Messen, Steuern und Regeln mit PC's

SORCA - Eine neue Programmiersprache für schnelle Steuerungen

Dr. Hartmut B. Brinkhus

Sorcus Computer GmbH
Berghalde 54
D-6900 Heidelberg
Tel. 06221-384236

Zusammenfassung:

SORCA ist eine neue, leicht zu erlernende Programmiersprache für Echtzeit-Anwendungen. Sie eignet sich für viele analoge und digitale Meß- und Steuerungsaufgaben.
Das zugrunde liegende neue Prinzip beruht auf dem "Variablen-Prozessor" VP, der auf eine beliebig große Zahl von Variablen arbeitet. Er besitzt einen Funktions-Bus und einen Variablen-Bus. Über den Funktions-Bus erhält der VP seine Instruktionen und die Adressen aller Variablen, auf die die Funktion arbeitet. Der Variablen-Bus erlaubt gleichzeitig den Zugriff auf alle Variablen (interne und externe). Die Arbeitsweise des VP ist zyklisch, ein Zyklus durchläuft alle Funktionen.
Das Prinzip ist besonders vorteilhaft, wenn mehrere Prozesse gleichzeitig gesteuert werden müssen. Die Zahl der parallel laufenden Prozesse ist praktisch nicht begrenzt.
Z.Zt. emuliert ein herkömmlicher Mikroprozessor einen VP. Dazu wurde eine intelligente Zusatzkarte "Multi-LAB" für IBM-PC/XT/AT entwickelt, die einen eigenen Prozessor mit allen erforderlichen analogen und digitalen Interfaces beinhaltet. Damit konnte die Leistungsfähigkeit des SORCA-Compilers und des SORCA-Konzeptes bewiesen werden. Der nächste Schritt wäre die Entwicklung eines SORCA-Single-Chip-Prozessors.

1. Einführung

SORCA heißt: S oftwaremäßig
OR ganisierte
C yclische
A blaufsteuerung.

SORCA ist eine Programmiersprache für prozeß- und zeitplankontrollierte Ablaufsteuerungen. Die Sprache wurde ursprünglich für die Steuerung von Labor-Experimenten in Echtzeit entwickelt. Sie eignet sich aber für viele Meß- und Steuerungsaufgaben, die mit Reaktionszeiten von wenigen 100 us auskommen. Diese Zeit bezieht sich Emulationen mit herkömmlichen Prozessoren. Sie könnte mit einem speziellen SORCA-Prozessor (Hardware) um etwa den Faktor 100 verbessert werden.

Durch die bereits jetzt mit Emulationen erreichbare Geschwindigkeit kann SORCA in vielen Fällen programmierbare Steuerungen (SPS und FPS) ersetzen. Von Vorteil ist dabei, daß keine spezielle Hardware benötigt wird. SORCA läuft auf vielen Personal Computern.

Die Zahl der Funktionen ist für ein SORCA-Programm nur vom Speicherplatz abhängig, mit 64 KByte RAM können mehrere Tausend Funktionen realisiert werden, wobei auch mehrere, voneinander unabhängige Funktionsblöcke (Tasks) programmiert werden können.

SORCA ist Prinzip-bedingt Multi-Tasking-fähig.

Jede Funktion, mit Ausnahme der sog. Systemfunktionen, kann dabei beliebig oft verwendet werden. Die minimal mögliche Reaktionszeit verlängert sich aber mit der Zahl der programmierten Funktionen.

Da alle Funktionen z.Zt. noch softwaremäßig realisiert sind, können auch neue, beliebig komplexe Funktionen realisiert werden. Beispiele sind das Majoritätsgatter mit bis zu 126 Eingängen, Richtungsdiskriminatoren oder stochastische Encoder und Decoder. Bei vielen Funktionen, z.B. beim Zähler, können die Parameter auch als Ergebnis einer Zufallsfunktion angegeben werden.

2. Das SORCA-Projekt

Stufe 1: Definition der Sprache und Festlegung der Syntax

Stufe 2: Implementierung der Sprache als Compiler für einen herkömmlichen Mikroprozessor. Dabei emuliert ein Z80-Prozessor alle SORCA-Funktionen.

Stufe 3: Bei der Definition der Sprache wurde davon ausgegangen, daß ein spezieller SORCA-Prozessor (Chip) entwickelt wird, der alle Funktionen als Befehls-Codes (op-codes) unmittelbar ausführen kann. Die Verarbeitungsgeschwindigkeit läßt sich damit ca. um den Faktor 100 erhöhen. Damit eröffnen sich neue Anwendungsbereiche, wie z.B. Bildverarbeitung, Mustererkennung, etc.

Der aktuelle Stand

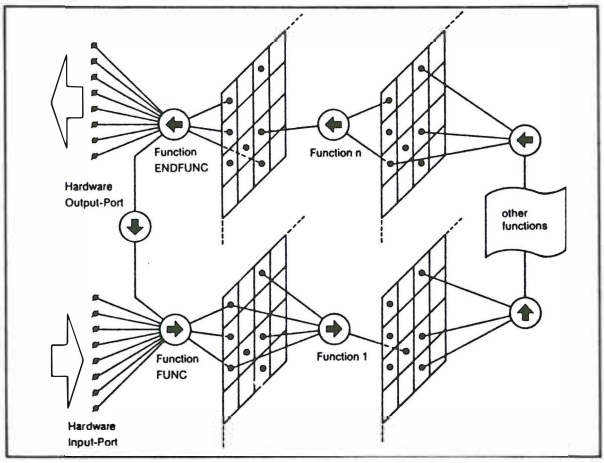
Stufe 2 ist bzgl. der digitalen Funktionen bereits realisiert und in vielen praktischen Anwendungen erprobt. Für verschiedene Personal Computer (z.B. IBM-PC/XT/AT, Apple II mit Z80-Karte und für die meisten Z80-Systeme) sind Compiler verfügbar (Version 1.x).

Für IBM-PC/XT/AT und damit kompatible Rechner gibt es von Sorcus Computer GmbH die intelligenten SORCA-Zusatzkarten "Multi-LAB" und "Multi-PIO" mit einem Z80H (8 MHz) als Co-Prozessor und vielen analogen und digitalen Schnittstellen (Beschreibung s.u.).

3. Das Funktionsprinzip

Jede Ablaufsteuerung ist durch Logikelemente realisierbar, wie sie z.B. in Form von integrierten Schaltungen (IC's) verfügbar sind. Wird die hohe Schaltgeschwindigkeit dieser IC's nicht benötigt, liegt es nahe, die Logikfunktionen in Software zu realisieren. Obwohl die heutigen Mikroprozessoren nicht dafür ausgelegt sind, sind bei geeigneter Programmstruktur doch gute Ergebnisse zu erzielen.

Die Abbildung zeigt das Prinzip, den "Variablen-Prozessor":



Der "Variablen-Prozessor", die prinzipielle Arbeitsweise eines SORCA-Programms

Alle Eingänge und Ausgänge von logischen Verknüpfungen und von anderen Funktionen werden als binäre Input- bzw. Output-Variable mit beliebig wählbaren Namen definiert. Über diese Namen kann dann jede Funktion jede Variable als Input- oder als Output-Variable verwenden.

Jedes SORCA-Programm ist so aufgebaut, daß nach einigen Funktionen zur Initialisierung die Funktionen folgen, die zyklisch durchlaufen werden sollen. Der Beginn dieser Funktionen wird mit der Funktion 'FUNC' angegeben. Danach beginnt der erste Zyklus mit dem Lesen der Zustände der Input-Ports. Anschließend werden die Funktionen abgearbeitet, also der Zustand aller Variablen neu bestimmt, und zwar in der Reihenfolge, wie sie im Programm angegeben wurden. Hat sich eine Variable verändert, so wird in den nachfolgenden Funktionen bereits der neue Zustand dieser Variablen verwendet.

Nach Durchlaufen eines Zyklus wartet das Programm, bis die vorprogrammierte Zykluszeit vorbei ist. Dann werden in der Funktion 'ENDFUNC' die externen Variablen entsprechend dem Zustand der zugeordneten Output-Variablen gesetzt. Anschließend beginnt das Programm den nächsten Zyklus wieder bei der Funktion 'FUNC'.

4. Hardware-Voraussetzungen (Version 1.10)

1. Mikroprozessor-System mit Z80-CPU und mind. 8 KRAM, möglich ist auch die Verwendung eines Single Board-Rechners. Als Schnittstellen sollten vorhanden sein:
2. Eingänge
3. Ausgänge
4. Ein Zeitgeber
5. Eine sog. SORCA-Schnittstelle

Prinzipiell läuft ein SORCA-Programm auch ohne die Voraussetzungen 2. bis 5. Ein- und Ausgangsleitungen (jeweils bis zu 64 bei der Version 1.10) sind nur erforderlich, soweit sie vom Anwendungsprogramm benutzt werden. Die SORCA-Schnittstelle kann außer zum Down-Loading des vom Compiler erzeugten Programms auch benutzt werden, um während des Programmlaufs mit dem Host-Rechner zu kommunizieren, z.B. Ergebnisse (Zählerstände) und Zustände von Variablen zu melden. Der Zeitgeber schließlich dient dazu, die Zykluszeit vorzugeben. Ist kein Zeitgeber vorhanden, dann kann die Zykluszeit auch von außen über eine Eingangsleitung vorgegeben werden.

Es ist möglich, SORCA-Programme der Version 1.10 auf allen Rechnern mit einer Z80-CPU laufen zu lassen, die die beschriebenen Voraussetzungen erfüllen und die über eine Kommunikations-Schnittstelle verfügen. Über diese Schnittstelle wird das Programm vom Haupt-Rechner, auf dem der Compiler läuft, in den Arbeitsspeicher (RAM) des Z80-Satellitenrechners geladen und gestartet.

Für IBM-PC/XT/AT und kompatible Rechner gibt es von SORCUS Computer GmbH lieferbare Satellitenrechner "Multi-LAB" und "Multi-PIO". Diese PC-Zusatzkarten enthalten alle erforderlichen Hard- und Software-Voraussetzungen (s.u.). Es ist auch möglich, mehrere dieser Karten gleichzeitig in einem PC zu benutzen.

5. Die intelligenten PC-Zusatzkarten "Multi-LAB" und "Multi-PIO"

Diese beiden Satellitenrechner wurden speziell für IBM-PC/XT/AT und kompatible Rechner entwickelt. Sie können aber auch als Single-Board Rechner eingesetzt werden. Die Kommunikation kann entweder über die parallele PC-Schnittstelle (wenn die Karte in einem IBM-PC eingesteckt ist) oder über die serielle RS 232 C Schnittstelle erfolgen.

Die Karten enthalten ein eigenes Multi-Tasking Betriebssystem und fertige Echtzeit-Meßprogramme im ROM, z.B. zur Analogdatenerfassung, für Frequenz- und Pulsbreitenmessungen, frei programmierbare Funktionsgeneratoren, und zur digitalen Signalverarbeitung (z.B. Filter, Fourier Transformation und Power Spektrum).

"Multi-LAB" enthält analoge und digitale Schnittstellen und Funktionsgruppen, "Multi-PIO" nur digitale.

Technische Daten Multi-LAB :

Bus	IBM PC, XT oder AT und kompatible Rechner
Benötigter Platz	Ein Erweiterungs-Steckplatz im PC (Slot)
Prozessor	Z80H, Taktfrequenz 8 MHz
Speicher	64 KByte RAM und 64 KByte EPROM
PC-Schnittstelle	Die Karte belegt 5 I/O-Adressen (einstellbar)
Kommunikation	Mailbox-Prinzip (1,2 MBit/s)
Interrupt	In beide Richtungen Interrupt-fähig.
Analog-Eingänge	16 Kanäle, 12-Bit Auflösung, 8 us Wandlungsz.
Analog-Ausgänge	10 Kanäle, 2 mit 12-Bit und 8 mit 8-Bit Aufl.
Plotter-Ansteuerung	Über 12-Bit D/A-Wandler mit Pen-lift Ausgang
Digitale Eingänge	24 Schmitt-Trigger Eingänge, TTL-kompatibel
Digitale Ausgänge	24 o.C. Ausgänge (je 30 V, 300mA), TTL-komp.
Trigger Eingänge	3 (für Trigger, Zähler, Pulsbreitenmessungen)
Zeitgeber	4 16-Bit Timer, Interrupt-fähig
Centronics-Drucker	1 parallele Schnittstelle
RS-232 C	1 serielle Schnittstelle, Voll-duplex
Uhr	Batterie-gepuffert für Datum und Uhrzeit
Kontroll-LED	Anzeige des Funktionszustandes der Karte

6. Die Syntax von SORCA

Das Eingabeformat ist für alle Funktionen gleich:

```
FUNKTION (PARAMETER): Q'VARIABLE-1 I'VARIABLE-2 .....
```

Jede Zeile beginnt mit dem Namen der Funktion. Erfordert die Funktion Parameter, so folgen diese, in runde Klammern gesetzt, dem Namen. Wenn mehrere Parameter angegeben werden, werden diese durch Komma getrennt.

Nach einem Doppelpunkt folgen dann die Namen aller Variablen, die mit dieser Funktion verknüpft sind. Dabei steht vor jedem Namen ein Zeichen und ein Apostroph. Das Zeichen nennt die Bedeutung der Variablen, gibt also an, mit welchem logischen Eingang oder Ausgang der Funktion die angegebene Variable verbunden ist.

Die Namen der Funktionen, Anweisungen, Systemvariablen und -konstanten sind reserviert. Die Namen der vom Benutzer definierten internen Variablen können frei gewählt werden (16 Zeichen).

Externe System-Eingangs-Variable:	IP0	IP1	IP2	...	IP63
Externe System-Ausgangs-Variable:	OP0	OP1	OP2	...	OP63
Interne System-Variable:	STOP		PRINT		NOISE
System-Konstante:	0	1			

"IP0" bis "IP63" sind die den 8 x 8 Bit der Input-Ports zugeordneten externen Eingangs-Variablen. "OP0" bis "OP63" sind die den 8 x 8 Bit der Output-Ports zugeordneten externen Ausgangs-Variablen. "STOP" ist eine System-Ausgangs-Variable, die den Programmablauf so lange unterbricht, bis das ASCII-Zeichen 47H = 'G' über die SORCA-Schnittstelle empfangen wird. Eine pos. Flanke an der System-Ausgangs-Variablen "PRINT" führt zur Ausgabe des aktuellen Inhaltes von 8 Zählern (SUM0 bis SUM7) über die SORCA-Schnittstelle. "NOISE" ist eine System-Eingangs-Variable, die Rauschen erzeugt. "0" und "1" sind Konstante, die den Werten 0 und 1 entsprechen.

7. Die Funktionen der Version 1.10

Dynamische Variable sind unterstrichen. Zur Vereinfachung der Schreibweise ersetzt im folgenden ein * eine Eingangsvariable und ein $ eine Ausgangs-Variable.

Programmablauf	INIT ENDINIT FUNC ENDFUNC
Konfiguration	INPORT (n) OUTPORT (n) PIO (n) TIMER (n) IN (n): L'* 0'* 1'* 2'* 3'* 4'* 5'* 6'* 7'* OUT (n): L'* 0'* 1'* 2'* 3'* 4'* 5'* 6'* 7'*
Initialisierung	CYCLE (cc) CLEARV CLEARO SET: Q'$ RESET: Q'$
Gatter	EQU: I'* Q'$ INV: I'* Q'$ AND: I'* I'* Q'$ OR: I'* I'* Q'$ XOR: I'* I'* Q'$ NAND: I'* I'* Q'$ NOR: I'* I'* Q'$ NXOR: I'* I'* Q'$
Majoritäts-gatter	MG3/2: I'* I'* I'* Q'$ MG: +'* +'* +'* -'* -'* -'* Q'$ (+/- max. je 63)
Zähler	Fix: COUNTF (nn): C'* P'* Q'$ Normalverteilung: COUNTN (mm,ss): C'* P'* Q'$ Gleichverteilung: COUNTG (uu,oo): C'* P'* Q'$ Stufen: COUNTS (aa,hh,z): C'* P'* Q'$
Mono-Flop und Nachtriggerbares Mono-Flop	MONOF (nn): T'* R'* Q'$ RMONOF (nn): T'* R'* Q'$ (Normal-, Gleichverteilung und Stufen s. Zähler)
Flip-Flops	RSFF: S'* R'* Q'$ DFF: D'* C'* R'* Q'$
Oszillator	OSCI (pp,nn): R'* Q'$
Verzögerung	DELAY (d): D'* Q'$ (d = 1 - 7)
Auf/Ab-Zähler	SUMi: U'* D'* R'* (i = 1 - 7)

8. Hinweise zur Angabe von Funktionen und Parametern

Alle Funktionen können in einem Programm beliebig oft aufgerufen werden. Das gilt auch für die normalerweise nur einmal benötigten Funktionen, wie z.B. 'CLEARO', 'PIO (n)' oder andere. Jedes SORCA-Programm kann z.B. beliebig oft seine eigenen I/O-Ports definieren.

Als Parameter sind nur ganze Zahlen erlaubt. Bei den Funktionen COUNTx, MONOx und NMONOx kann der Parameter (nn) in verschiedenen Formaten angegeben werden:

1. Als Konstante, dann ist x = F, z.B. COUNTF (nn): Q'$ C'* P'*

2. Als Zufallszahl einer Normal- bzw. Poissonverteilung, dann ist x = N, z.B. COUNTN (mm,ss): Q'$ C'* P'*
   (mm = Mittelwert, ss = Standardabweichung)
   Ist eine Normalverteilung nicht möglich, generiert der Compiler eine Poissonverteilung, gibt aber eine Warnung aus.

3. Als Zufallszahl einer Gleichverteilung, dann ist x = G, z.B.:
   COUNTG (uu,oo): Q'$ C'* P'*
   (uu = untere Grenze, oo = obere Grenze)

4. Als Zahlenfolge mit festen Stufen, dann ist x = S, z.B.:
   COUNTS (aa,hh,z): Q'$ C'* P'*
   (aa = Anfangswert, hh = Stufenhöhe, z = Zahl der Stufen)

9. Das Erstellen von Quell-Programmen

Ein Quell-Programm wird mit einem Editor so eingegeben, wie bei anderen Programmiersprachen. Der Name der Datei kann frei gewählt werden, der File-Typ muß aber .SOR sein. Kommentare innerhalb eines Programms sind in spitze oder eckige Klammern zu setzen.

```
(*************************************************************)
(             SORCA-Beispiel-Programm "UHR.SOR"               )
(                                                             )
(    Liefert Impulse von 100 Hz an OP0, von 1 Hz an OP1, von  )
(    1/min an OP2 und von 1/h an OP3. Die Uhrzeit wird automa-)
(    tisch jede Sekunde ausgedruckt, wenn IP1 = 1 ist. Wenn   )
(    IP1 = 0 ist, dann kann ueber eine positive Flanke an IP0 )
(    ausgedruckt werden.                                      )

INIT
INPORT (128)
OUTPORT (128)
TIMER (136)
CLEARO
CLEARV
CYCLE (10)                    (= 1 ms)
ENDINIT

FUNC
OSCI (1,1) : Q'TAKT           (erzeugt 500 Hz Rechteck-Impulse)
COUNTF (5) : C'TAKT  Q'OP0    (Teiler /5)
COUNTF (100) : C'OP0  Q'OP1   (Teiler /100, liefert Sekundentakt)
COUNTF (60) : C'OP1  Q'OP2    (Teiler /60, liefert Minutentakt)
COUNTF (60) : C'OP2  Q'OP3    (Teiler /60, liefert Stundentakt)

SUM3 : U'OP0  R'OP1           (Displayzaehler für 1/100 s)
SUM2 : U'OP1  R'OP2           (   "               SEKUNDEN)
SUM1 : U'OP2  R'OP3           (   "               MINUTEN)
SUM0 : U'OP3                  (   "               STUNDEN)

AND : I'IP1  I'OP1  Q'PRINT/SEC
INV : I'IP1  Q'INVIP1
AND : I'INVIP1  I'IP0  Q'PRINTIP0
OR : I'PRINT/SEC  I'PRINTIP0  Q'PRINT

ENDFUNC
(*************************************************************)
```

10. Der Compiler

Nach Eingabe des Quell-Programms wird der Compiler aufgerufen. Das Hauptmenü zeigt die Möglichkeiten:

```
SORCA, Copyright 1985, 1986 (c) Sorcus Computer GmbH, Heidelberg
Compiler-Version: 1.10                                Timer: Z80-CTC

      S   Syntaxcheck
      C   Compilation
      P   Print-file-generation
      R   Run
      I   Install SORCA environment
      X   Install driver for communication with remote system
      T   Transmit program to remote system via X-driver
      E   End
```

Hat die Syntaxprüfung keine Fehler ergeben, kann das Programm übersetzt werden. Im Falle von Fehlern wird der Fehlertyp und die Nummer der Zeile, in der der Fehler gefunden wurde, während der Syntaxprüfung ausgegeben oder in eine Datei geschrieben. Bei der Syntaxprüfung wird unterschieden zwischen Fehlern ('ERROR') und Warnungen ('WARNING').

Außerdem ermittelt der Compiler die minimal mögliche Zykluszeit für die Funktionen, die innerhalb der Schleife FUNC bis ENDFUNC programmiert sind. Sie ergibt sich aus der Summe der Durchlaufzeiten (maximale Werte) aller aufgerufenen Funktionen. Die programmierte Zykluszeit (bzw. der extern angelegte Takt) muß größer sein als diese minimale Zykluszeit.

Während des Compilierens kann auf dem Bildschirm die Arbeit des Compilers verfolgt werden (Phase, Zeilen-Nr., etc.). Der Compiler erzeugt eine Datei mit dem Namen UHR.LAB. Sie kann auf den Satellitenrechner übertragen und dort gestartet werden. Das Programm kann auch während des Laufes vom Haupt-Rechner über die KommunikationsSchnittstelle beeinflußt werden.