PROZESS-FORTRAN 330

G. Teuschler
SIEMENS AG
Erlangen
E STE 322

Gliederung

1. Die Sprache FORTRAN
2. Der Compiler - ein Dienstprogramm
3. Fehlerauswertung zur Laufzeit

PROZESS-FORTRAN

Aufbau der Sprache:

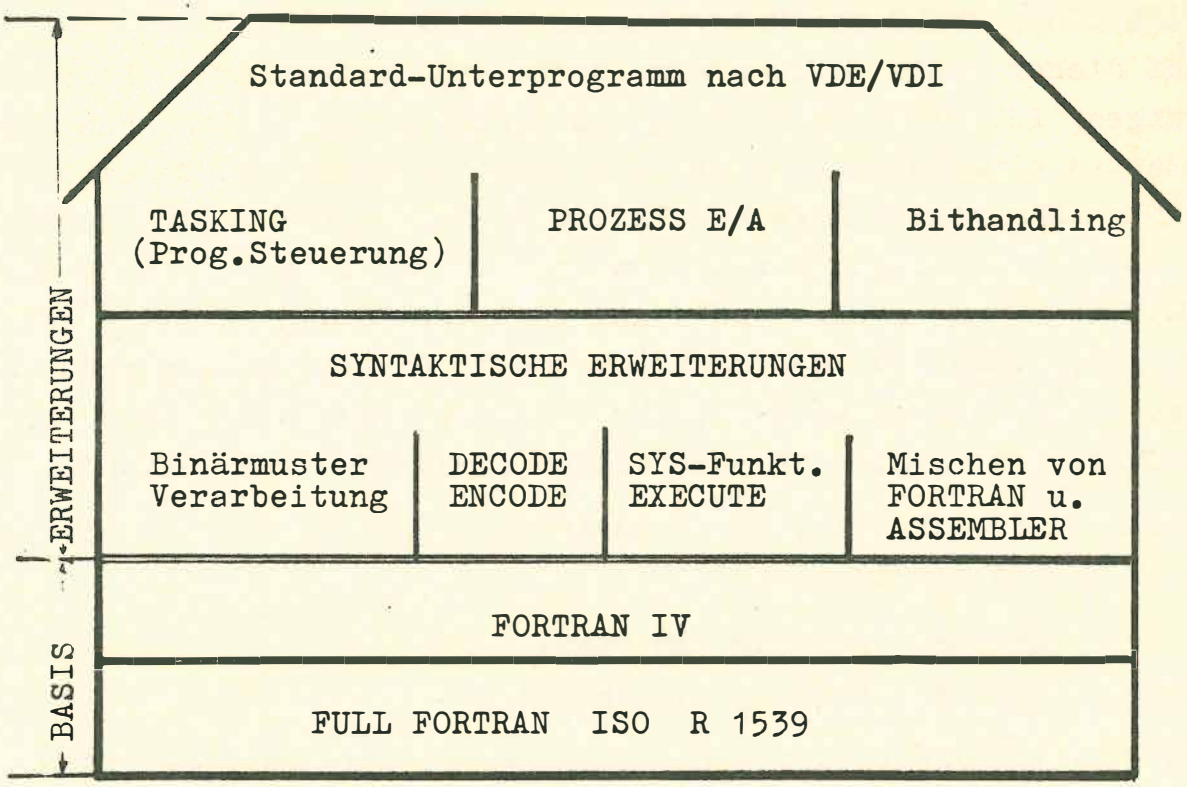

Bild 1

1. Die Sprache FORTRAN

FORTRAN ist eine problemorientierte und maschinenunabhängige Programmiersprache. PROZESS FORTRAN ist eine geläufige Bezeichnung für die Erweiterungen und Anpassung des Sprachumfanges von FORTRAN an die Aufgaben der Prozeßsteuerung (Bild 1).
PROZESS FORTRAN für den Prozeßrechner 330 baut auf den Sprachumfang von FULL FORTRAN auf, wie er international genormt ist (ISO R 1539) und umfaßt darüber hinaus alle Sprechelemente von FORTRAN IV (wie z.B. die IMPLICIT-Spezifikation, DEFINE FILE, NAME LIST, ENTRY bei Unterprogrammen usw.).
Auf dieser Basis gibt es eine Reihe von syntaktischen Erweiterungen. Für die Verarbeitung von Binärmustern wurden neue Konstanten eingeführt, die es erlauben, Bitmuster und Sedezimalzahlen als ganzzahlige Konstanten zu formulieren. Für die Verknüpfung von Bitmustern stehen die logischen Operatoren von FORTRAN IV wie .NOT., .AND. und .OR. zur Verfügung sowie der Operator .EOR. für das exklusive Oder. Darüber hinaus gibt es Schiebeoperatoren .ISHIFT. für arithmetisches Verschieben und .LSHIFT. für logisches Verschieben von Bitmustern (Beispiel 1).

Als neue Sprachmittel für eine leistungsfähige Textverarbeitung gibt es die Anweisungen

DECODE (analog zum READ)
und ENCODE (analog zum WRITE)

für die formatgesteuerte Übertragung von Zeichenfolgen im Arbeitsspeicher in die Interndarstellung und umgekehrt (Beispiel 2).
Für die Abfrage von Systemwerten, wie Uhrzeitzellen, Datumszellen, Anzeigen wurden parameterlose Systemfunktionen eingeführt, damit der Anwender leicht zu den Informationen des Organisationsprogrammes und des Laufzeitsystems zugreifen kann.
Für den Aufruf von Subroutine-Unterprogrammen aus der Standardbibliothek steht ein neues Schlüsselwort zur Verfügung:

EXECUTE name (Parameterliste).

Es dient zur eindeutigen Kennzeichnung des Aufrufs von Standard-Subroutinen und ist formal der CALL-Anweisung gleichwertig (Beispiel 3).

Neben den bewährten Möglichkeiten des Koppelns von FORTRAN mit Assemblersprache über Unterprogramme gibt es in PROZESS-FORTRAN die Möglichkeit, Befehls- und Datenzeilen in Assemblersprache direkt zwischen FORTRAN-Anweisungen einzufügen. Damit kann PROZESS FORTRAN jeweils optimal an spezielle Wünsche des Anwenders angepaßt werden. In den eingefügten Assemblerzeilen können alle symbolischen Adressen des FORTRAN-Rahmenprogrammes und alle den Externadressen benutzt werden.
Als Erweiterung der Standardbibliothek stehen standardisierte Unterprogramme für die Programmsteuerung für Prozeß-Ein/Ausgabe und für Bithandling zur Verfügung.

Beispiele

Beispiel 1

```
C          BINAERMUSTER VERARBEITUNG
           IMPLICIT INTEGER *2 (I-Z  )
           LBYTE = I. LSHIFT. - 8
           RBYTE = I .AND. HR'FF'
```

Erklärung:

INTEGER*2	Werte belegen 16 Bit
.LSHIFT.	Operator für logisches Schieben
HR'FF'	Sedezimalkonstante ØØFF

Beispiel 2

```
C          DECODE / ENCODE
           INTEGER *2 ZFELD(3), YFELD(4)
           DECODE (6, 100, ZFELD) R
100        FORMAT (F 6.Ø)
           ENCODE (8, 200, YFELD) I
200        FORMAT ('I=',   I6)
```

Erklärung:	1. Parameter von DECODE und ENCODE	Satzlänge in Byte
	2. Parameter	Format Nr.
	3. Parameter	Zeichenfeld

Beispiel 3

```
C           SYS-FUNKT UND EXECUTE
            IF (SYSERRO .NE. Ø)  STOP 99
            EXECUTE WAIT (K,L,M)
```

Erklärung:	SYSERRO	Systemfunktion zur Abfrage der Anzeigenzelle des Laufzeitsystems
	EXECUTE	geschützter Aufruf (analog zu CALL)

Beispiel 4

```
C           MISCHEN FORTRAN-ASSEMBLER
            SUBROUTINT SPEZI (I1, I2, I3)
            INTEGER *2 I1, I2, I3
ASSEM
            $SPEZIAL "P1 @ I1"P2 @ I2"P3 @ I3;
FORTRAN
            RETURN
            END
```

Erklärung:	ASSEM	Umschaltkarte FORTRAN-ASSEMBLER
	FORTRAN	Umschaltkarte ASSEMBLER-FORTRAN

Als Assemblerzeile wurde hier ein Makroaufruf mit Kennwortparametern benutzt.

Beispiel 5

```
C               STANDARD UP
C               TASKING
                DIMENSION IPROG(2)
                EXECUTE START (IPROG, Ø, Ø, M)
```

Erklärung: START Routine zum Starten eines Programms

1. Parameter Beschreibung des Programmes
2. Parameter Zeitverzögerung
3. Parameter Zeiteinheit der Zeitverzögerung
4. Parameter Fehlervariable

```
C               PROZESS E/A
                INTEGER *2 DIADR(3), DIWERT(2), DOADR(2)
                EXECUTE DI (2, DIADR, DIWERT, M)
                EXECUTE DOM (1, DOADR, DOWERT, 1, M)
```

Erklärung: DIADR, DOADR Feld für die Beschreibung der Hardware-Adressierung

DIWERT Feld für den Digitalwert

DI Digitaleingabe

1. Parameter Anzahl der Werte
4. Parameter Anzeigenvariable

DOM Impuls-Ausgabe

1. Parameter Anzahl der Werte
4. Parameter Zeit für die Impulsdauer
5. Parameter Anzeigenvariable

```
C               BIT HANDLING
                CALL BSET (I,5)
                CALL BCLR (J,10)
                IF (BTEST(K,15) .AND.Z.GT.Ø) GOTO 100
```

Erklärung: BSET Unterprogramm zum Bitsetzen

BCLR Unterprogramm zum Bitlöschen

BTEST Logische Funktion zur Bitabfrage

1. Parameter Wert
2. Parameter Bitnummer

2. Der Compiler - ein Dienstprogramm

Der Compiler ist ein segmentiertes Programm und in einem Hauptspeicherbereich von minimal 8 K Wörtern ablauffähig. An einen größeren Laufbereich paßt er sich automatisch an und verkürzt die Kompilierzeit.
Standardmäßig übersetzt er von Quellsprache in Grundsprache. Alternativ kann aber auch eine Übersetzung in Assemblersprache gewünscht werden. Die Protokollierung erfolgt am Ende der Übersetzung, wobei die Fehlermeldungen als Klartext-Protokoll vor der fehlerhaften Zeile erscheinen.
Auf der Eingabeseite (Quellprogramme) und auf der Ausgabeseite (Moduln in Grundsprache bzw. Assembler und Protokolle) arbeitet der Compiler mit Bibliotheken im Sinne des Systems 300-16 Bit. Dadurch ist es möglich, daß auch alle Ein- und Ausgaben über Peripherspeicher erfolgen können.
Über die Bedienung werden die unterschiedlichsten Funktionen der Übersetzung aktiviert und ausgewählt, z.B.:

- Eingabedatenstrom (Quellprogramm)
- Ausgabedatenstrom (Grundsprachemoduln)
- Ausgabedatenstrom (Assemblersprache)
- Protokollgerät ('Papier' oder Bibliothek)
- Testhilfewünsche
- Syntaxcheck

(Bild 2)

3. Fehlerauswertung zur Laufzeit

In der Regel wird ein Anwenderprogramm bei einem Laufzeitfehler mit einer entsprechden Fehlermeldung abgebrochen. Dieses Verhalten ist jedoch in Echtzeitprogrammen nicht erwünscht, deshalb gibt es in PROZESS FORTRAN für den PR 330 die Möglichkeit, von der Standardfehlerbehandlung auf eine Anwenderfehlerbehandlung umzuschalten. In diesem Zustand wird die Fehlerkennzahl (Fehlerursache) in einer Fehlervariablen abgespeichert.
Durch den Aufruf

```
          CALL ERROR (1)
bzw.      EXECUTE ERROR (1)
```

wird die Standardfehlerbehandlung ausgeschaltet und die Anwenderfehlerbehandlung eingeschaltet.
Tritt ein Laufzeitfehler auf, dann kann er durch die SYS-Funktion SYSERRO abgefragt werden, dabei wird der interne Wert der SYS-Funktion jeweils auf Ø zurückgesetzt.

Beispiel: Eingabe über READ mit Fallunterscheidung Gerätefehler oder FORMAT-Fehler.

```
          CALL ERROR (1)
          READ (10,100) (KARTE (I), I = 1,20)
          IFEHL = SYSERRO
          IF(IFEHL .GT. 300 000) GOTO 30
          IF(IFEHL .GT. 200 000) GOTO 20

C         GERAETEFEHLER
30           CONTINUE

C         FORMATFEHLER
20           CONTINUE
```

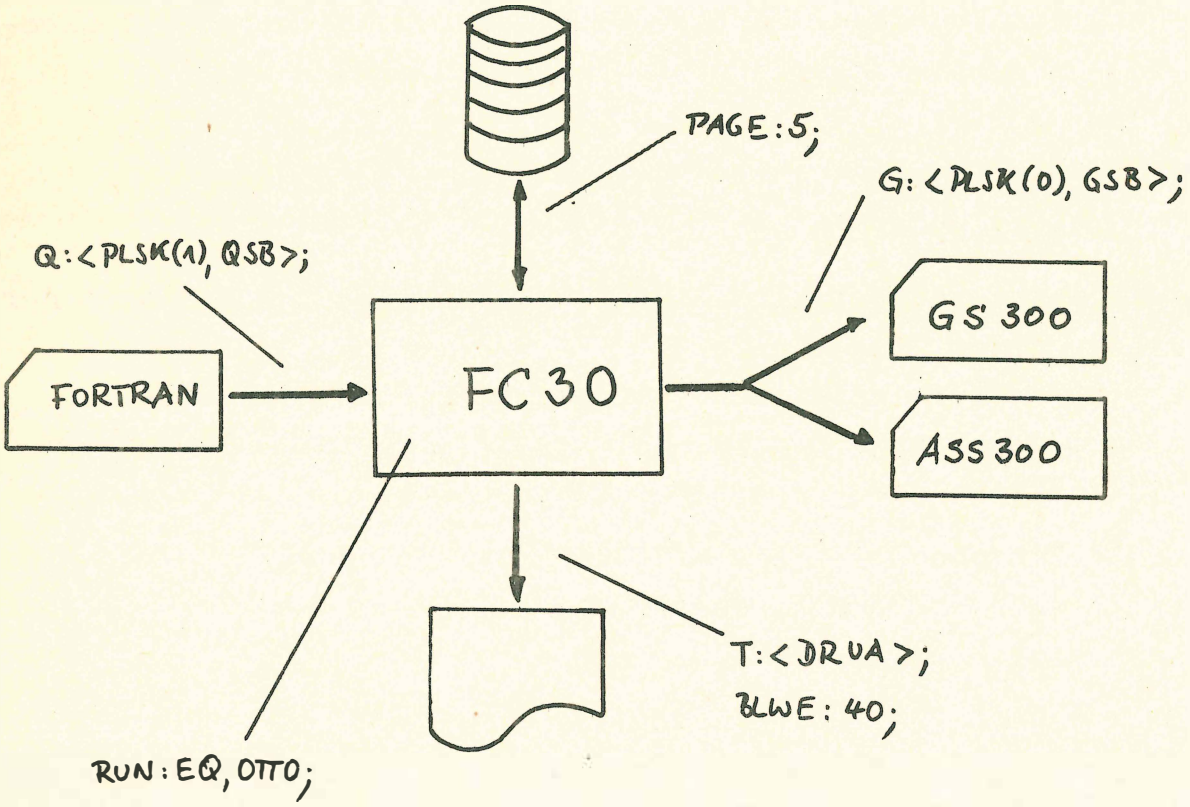
PAGE:5;
G:<PLSK(0), GSB>;
Q:<PLSK(1), QSB>;
GS 300
FORTRAN
FC30
ASS 300
T:<DRUA>;
BLWE:40;
RUN:EQ, OTTO;

Bild 2