase werden Funktion und Verhalten von Anlagen und Maschinen definiert. Das klassische Mittel hierzu ist das Pflichtenheft.
In verbaler Form wird das Vorhaben skizziert und Eckpunkte festgeschrieben. Abläufe werden grafisch in Diagrammen verständlich gemacht. An dieser allgemeinverständlichen Form hat sich bis heute nichts geändert; schon aus dem Grund, weil zum Teil auch technisch weniger versierte Entscheidungsträger zustimmen müssen.

Wandel der Entwurfsmethoden

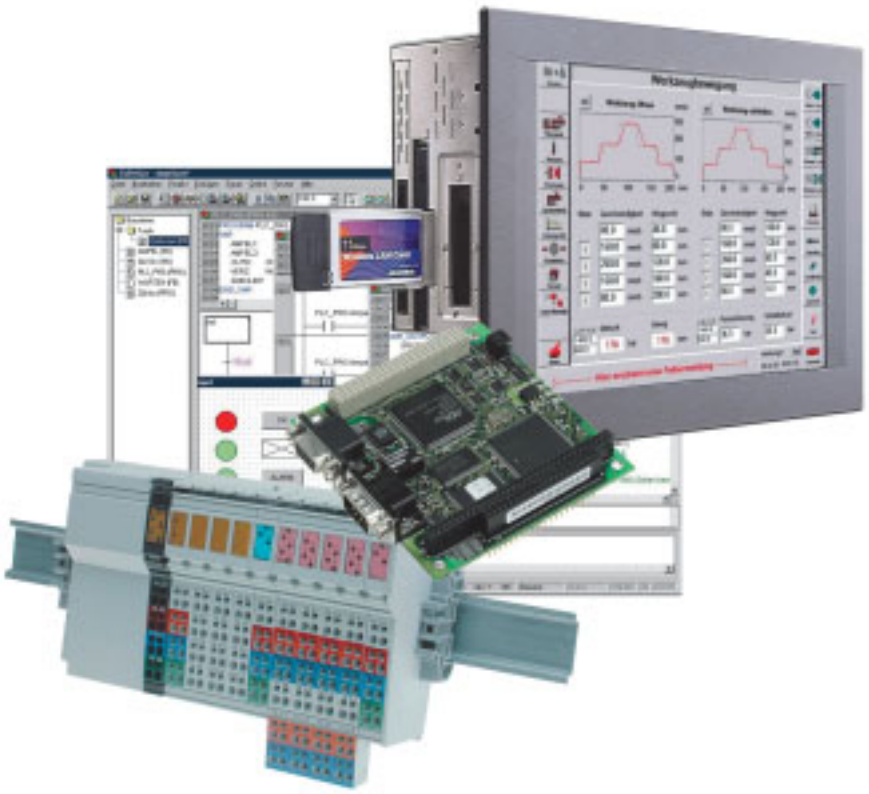
Anders sieht es in der Entwurfs- und Realisierungsphase aus. Hier hat sich ein lebhafter Wandel der Entwurfsmethoden vollzogen. Anfangs wurde in der Relaistechnik die Funktion mit Hilfe von Blockdiagrammen und Schaltplänen entworfen, die dann fest in Hardware (Relais/Schütz) verdrahtet wurde. Diese Vorgehensweise war natürlich zeitintensiv, fehlerträchtig und wenig flexibel.
Mit dem Aufkommen der SPS-Technik entwickelte sich aus der Entwurfsmethode «Schaltplan» der Kontaktplan. Die Kontaktplan-Programmierung kam dem Ausbildungsstand der Zielgruppe entgegen, welche hauptsächlich aus Industrieelektrikern bestand. Für komplexere Aufgabenstellungen ist der Kontaktplan in seiner Reinform jedoch ungeeignet. Deshalb wurde gemäss dem Vorbild Blockdiagramm der Funktionsplan kreiert.
Im Laufe der Zeit sind auch unterschiedlichste Mischformen entstanden. Heute ist die Grenze zwischen Kontaktplan und Funktionsplan fliessend. Als neue Errungenschaft führte die SPS-Technik die Anweisungsliste in die Automatisierung eine – eine Art Assemblerprogrammierung, zugeschnitten auf die eher binäre Welt der Steuerungstechnik. Diese hatte mit Schaltplan und Blockdiagramm nichts mehr gemeinsam und eröffnete neue Freiheitsgrade in der Programmierung und ermöglichte eine effizientere (und damit schnellere) Programmierung. Eine neue Generation von Anwendern entstand: der SPS-Programmierer.

«Handarbeit» bei typischen SPS-Sprachen

Bei den typischen SPS-Sprachen Kontaktplan, Funktionsplan und Anweisungsliste ist noch viel «Handarbeit» notwendig. Der Programmierer muss neben den gewünschten Funktionen oder Abläufen auch deren Umsetzung auf unterster Ebene realisieren. Abläufe werden also durch eine clevere Verschaltung von Kontakten und Zeitgliedern generiert (Kontaktplan) oder durch eine gekonnte Abfrage von Eingangssignalen und Setzen von internen Merkern (Anweisungsliste).
Eine Maschine oder Anlage kann man jedoch auch als ein System mit Zu-

Autor
Peter Steib
European Channel Partner Manager, eAutomation bei Advantech, Meyriez (Schweiz)

Vertriebsgesellschaft Schweiz:
Omni Ray AG, 8600 Dübendorf
Tel. 044 802 28 80, info@omniray.ch



ständen und Zustandswechseln (Transitionen) auffassen. Bei diesem Modell verharrt das System in einem Zustand, bis es durch ein Trigger-Ereignis in einen neuen Zustand wechselt. Die Weiterschaltbedingung kann hierbei durch ein externes Signal oder durch einen internen Ablauf erfolgen. Dieses Modell eignet sich sehr gut zur Beschreibung von Automaten also Maschinen und Anlagen. Auf dieser Sichtweise basiert die Programmierung mit Zustandsgraphen.

Der Programmierer braucht lediglich Funktion und Ablauf programmieren – um Bits und Bytes muss er sich nicht mehr kümmern. Die Programmierung wird so auf eine neue Ebene der Abstraktion gehoben und dadurch effizienter. Allerdings beanspruchen solche abstrakteren Programmiermethoden mehr Ressourcen (Speicher, Verarbeitungsleistung), weshalb sie bedingt durch die Einschränkungen der SPS-Technik wenig eingesetzt wurden.

Effizienz der Programmierung

Ähnlich wie die SPS mit der Kontaktplanprogrammierung sich an die «alte» Methode der Hardware-Verdrahtung anpasste, schlägt heute die PC-Technik mit der Soft-SPS die Brücke zur SPS-Welt. Dank der Tatsache, dass man sich in der PC-Technik nicht mehr mit Ressourcen-Knappheit herumschlagen muss, ist der Abstrahierungslevel und damit die Effizienz der Programmierung weiter gestiegen. Anstelle der Anweisungsliste kann jetzt Hochsprache eingesetzt werden, aus Kontakt- und Funktionsplan ist der Technologieplan geworden. Zudem können die bewährten SPS-Sprachen auf PCs weiterhin eingesetzt werden.

Uneingeschränkte Kommunikation

Der grösste Vorteil der PC-basierten Automation ist ihre beispiellose Integrationsfähigkeit. Neben der reinen Steuerungsfunktion lassen sich problemlos weitere Funktionen wie Datenbanken, Visualisierung und Vernetzung verwirklichen. Die Realisierung aller Anforderungen einer Applikation in einem System gewährleistet einen optimalen Datenaustausch zwischen den Einzelfunktionen ohne Systembruch und macht das Leben einfacher.

Gerade auf dem Gebiet der Kommunikation kann die PC-basierte Automation ihre Stärken voll ausspielen. Sei es über Telefon(-modem), E-Mail oder Web-Server, PC-basierte Steuerungssysteme lassen sich ohne Einschränkungen sowohl lokal als auch weltweit vernetzen.

Soft-SPS – Betriebssystem im Betriebssystem

Die Qualität einer PC-basierten Automatisierungslösung hängt im Wesentlichen von zwei Faktoren ab:

- Betriebssystem
- Implementierung der Soft-SPS

An dieser Stelle soll nur auf die gängigen Microsoft-Betriebssysteme eingegangen werden, da diese am weitesten verbreitet und akzeptiert sind. Der Einsatz anderer, zum Teil sehr stabiler Betriebssysteme wie Linux oder spezieller Echtzeit-Systeme kann in Bezug auf die Verfügbarkeit von Runtime-Modulen, Treibern und Software-Anwendungen Schwierigkeiten bereiten und ist daher im Einzelfall zu prüfen.

Um es vorweg zu nehmen: die älteren Windows-Betriebssysteme Windows 95/98/Me hatten ein Problem. So kann es bei diesen Betriebssystemen vorkommen, dass die internen Windows-Ressourcen nach und nach in Beschlag genommen und nicht mehr freigegeben werden. Am Ende sind alle Ressourcen aufgebraucht, was einen Neustart notwendig macht. Dies ist bei Büro-PCs kein Problem, da diese in der Regel sowieso jeden Tag neu gestartet werden.

Bei industriellen Anwendungen, die im Dauerbetrieb verfügbar sein müssen, ist ein solches Verhalten jedoch nicht akzeptabel. Bei der neuen Windows-Generation NT/2000/XP ist dies nicht mehr der Fall, stammt die zu Grunde liegende Technologie ja aus dem Server-Bereich, bei dem ein stabiler Dauerbetrieb das A und O ist.

Einbindung der Soft-SPS

Wesentlicher Einfluss hat die Art und Weise, wie eine Soft-SPS in ein Windows-System eingebunden ist. Ist diese z. B. als normale Applikation implementiert, wird deren Abarbeitung von Windows gesteuert. Ein anderer Weg besteht in der Implementierung der Soft-SPS in Form eines Kernel-Mode-Treibers, welcher mittlerweile von den meisten Soft-SPS-Anbietern beschritten wird.

Daher empfiehlt sich bei Automatisierungsanwendungen auf die neueren Windows-Betriebssysteme zu setzen.

Integration der Soft-SPS

Wesentlicher Einfluss hat auch die Art und Weise, wie eine Soft-SPS in das Windows-System eingebunden ist. Ist diese z. B. als normale Applikation implementiert, wird deren Abarbeitung von Windows gesteuert. Nun ist Windows auch heute noch kein dediziertes Echtzeitbetriebssystem. Dementsprechend sind Mechanismen zur Taskverwaltung und Priorisierung nur in eingeschränktem Masse vorhanden.

Man darf daher die Ansprüche in Bezug auf das Echtzeitverhalten nicht zu hoch ansetzen. Ausserdem besteht die Gefahr, dass bei einem Systemabsturz (Windows) auch die Soft-SPS und damit die Steuerungsfunktion in Mitleidenschaft gezogen wird. Anders verhält es sich mit Windows CE, welches speziell für embedded Applikationen entwickelt wurde und zuverlässige Echtzeiteigenschaften vorweisen kann.

Kernel-Mode-Treiber

Ein anderer Weg besteht in der Implementierung der Soft-SPS in Form eines Kernel-Mode-Treibers, welcher mittlerweile von den meisten Soft-SPS-Anbietern beschritten wird. Hierbei wird die Soft-SPS wie ein Hardware-Treiber direkt dem Timer-Interrupt des PCs zugeordnet. Der Timer-Interrupt ruft die Soft-SPS auf, die dann das Steuerungsprogramm abarbeitet. Danach gibt die Soft-SPS an Windows ab.

In diesem Fall ist der Aufruf der Soft-SPS nicht mehr von Windows abhängig – vielmehr ruft nach getaner Arbeit die Soft-SPS Windows auf. Da der Timer-Interrupt «hochprior» ist, unterbricht dieser immer das Windows-Betriebssystem – selbst wenn dieses abgestürzt sein sollte. Die Steuerungsfunktion ist also auch noch bei einem «Blue-Screen» gewährleistet.

Eigenständiges Multitasking-Betriebssystem

Die Soft-SPS ist für sich betrachtet ein eigenständiges Multitasking-Betriebssystem. Die einzelnen Bearbeitungsebenen werden mittels Tasks realisiert. Wie bei der herkömmlichen SPS werden folgende Bearbeitungsebenen unterschieden:

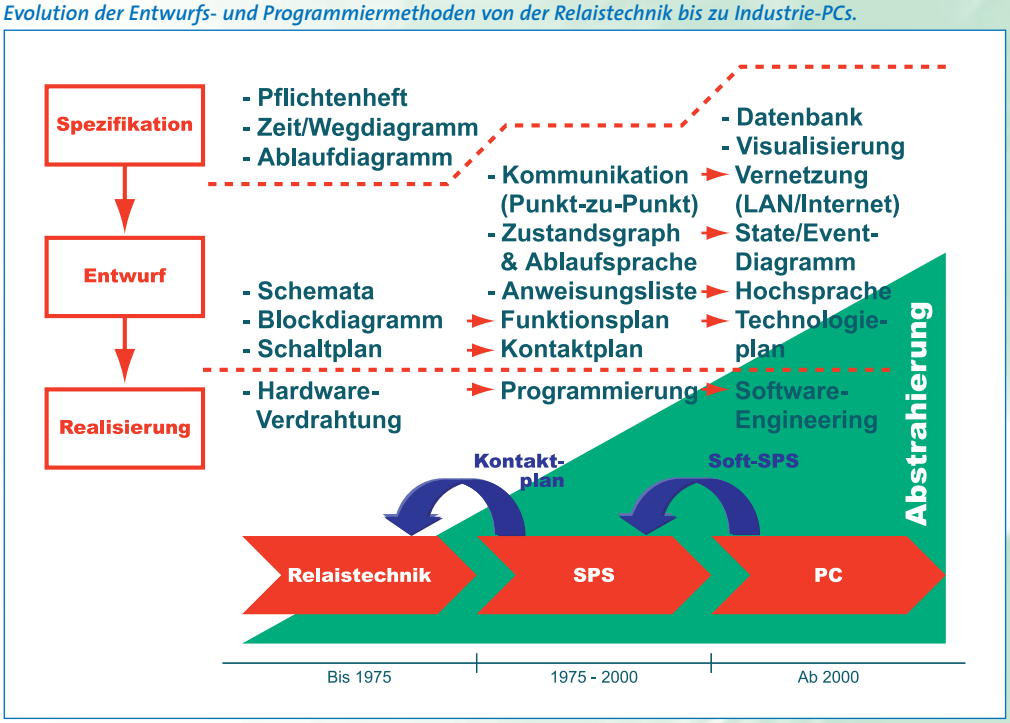
- Zyklisch
  Der normale SPS-Zyklus. Das Programm auf dieser Bearbeitungsebene wird kontinuierlich bearbeitet. Am Programmende wird es automatisch wieder von Neuem gestartet.
- Zeitgesteuert
  Aufruf von Programmen in einem festen Zeitraster (z. B. Regelungen).
- Ereignisgesteuert
  Aufruf eines Programms abhängig durch ein Ereignis (z. B. externes Signal, Interrupt oder Laufzeitfehler).

Anstelle der bekannten Organisationsbausteine (OBs), welche bei der klassischen SPS die Bearbeitungsebenen darstellen, wird bei der Soft-SPS eine entsprechende Task eingerichtet und parametriert. Innerhalb dieser Tasks werden nun wie gewohnt Funktionsbausteine programmiert.

Remanente Daten

Immer wieder Gegenstand von Diskussionen in Zusammenhang mit PC-basierter Automatisierung sind Remanenz und das Abschaltverhalten von Windows.
Unter remanenten Daten versteht man Daten, die auch nach Abschalten der Steuerung (also des PCs) erhalten bleiben.

Evolution der Entwurfs- und Programmiermethoden von der Relaistechnik bis zu Industrie-PCs.



Soft-SPS und Datenaustausch

Eine Stärke der Soft-SPS ist der Datenaustausch zu anderen Software-Anwendungen, die ebenfalls auf dem PC ablaufen. Vier Methoden zum Datenaustausch haben sich etabliert:

- Shared Memory
  Ein gemeinsamer Datenbereich im Arbeitsspeicher des PCs wird für Kommunikationszwecke definiert.
- Datei
  Beschreiben und Lesen einer Datei.
- DDE
  Einfacher Zugriff auf SPS-Daten von Windows-Anwendungen aus (z. B. Excel, Access)
- OPC
  Datenaustausch via OPC-Server. Wird von den meisten Automatisierungsanwendungen (z.B. Visualisierung) unterstützt.

Da diese Kommunikationsmechanismen alle im gleichen Rechnersystem

ablaufen, weisen sie eine entsprechend hohe Transferrate auf und sind einfach in der Handhabung.

Remanenz und Abschaltverhalten

Immer wieder Gegenstand von Diskussionen in Zusammenhang mit PC-basierter Automatisierung sind Remanenz und das Abschaltverhalten von Windows.

Unter remanenten Daten versteht man Daten, die auch nach Abschalten der Steuerung (also des PCs) erhalten bleiben. Diesen Komfort ist man von klassischen SPS-Systemen in der Art gewohnt, dass Datenbausteine und Merker in einem batteriegepufferten RAM abgelegt sind. Nun ist bei Standard-PCs ein solcher batteriegepufferter Speicher nicht vorgesehen. Deshalb geht man bei einer Soft-SPS zwecks Erreichens der Remanenz andere Wege.

Als Erstes stellt sich die Frage, ob remanente Daten überhaupt notwendig sind. Nicht jeder Ablauf oder Prozess ist darauf angewiesen. Mit etwas Überlegung bei der Programmierung kann dann oft auf remanente Daten verzichtet werden. Ist dennoch Remanenz unumgänglich, ist diese bei einer Soft-SPS ebenfalls möglich. Hierzu sind die remanenten Daten als solche zu deklarieren.

Über einen Funktionsaufruf lassen sich diese in einen Flash-Speicher oder auf die Festplatte speichern. Dies geschieht komplett unter der Kontrolle des Programmierers. Sind also beispielsweise in einer Anlage entsprechende Prozessschritte erreicht, kann der Anwender die dazugehörigen Daten remanent abspeichern, so dass sie bei einem eventuellen Aus- und Wiedereinschalten der Steuerung wieder vorhanden sind.

Häufigkeit der Datenspeicherung

Allerdings sollte man das Wegspeichern von Daten z. B. auf Flash nicht übertreiben. Es ist die Anzahl möglicher Schreibzyklen auf den Flash-Speicher zu beachten (Lebensdauer). Ein simples Abspeichern aller Daten nach jedem Programmzyklus ist in der Regel also nicht ratsam.

Geschickter ist es daher, ein sicheres Herunterfahren des PCs durch eine unterbrechungsfreie Stromversorgung (USV) zu gewährleisten. Diese überbrückt die Zeit bei einem Spannungsverlust (einfaches Ausschalten des PCs ohne explizites Herunterfahren) für eine korrekte Beendigung des Windows-Betriebssystems und leitet das Abspeichern der remanenten Daten ein. USVs für diesen Zweck, die ja nur wenige Minuten puffern müssen, sind kompakt und können z. T. auch direkt in den PC eingebaut werden.

Testmöglichkeiten und Debug-Werkzeuge

Eine grundlegende Diskrepanz zwischen Hochsprachen-Programmierung und SPS-Programmierung gibt es in punkto Testmöglichkeiten und Debug-Werkzeugen. Denn: Geht es um den Test von Steuerprogrammen für Maschinen oder Prozesse, ist es nicht möglich, den Programmablauf an x-beliebiger Stelle zu unterbrechen.

Programmtest – am «lebenden» Objekt

Eine grundlegende Diskrepanz zwischen Hochsprachen-Programmierung und SPS-Programmierung gibt es in punkto Testmöglichkeiten und Debug-Werkzeugen. Sie beruht auf den zu Grunde liegenden Testphilosophien: Bei den Hochsprachen mit ihrem Ursprung in der EDV-Welt geht es in stark abstrahierter Sichtweise um Berechnungen in jedweder Form, die gespeichert, weiterverarbeitet oder auch nur ausgedruckt werden. Diese Art der Datenverarbeitung ist entkoppelt von der realen Umwelt zu betrachten – das heisst, sie hat keine Interaktion mit einem realen System.

Ziel ist, so viel wie möglich in kürzester Zeit zu berechnen. Das Endergebnis ist letztlich unabhängig von der Rechengeschwindigkeit. Daher stört es auch nicht, wenn der Programmlauf zu Testzwecken unterbrochen wird, um Zwischenergebnisse zu untersuchen. Hochsprachenprogramme werden daher mit Methoden wie Breakpoints und Einzelschrittbetrieb getestet, wobei der Programmablauf gezielt stoppt.

Test von Steuerprogrammen

Bei Steuerungsanwendungen sind diese Testmethoden hingegen kaum geeignet: Geht es um den Test von Steuerprogrammen für Maschinen oder Prozesse, ist es nicht möglich, den Programmablauf an x-beliebiger Stelle zu unterbrechen, da eine Steuerung in Interaktion mit einem realen System, der Maschine, steht.

Ist beispielsweise der Positioniervorgang eines Antriebes, mit dem von Punkt A nach Punkt B zu fahren ist, einmal in Gang, darf nicht einfach die Programmabarbeitung unterbrochen werden: Die unkontrollierte Fahrt des Antriebes hätte schwere Schäden zur Folge.

Aber nicht nur die Vermeidung von Schäden widerspricht der Verwendung der Testmethoden aus der Hochsprachenprogrammierung. Dynamische Vorgänge lassen sich auf diese Weise überhaupt nicht wahrnehmen: denn gerade die Konstellation in der Maschine, die ein eventuelles Fehlverhalten hervorruft, lässt sich ohne laufendes Programm gar nicht erfassen.

Online-Testmethoden

Solche dynamischen Problemstellungen erfordern Online-Testmethoden, wie sie heute von einer Soft-SPS zur Verfügung gestellt werden. Online heisst, dass sämtliche Tests parallel zum laufenden Programm durchführbar sind, ebenso wie der Download von Programmteilen. Parallel zur Programmbearbeitung lassen sich interne SPS-Daten oder Variablen sowie Ein- und Ausgänge direkt anzeigen und modifizieren.

Funktionen wie z. B. Programmstatus zeigen zu jeder Programmzeile verschiedenste Statusinformationen und interne Zustände an und zwar genau so, wie sie zum Zeitpunkt der Abarbeitung der jeweiligen Programmzeile tatsächlich sind. Ein nicht zu unterschätzendes Qualitätsmerkmal einer Soft-SPS ist, wie gut Online-Testfunktionen neben den üblichen Debug-Hilfen wie Breakpoints, Einzelschritt, usw. unterstützt werden. (rb) ■