Implementierung von PEARL auf IBM PC unter PC-DOS

Erwin Kneuer, Werum GmbH
Glogauer Straße 2 A, 2120 Lüneburg

Zusammenfassung

Mit der steigenden Leistungsfähigkeit von Personal-Computern haben sich auch deren Einsatzgebiete erweitert. In zunehmendem Maße werden PCs auch in Echtzeitanwendungen eingesetzt, für deren Programmierung die Sprache PEARL besonders geeignet ist.
Die Firma Werum bietet bereits seit einigen Jahren ein PEARL-Programmiersystem /4/ auf IBM PCs unter dem Echtzeit-Betriebssystem AT/RTX an. Die Nachfrage nach diesem System war erstaunlich groß, der insgesamt hohe Preis verhinderte jedoch eine entsprechende Verbreitung. Außerdem hat IBM im März dieses Jahres eine neue PC-Familie, das System PS/2, mit dem multitasking-fähigen Betriebssystem OS/2 angekündigt, das ein gemeinsames Programmierinterface zu PC-DOS (ab Version 3.3) besitzt.

Aus diesen Gründen beschloß Werum, PEARL "auf dem Wege zu OS/2" unter PC-DOS zu implementieren. Im folgenden Beitrag wird diese Implementierung vorgestellt.

1. Implementierter Sprachumfang

Auch für die PC-Implementierung unter DOS wurde das von der Firma Werum GmbH, Lüneburg, entwickelte portable PEARL-Programmiersystem portiert. Damit steht unter DOS der gleiche Sprachumfang zur Verfügung wie für die PEARL- Implementierungen von Werum auf den folgenden Rechnersystemen:

- Apollo Domain Workstation mit UNIX
- Hewlett Packard HP 9000-3xx mit HP-UX
- Hewlett Packard HP 3000 mit MPE IV
- IBM PC AT mit AT/RTX
- IBM 43xx mit CMS
- intel-Systeme mit iRMX86 oder iRMX286
- Motorola 68000 mit VERSADOS
- Norsk Data NORD 100 mit SINTRAN
- PCS CADMUS 9000-Serie mit MUNIX
- PEARL Engine 68000 mit BAPAS-K
- Siemens PC16-20 mit FLEXOS
- Siemens R/M-Serie mit ORG 300 PV / ORG-M
- Siemens 7.xxx mit BS 2000
- SUN Workstation mit UNIX
- VAX 11/7xx mit VMS
- Zilog Z 8000 mit BAPAS-K (Compilierung auf VAX 11/7xx)

Konkret umfaßt der Sprachumfang Basic PEARL /2/ mit mächtigen Erweiterungen in Richtung Full PEARL /3/, die hauptsächlich aus den folgenden Sprachkonstrukten bestehen:

- Benutzerdefinierte Datentypen (TYPE)
- Referenzen (REF)
- BOLT als weiterer Datentyp für Synchronisationsvariablen
- Definition von neuen Operatoren (OPERATOR)
- Felder und Strukturen bzw. benutzerdefinierte Datentypen als Komponenten in STRUCT- bzw. TYPE-Definitionen
- Lokale Prozeduren in Tasks bzw. Prozeduren
- Listen von SEMA- bzw. BOLT-Variablen in Synchronisationsanweisungen
- Variable CHARACTER- bzw. BIT-Selektion.

Der genaue Sprachumfang ist in dem Buch "Introduction to PEARL" /5/ beschrieben.

2. DOS und Echtzeit, Einschränkungen hinsichtlich Multitasking

Das Betriebssystem DOS wurde für single-user - single-task - Betrieb entworfen und ist eigentlich damit als Host-Betriebssystem für eine Echtzeitsprache ungeeignet. Wenn aber ein Programm, das unter DOS abläuft, aus einem Betriebssystem mit Multitaskingfähigkeiten und seinen Anwendungstasks besteht, so gibt es solange keinerlei Einschränkungen im Echtzeitverhalten wie keine DOS-Funktion benutzt wird.
In der in diesem Beitrag vorgestellten Implementierung wird DOS für die Ein- und Ausgabe mit den Standard-Geräten benutzt. Die Benutzung einer DOS-Funktion, was auf Anwenderebene

E/A-Operationen mit Standardgeräten entspricht, hat dann zur Folge, daß solange kein Taskwechsel in der Anwendung stattfindet, bis DOS wieder verlassen wurde. Hinzu kommt, daß prozessorfreie Zeit, die während der Ein-/Ausgabe-Operation im Gerätetreiber anfällt, nicht genutzt werden kann. Für schnelle E/A-Vorgänge im Millisekundenbereich, z.B. E/A mit Harddisk oder Console-Ausgabe, spielt dies kaum eine Rolle; es kann hier sogar durch die Einfachheit des Betriebssystems zu Performance-Vorteilen gegenüber Echtzeit-Betriebssystemen kommen. Der Nachteil wird natürlich umso größer, je länger eine E/A-Operation dauert. Fatal wird dies bei Eingabe vom Terminal, wo die Dauer der Operation davon abhängt, wie schnell eine Keyboard-Taste gedrückt wird. Für Eingabe von der Console gilt diese Einschränkung allerdings nicht, da DOS die Möglichkeit bietet, den Console-Eingabe-Buffer auf vorhandene Eingabe abzufragen. Eine PEARL-Task kann also auf Eingabe von der Console warten, während andere Tasks der Anwendung ausgeführt werden.

Weitere Einschränkungen hat diese PEARL-Implementierung gegenüber anderen unter Host-Betriebssystemen mit Echtzeiteigenschaften nicht. Auch PEARL-Schedules werden, basierend auf einem 18 ms Takt, den die batteriegepufferte Uhr des PC's liefert, zeitgerecht aktiviert, wenn zu diesem Zeitpunkt ein Taskwechsel erlaubt ist.

Eine zeitkritische Prozeßdatenerfassung (nicht Verarbeitung) wird beim Einsatz von interruptgesteuerten und DOS-unabhängigen Treibern von dem PEARL-System durch die Bereitstellung einer Treiberschnittstelle, auf die in diesem Beitrag noch näher eingegangen wird, unterstützt.

3. Implementierung

3.1 Hardware- und Software-Voraussetzungen zur Programmentwicklung

Zur Compilierung von PEARL-Moduln muß der PC mindestens die folgenden Betriebsmittel bereitstellen:

- 512 KB Speicher
- 10 MB Festplatte
- Arithmetischer Coprozessor
- PC-DOS oder MS-DOS ab Version 3.00
- Microsoft Assembler ab Version 2.00
- Console.

Zur Ausführung von PEARL-Programmen wird mindestens benötigt:

- Arithmetischer Coprozessor
- PC-DOS oder MS-DOS ab Version 3.00
- Console

Der Speicher hängt von der Größe der Anwendung ab, das PEARL-Laufzeitsystem benötigt maximal 50 KB.

3.2 Portierung

Das portable PEARL-Programmiersystem von Werum besteht aus folgenden Komponenten:

- PEARL-Compiler Front End
- PEARL-Codegeneratoren für eine Vielzahl von Rechnerarchitekturen, die C, Assembler oder Objectcode erzeugen
- PEARL-Betriebssystem BAPAS-K
- PEARL-E/A-Routinen BAPAS-EA

Alle hier aufgeführten Komponenten sind in PEARL programmiert und deshalb mit einem geeigneten Codegenerator sehr einfach auf ein beliebiges Zielsystem zu portieren. Für diese Implementierung konnte auf einen bereits existierenden Codegenerator (aus der INTEL-Implementierung), der Code für den 8086-Prozessor erzeugt, zurückgegriffen werden.
Für eine vollständige Implementierung waren dann noch folgende Interfaces zu erstellen:

- E/A-Interface für den PEARL-Compiler
- E/A-Interface als Treiber für die PEARL-E/A-Anweisungen
- BAPAS-K-Interface wie Timer, Taskwechselroutine usw.

Außerdem benötigen PEARL-Programme noch eine Laufzeitunterstützung für bestimmte Sprachkonstrukte wie z.B. CHARACTER- und BIT-Selektion. Hier konnte auch auf einen Assembler-Modul aus der INTEL-Implementierung zurückgegriffen werden.

3.3 Eigenschaften von PEARL-DOS

Der Compiler übersetzt einen PEARL-Modul in einen Assembler-Modul, der dann noch assembliert werden muß, um einen linkbaren Objekt-Modul zu erhalten. Im erzeugten Assembler-Modul werden als Kommentar Hinweise auf die Ursprungzeile im PEARL-Modul eingefügt. Im Benutzerhandbuch ist außerdem die PEARL-Laufzeitorganisation genau beschrieben, so daß leicht in Assembler oder auch anderen Sprachen geschriebene Module zu einem PEARL-Programm hinzugefügt werden können.

Steuerbar durch Compiler-Parameter, wird ein Quell-Listing und/oder eine Cross-Referenzliste erzeugt. Zudem kann die Laufzeit-Überwachung von Feldgrenzen und Referenzen durch Compiler-Parameter ein- oder ausgeschaltet werden.

Ein Preprozessor gestattet das Einfügen von Programmtexten aus Files (%INCLUDE) und bedingte Compilierung (%IF). Damit können zum Beispiel in mehreren Moduln verwendete Schnittstellen aus einer getrennten Textdatei zur Übersetzungszeit textuell einkopiert werden, so daß die Einhaltung der Modulschnittstellen implizit gewährleistet wird.

Mit Hilfe des DOS-Programms LINK und einer PEARL-spezifischen Library kann dann ein ablauffähiges Programm erzeugt werden, das wie jedes andere DOS-Programm gestartet wird.
Der Anwender kann die Stackgröße jeder Task in einfacher Weise festlegen, ansonsten wird vom PEARL-Laufzeitsystem ein Default-Wert angenommen. Stack-Overflow wird vom Laufzeitsystem erkannt und gemeldet.

PEARL-DOS unterstützt alle DOS-Standardgeräte wie

- Console (CON)
- Serial Adapter (COM1, COM2)
- Parallel Adapter (LPT1, LPT2, LPT3)
- Harddisk und Floppy (A:, B:, C:, D:)

Um ein laufendes Programm jederzeit unterbrechen und beliebige Aktionen starten zu können, wurde ein sogenannter Software-Interrupt implementiert, der mit Hilfe von CONTROL-C und der Eingabe einer Zahl von der Console aus getriggert werden kann.

Beispiel:

```
    MODULE (MAIN);

    SYSTEM;
        SWI2:   SOFTIN (2) * 0;

    PROBLEM;

STRT:   TASK;
    WHEN SWI2 ACTIVATE DIALOG;
        .
        .
        .
    END;

DIALOG: TASK PRIO 10;
        .
        .
        .
    END;
    .
    .
    .
```

Während des Ablaufs der Applikation kann die Task DIALOG durch Eingabe von CONTROL-C und der Zahl 2 an der Console aktiviert werden.

3.4 Die offene Treiberschnittstelle für Prozeß-E/A

Für den IBM PC gibt es bereits eine Vielzahl von Interface-Karten zur Meßdatenverarbeitung mit steigender Tendenz. Es ist deshalb für Implementatoren von höheren Programmiersprachen unmöglich, Treiber für alle Interface-Karten anzubieten. Außerdem gibt es viele Anwender, die eigene Treiber oder sogar Interface-Karten einsetzen möchten. Deshalb stellt diese PEARL-Implementierung von Werum eine Schnittstelle zur Verfügung, die es erlaubt,

beliebige Treiber in das PEARL-Programmiersystem zu integrieren und mit den PEARL-Anweisungen TAKE / SEND anzusprechen.
Diese Schnittstelle ist im Detail im Benutzerhandbuch des PEARL-Programmiersystems für DOS beschrieben, deshalb wird in diesem Beitrag nur das Prinzip dargestellt.

Die Funktionen eines interrupt-gesteuerten Treibers für Prozeß-E/A lassen sich wie folgt charakterisieren:

1. Treiber-Start

   - Treiber-spezifische Variablen und Buffer initialisieren
   - Interrupt-Vektor für Interrupt-Handler setzen
   - Interface-Karte initialisieren.

2. Treiber-E/A

   - Interface-Karte entsprechend E/A-Auftrag programmieren
   - Laufende PEARL-Task suspendieren und Prozessor für andere lauffähige Task freigeben.

3. Treiber-Ende

   - Interface-Karte deaktivieren
   - Interrupt-Vektor zurücksetzen.

4. Interrupt-Handler

   - Meßwert-Übernahme
   - Wenn gesamter E/A-Auftrag schon bearbeitet,
     dann: suspendierte PEARL-Task fortsetzen
     sonst: Interface-Karte erneut programmieren.

Für die Funktionen

- Laufende PEARL-Task suspendieren und
- Suspendierte PEARL-Task wieder fortsetzen

stellt das PEARL-Programmiersystem Routinen zur Verfügung, die aus Treibern heraus aufgerufen werden können.

Für den Data Acquisition and Control Adapter von IBM (1 Interface-Karte mit 2 analogen Eingabe- und 4 analogen Ausgabekanälen sowie je einem 16 Bit breiten binären Eingabe- und Ausgabekanal) stellt Werum einen Treiber zur Verfügung.

Im folgenden Beispiel ist die Benutzung dieses Treibers aus Anwendersicht kurz dargestellt.

Beispiel 1: Es sollen mit einer TAKE-Anwendung jeweils 100 Werte von den Analog-Kanälen 0 und 1 mit einer Abtastrate von 1 KHz eingelesen werden.

```
.
.
.
SYSTEM;
      ANALOG_IN:   DACAAI  (0)  * 0  * 1 ;
```

Ende Kanalbereich Kanal 1

Anfang Kanalbereich Kanal 0

Adapternummer

Systemname für Analogeingabe

```
.
.
.
TYPE   T_ANALOG_MW   STRUCT  [ TAB (100)  STRUCT  [ Kanal_0  FIXED,
                                                    Kanal_1  FIXED ] ];

DCL      ANALOG_MW   T_ANALOG_MW ;
.


.
TAKE     ANALOG_MW   FROM ANALOG_IN
     BY  CONTROL (200, 1000) ;
```

Abtastrate 1 KHz

Anzahl Eingaben 100 * Kanal 0 + 100 * Kanal 1

Beispiel 2: Ein Gerät kann über den binären Ausgabe-Kanal ein- bzw. ausgeschaltet werden. Über den binären Eingabe-Kanal erfolgt eine Zustandsanzeige, die Ein, Aus oder Änderung (Wechsel von einem Zustand in den anderen, wozu das Gerät einige Zeit benötigt) signalisiert.

```
SYSTEM;
     DEVICE_SET:     DACABO (0) * 0 * 7, 1 ;    /* Bit 7 */
                                    |  |
                                    |  Anzahl Bits
                                    Start-Bit
     DEVICE_STATE:   DACABI (0) * 0 * 4, 2 ;  /* Bit 4 und 5 */
   .
   .
   .
   .
DCL  DEVICE_START    BIT (1)    INIT   ('1'B) ,
     DEVICE_STOP     BIT (1)    INIT   ('0'B) ;

DCL  DEVICE_STATUS   BIT (2) ;

   .
   .
   .
   .
SEND  DEVICE_START TO  DEVICE_SET ;
   .
   .
   .
   .
TAKE  DEVICE_STATUS FROM DEVICE_STATE ;
   .
   .
   .
   .
```

4. Charakterisierung des verfügbaren PEARL-Testsystems

Für den interaktiven Test von PEARL-Programmen auf Sprachebene hat Werum einen portablen, in PEARL programmierten Debugger entwickelt /1/, der auch unter DOS verfügbar ist. Die Kommunikation mit dem Debugger kann über die Console (CON) oder COM1 erfolgen; dies ist besonders hilfreich, wenn eine Anwendung umfangreiche Dialoge mit der Console durchführt.
Dieser Debugger bietet für den Einzeltest von Programmkomponenten komfortable Display-, Trace- und Unterbrechungsmöglichkeiten (Breakpoints):

```
DISPLAY Mess.Wert
Mess-Wert := '1101'B
LINE TRACE FROM 55 TO 75 ON
CALL TRACE IN Erfassung OFF
BREAK ON 457; HALT; END
BREAK ON ENTRY Regelung; DISPLAY Mess; END
```

Das letzte Beispiel zeigt einen Breakpoint auf den Eingang der Prozedur Regelung mit einem Schnappschuß der Variablen Mess, ohne daß der Debugger in den Dialogbetrieb übergeht, wie es beim vorletzten Beispiel durch das HALT-Kommando verlangt wird.

Über diese "sequentiellen" Möglichkeiten hinaus unterstützt der Debugger auch den Integrations- und Gesamttest durch die Möglichkeiten der Diagnose und Veränderung der Zustände von Tasks und Synchronisationsvariablen sowie durch Schnittstellen zur Simulation der Prozeßperipherie:

```
STATUS TASK
STATUS TASK Erfassung
ACTIVATE Erfassung
STATUS SEMA Puffer_Sema
RELEASE Puffer_Sema
```

Durch die STATUS-Kommandos kann der Anwender gewissermaßen auf PEARL-Sprachebene in das PEARL-Betriebssystem hineinschauen. Z.B. erhält er als Antwort auf das Kommando STATUS TASK Erfassung folgende Informationen im Klartext ausgegeben:

- Aktueller Zustand der Task Erfassung
  (nicht aktiv, aktiv, laufend, blockiert, suspendiert, wartend auf I/O)
- Gegebenenfalls Zeitpunkt der Aktivierung, Blockierung, Suspendierung
- Verursacher der Aktivierung, Blockierung, Suspendierung
- Zur Ausführung anstehende Anweisung mit Rückverfolgung bei (geschachtelten) Prozeduraufrufen.

Der Anwender kann nun den Zustand einer Task interaktiv mit den PEARL-Tasking- und Synchronisier-Anweisungen ändern (ACTIVATE, TERMINATE, SUSPEND, CONTINUE, PREVENT, REQUEST, RELEASE, ENTER, LEAVE, RESERVE, FREE).

5. Ausblick, PEARL unter OS/2

Sobald OS/2 auf dem Markt verfügbar ist, wird Werum sein portables PEARL-Programmiersystem auch unter OS/2 implementieren. Da OS/2 ein Echtzeit-Betriebssystem ist, fallen die in Kapitel 2 aufgeführten Einschränkungen natürlich weg und PEARL wird, wie in der Sprachdefinition angegeben, verfügbar sein.
Unter OS/2 können bis zu 16 Programme (jedes Programm kann wiederum mehrere Tasks (Threads) enthalten) gleichzeitig aktiv sein. Ein PEARL-Programm wird wie ein normales OS/2-Programm gestartet. Das in OS/2 vorhandene PIPE-Konzept wird direkt aus PEARL heraus benutzbar sein, so daß verschiedene PEARL-Programme über PIPES miteinander kommunizieren können.
Natürlich wird auch der PEARL-Debugger verfügbar sein sowie das Echtzeit-Datenbanksystem BAPAS-DB, das über eine SQL-Schnittstelle ansprechbar sein wird.
In etwa 1 bis 2 Monaten nach Verfügbarkeit von OS/2 werden das PEARL-Programmiersystem sowie BAPAS-DB verfügbar sein.

Literatur

/1/ Brunner, P.J.; Meffert, K.; Windauer, H.:
PEARL-Testsystem. PEARL Rundschau, Band 2, Nr. 6, Dezember 1981, S. 8 – 13.

/2/ DIN 66253, Teil 1:
Programmiersprache PEARL. Basic PEARL. Beuth Verlag, Berlin 1981.

/3/ DIN 66253, Teil 2:
Programmiersprache PEARL. Full PEARL. Beuth Verlag, Berlin 1982.

/4/ Kneuer, E.:
PEARL-PC – Ein Echtzeit-Programmiersystem für IBM PCs. Workshop Personal Realtime Computing, München-Neubiberg, September 1986.

/5/ Werum, W.; Windauer, H.:
Introduction to PEARL. Vieweg 1985, 3. Auflage.