Zur Verwendung von PEARL bei der Programmierung von Knotenrechnern in dedizierten verteilten Systemen

Dr. W. Röhrl

G P P Gesellschaft für
Prozessrechnerprogrammierung mbH
Kolpingring 18a
8024 Oberhaching
Tel.: 089/ 61 30 4-1

Zusammenfassung:

Ein PEARL-System, mit dem Mikroprozessor-Systeme in spezialisierten lokalen Rechnernetzen (dedizierte verteilte Systeme) programmiert werden sollen, muß aufgrund der speziellen Eigenschaften der Zielrechner bestimmte Anforderungen genügen; dies trifft sowohl auf die Basis-Software - Betriebssystem und PEARL-Laufzeitsystem - als auch auf das Übersetzungssystem - Compiler-Oberteil und Code-Generator - zu. Vor allem folgende Problemkreise sind hierbei zu berücksichtigen: Laden des Zielrechners, RAM-ROM-Aufteilung, Echtzeit-Verhalten, Konfigurierbarkeit der Basis-Software.

1. Dedizierte verteilte Systeme

Durch das Aufkommen billiger Mikroprozessor-Systeme werden seit einiger Zeit verteilte Daten-Verarbeitungs-Systeme (VS) realisiert; dies sind Systeme, bei denen Komponenten eines Programmsystems auf mehrere, zu einem Netz zusammengeschaltete Rechner verteilt sind. Die Vorteile eines derartigen Systems wurden bereits vielerorts diskutiert und sollen nur noch summarisch genannt werden:

- erhöhte Rechenleistung durch parallel operierende Knoten-Rechner und die Möglichkeit der Lastverteilung
- erhöhte Ausfall-Sicherheit durch elektrische Trennung und redundante Auslegung der einzelnen Rechner
- Unterstützung eines modularen Hardware- und Software-Konzepts, so daß einzelne Komponenten möglichst rückwirkungsfrei modifiziert werden können.

Im allgemeinen besteht ein VS also aus einer Reihe von Knoten-Rechnern (KR), die über irgendein Kommunikations-Medium (Lichtleiter, Draht-Verbindung, Funkstrecke usf.) zu einem Netz verbunden sind. Im folgenden soll nun eine Untergruppe von VS betrachtet werden, die als dediziertes VS bezeichnet werden soll und gegenüber dem allgemeinen VS in etwa so abgegrenzt werden kann:

- Das Rechner-Netz besitzt eine nur lokale Ausdehnung, die in der Größen-Ordnung von wenigen Metern bis hin zu einigen Kilometern liegen kann. Typische Beispiele wären eine Fabrikhalle, in der das VS zur Fertigungs-Steuerung verwendet wird, oder spezielle rechnergestützte Nahbereichs-Kommunikations-Systeme. Weiterhin soll das Netz eines dedizierten VS eine relativ einfache Topologie besitzen: dies ist dann gegeben, wenn je zwei KR unmittelbar Nachrichten austauschen können und dies nicht über dazwischenliegende KR, die nur als transiente Knoten fungieren, tun müssen (Ring-Netze, Punkt-Punkt-Verbindungen, Stern-Netze etc.). Aufgrund der Lokalität der Netze müssen die einzelnen KR nicht über globale öffentliche Kommunikations-Kanäle verbunden werden und sind damit auch nicht gezwungen,

aufwendige Protokolle wie das ISO/OSI-Schichtenmodell zur Inter-KR-Kommunikation zu realisieren.

- Das Rechner-Netz dient zur Bearbeitung einer fest umrissenen Aufgabe. Dies bedeutet negativ, daß die einzelnen KR keine Vielzweck-Rechner mit umfangreicher Software-Ausstattung sind. Vielmehr kann ein KR auf eine bestimmte Aufgabe derart spezialisiert sein, daß er nur im Verbund mit anderen KR, d.h. nur als Komponente eines VS sinnvoll arbeiten kann (eingebettetes System). Als Beispiel seien Meßwert-Abnehmer, die eine Vorverarbeitung und Reduktion der Meßdaten durchführen, primär als 'Rechenknechte' ausgelegte KR, oder KR, die eine komfortable Mensch-Maschine-Schnittstelle anbieten, genannt. Derartige spezialisierte KR werden meist als Mikroprozessor-Systeme realisiert. Die Spezialisierung der KR bezieht sich aber nicht nur auf die Software, sondern oft auch auf die eingesetzte Hardware. So kann ebenso Spezial-Peripherie wie auch spezielle Prozessoren (EA-Prozessoren, Arithmetik-Prozessoren, Signal-Prozessoren) zum Einsatz kommen. Hier zeigt sich auch ein Vorteil von VS: die Änderung der Hardware-Ausstattung eines KR kann für die anderen KR völlig transparent bleiben - ebenso auch ein Wechsel von Basis- oder Anwender-Software im KR. Schließlich sei noch darauf hingewiesen, daß KR, die zur Verarbeitung von externen Ereignissen dienen, i.d.R. als Realzeitsystem betrieben werden müssen.

2. Die Programmierung von KR in einem dedizierten VS

Es soll nun betrachtet werden, welche Anforderungen an ein PEARL-System zu stellen sind, mit dem KR in einem dedizierten VS im eben umrissenen Sinn programmiert werden sollen. Es sei betont, daß es bei diesen Überlegungen nicht darum geht, Gründe für die Wahl von PEARL anzuführen (hierfür gibt es projektspezifisch jeweils eine Reihe von Gründen sowohl technischer als auch nicht-technischer Art); vielmehr soll lediglich dargestellt werden,

welchen Anforderungen ein PEARL-System genügen sollte/müßte, wenn mit dessen Hilfe KR i.o.S. zu programmieren sind. Grundsätzlich gilt aber, daß eine prozess-orientierte Sprache wie PEARL eine für die Programmierung von KR gut geeignete Sprache ist, wenn KR im hier besprochenen Sinne für die Bearbeitung von Realzeit-Aufgaben eingesetzt werden.
Die zu programmierenden KR sind durch folgende Eigenschaften charakterisiert:

- Spezialisierung des KR auf eine Aufgabe. Dies bedeutet, daß die KR meist nur als Ziel- oder Ausführungs-Rechner und nicht als Entwicklungs-Rechner anzusehen sind. Das Programmsystem für einen KR wird also zunächst auf einem Gast-Rechner erstellt; anschließend wird es in ablauffähiger Form in den KR gebracht und in diesem ausgeführt. Die Trennung von Gast- und Ziel-Rechner hat den positiven Nebeneffekt, daß das System, auf dem die Programme entwickelt werden, nicht auch die Eigenschaften des Zielsystems haben muß: der Entwicklungs-Rechner muß z.B. nicht unbedingt ein Echtzeit-Betriebssystem besitzen. Umgekehrt kann das Zielsystem klein und dediziert bleiben, da es nicht Hard- und Software für eine Entwicklungs-Umgebung bereitstellen muß.

- Spezial-Peripherie. Dedizierte KR müssen oft auf spezielle Umwelt-Ereignisse reagieren und können daher besondere Ein- und Ausgabe-Mechanismen besitzen. Diese externen Schnittstellen sind zumindest z.T. nicht durch Standard-Treiber bedienbar, sondern benötigen spezielle Treiber-Programme zu ihrem Betrieb.

- Eingeschränkte Hardware-Ausstattung. Dedizierte KR besitzen nicht unbedingt die komplette Standard-Peripherie wie Externspeicher, Terminal oder Standard-Hardware wie Arithmetik-Prozessor etc. Die Gründe für diese Einschränkungen mögen etwa in besonders rauhen Umweltbedingungen oder dem spezialisierten Aufgabenbereich des Rechners liegen. Die eingeschränkte Hardware-Ausstattung des KR führt zu dem Wunsch, auch nur eingeschränkte Basis-Soft-

ware einzusetzen. Es macht etwa wenig Sinn, im Betriebssystem ein Filesystem, einen Lader etc. vorzusehen, wenn der KR keinen Extern-Speicher besitzt und das gesamte Programmsystem des KR im ROM untergebracht ist. Außerdem führt die Minimierung von Hard- und Software-Einsatz meist zu einer höheren Betriebssicherheit des KR.

- Der KR muß oft unter Realzeitbedingungen operieren. KR werden etwa zur Steuerung technischer Prozesse, zur Messwert-Erfassung usw. eingesetzt. In diesen Einsatzgebieten müssen die KR oft Echtzeit-Verhalten zeigen, da die Antwortzeit des Systems einen bestimmten Maximalwert nicht überschreiten darf und das System eine bestimmte Rate von Ereignissen in einer vorgegebenen Zeit bearbeiten können muß.

2.1 Die Überführung von Programmen in das Zielsystem

Im folgenden seien einige Hinweise zur Basis-Konfiguration Entwicklungs-/Zielsystem gegeben, wenn die zu programmierenden KR kein Entwicklungs-Betriebssystem besitzen. In diesem Zusammenhang stellt sich vor allem die Frage, wie das auf dem Gast-Rechner entwickelte Programm in den Ziel-Rechner überführt werden soll. Hierbei sind mindestens drei Varianten denkbar:

1) Der KR besitzt keinen ROM-Bereich mit Basis-Software. In diesem Fall ist es notwendig, das entwickelte Programmsystem mithilfe eines weiteren Systems in den Speicher des KR zu überführen und dort zur Ausführung zu bringen. In der Testphase kann das z.B. eine Anordnung sein wie der In-Circuit-Emulator (ICE) der Fa. Intel.
   Anmerkung: Bei der eben geschilderten Variante muß es möglich sein, die gesamte Basis-Software (einschließlich Betriebssystem, Treibern, Selbsttest-Programm usw.) mit dem PEARL-Programm zusammen zu binden.
2) Der KR besitzt einen ROM-Bereich mit einer Lade-Funktion. Durch diese Lade-Funktion wird man vom Vorhandensein zusätzlicher Systeme wie eines ICE unabhängig. Es wird

jedoch nun eine externe Schnittstelle vorausgesetzt, über die das Programmsystem in den KR geladen werden kann. Es sei darauf hingewiesen, daß die Lade-Funktion unter bestimmten Voraussetzungen sehr einfach sein kann (und damit wenig Raum im Ziel-Rechner einnimmt):

Sämtliche Adreß-Bezüge im zu ladenden Programmsystem sind bereits aufgelöst und verabsolutiert (beim Intel-8086-System etwa nach dem Lauf der Programme LINK86 und LOC86).

Das zu ladende Objekt-Programm ist in einer einfachen internen Form dargestellt; außerdem wird es nicht in binärer, sondern in Alpha-Form mit Prüfsumme übertragen (bei Intel etwa nach dem Lauf des OH86). Die einfache interne Form garantiert eine wenig aufwendige Interpretation durch die Lade-Funktion im KR. Die Darstellung in Alpha-Zeichen erlaubt es, das Objekt-Programm über eine Standard-Schnittstelle des Gast-Rechners (z.B. Terminal-Schnittstelle) in den Ziel-Rechner zu 'kopieren'.

Die Schnittstelle, über die die Programme geladen werden, sollte einfach zu bedienen sein, wie dies etwa bei einer seriellen Leitung der Fall ist.

Anmerkung: Auch hier gilt wieder, daß die Basis-Software mit dem Anwender-Programm gebunden und geladen werden muß.

3) Der KR besitzt einen ROM-Bereich mit einer Lade-Funktion und der gesamten Basis-Software. Der Vorteil dieser Variante gegenüber der vorherigen besteht darin, daß weniger Code und Daten zu laden sind. Die Schnittstelle zwischen Anwender- und Basis-Software muß nunmehr so ausgelegt sein, daß die Anwender-Software ohne die Basis-Software gebunden werden kann. Die Anwender-Software selbst ist wieder über eine Lade-Funktion oder über ein System wie den ICE zu laden.

Für alle Varianten gilt, daß der KR während der Testphase im wesentlichen nur RAM-Speicher enthält. Nach der Testphase werden die invarianten Teile des Programms in einem ROM-Speicher festgehalten und dieser in den KR gebracht.

2.2 Anforderungen an die Basis-Software im Zielsystem

Aus den bisherigen Überlegungen läßt sich ein Forderungs-Katalog für die im KR eingesetzte Basis-Software ableiten.

- Die Basis-Software muß die Aufteilung in RAM/ROM-Bereiche unterstützen. Dies bedeutet, daß das System mit beliebigen RAM-Inhalten gestartet werden kann (Code und invariante Daten sind im ROM abgelegt). Werden bestimmte RAM-Daten vorausgesetzt, so sind diese RAM-Bereiche explizit zu initialisieren.

- Die Basis-Software sollte Realzeit-Verhalten zeigen. Ein PEARL-Programm kann nur dann als Echtzeit-System arbeiten, wenn dies auch durch die Basis-Software (und ggfs. Hardware) unterstützt wird. Grundsätzlich gilt, daß auf einer Echtzeit-Basis-Software sowohl Echtzeit- als auch Nicht-Echtzeit-Programme ablaufen können.

- Ein wichtiges Problem in einem VS besteht in der Gleichheit der Uhrzeit in allen KR. Aus Gründen der Flexibilität ist es durchaus wünschenswert, die Anwender-Software am Verteilen der Uhrzeit teilhaben zu lassen bzw. die aktuelle System-Zeit über die Anwender-Software im System zu verteilen. Die Basis-Software muß daher Schnittstellen zum Lesen und Setzen von Uhrzeit (und Datum) bereitstellen. Hierbei ist es wichtig, daß das Verändern der aktuellen Uhrzeit einen genau definierten Einfluß auf aktuell vorliegende PEARL-Schedules besitzt (etwa kein Einfluß auf relative Schedules wie "AFTER 10 SEC RESUME"). Weiterhin dürfte es bei der Bearbeitung von harten Echtzeit-Aufgaben oft hilfreich sein, wenn der Anwender die Zeit-Auflösung in der Basis-Software vorgeben kann. Dies beinhaltet sowohl die Konfigurierung der Zeitauflösung im Betriebssystem-Kern (Interruptrate der Software-Uhr), als auch die Festlegung der dem PEARL-System zugrundeliegenden Zeit-Einheit.

- Die Basis-Software sollte in ihrer Größe konfigurierbar sein. Wird etwa in einem PEARL-Programm kein Bezug auf eine Alphic-Dation genommen, so kann z.B. das Laufzeit-Paket der Formatierten Ein/Ausgabe entfallen. Diese Forderung läßt sich oft schon über den Binder erfüllen (es werden nur diejenigen Teile des Systems gebunden, die auch über die PEARL-Quelle(n) referenziert werden). Ebenso sollte auch das Betriebssystem konfigurierbar sein: ist in dem KR etwa keine Druckerschnittstelle vorhanden, so muß kein Drucker-Treiber im System vorhanden sein. Die Forderung nach Konfigurierbarkeit trägt der Tatsache Rechnung, daß KR für die Erfüllung einer bestimmten Spezial-Aufgabe programmiert werden müssen, der KR jedoch nur mit wenig Speicher versehen ist.

- Wenn ein KR spezielle Peripherie bedienen soll, so kann es notwendig werden, in das System eigene Treiber einzubringen. Die Basis-Software sollte eine genau definierte Schnittstelle besitzen, über die Treiber in das System eingebunden werden können.

- Eine Bemerkung zur Portabilität der Basis-Software: Ein VS besitzt u.a. den Vorteil, daß es möglich ist, die Hardware eines KR zu ändern, ohne daß dies Auswirkungen auf die restlichen KR besitzt. Dieser Vorteil kann dann wirklich ausgenutzt werden, wenn die im KR eingesetzte Basis-Software (möglichst) portabel ist, da sie dann bei (weitgehender) Konservierung ihrer alten Eigenschaften auf die neue Hardware übertragen werden kann.

2.3 Anforderungen an das PEARL-Übersetzungssystem

Zu den eben angeführten Forderungen an die Basis-Software gibt es auch entsprechende Forderungen an das PEARL-Übersetzungssystem (Compiler-Oberteil und Code-Generator).

- Der Code-Generator des Übersetzungssystems muß die RAM-ROM-Aufteilung unterstützen. Außerdem sollte der PEARL-Anwendungs-Programmierer die Möglichkeit besitzen, die Plazierung seiner Daten in das RAM bzw. ROM steuern zu können; andernfalls müßten sämtliche Daten per PEARL-Code initialisiert werden - also auch während des Programmlaufs invariante Daten.

- Zur Unterstützung der Echtzeit-Programmierung sollte die Einbindung von Assembler-Programmen genau spezifiziert sein (Prozedur-Schnittstelle mit Beschreibung der Ablage der einzelnen PEARL-Objekte). Damit können besonders zeitkritische Programm-Teile in Assembler geschrieben werden. Weiterhin besteht die Möglichkeit, über ein Assembler-Interface Programme, die in einer dritten Sprache geschrieben sind, anzuschliessen.

- Der generierte Code sollte ggfs. durch Optionen beeinflußbar sein. Derartige Optionen sind jedoch extrem maschinenabhängig, so daß hier keine allgemeine Aussagen gemacht werden können.

- Im Rahmen des Übersetzungssystems ist zwischen zwei Arten von Portabilität zu unterscheiden:
  - .. Das Übersetzungssystem kann selbst auf einem anderen Rechner zum Ablauf gebracht werden. Dies kann Vorteile bei der Programm-Entwicklung bringen, da die Entwicklungs-Umgebung nicht an einen Maschinen-Typ fixiert ist.
  - .. Das Übersetzungssystem kann Code für eine verschiedene Ziel-Maschinen generieren. Portabilität in diesem Sinn meint, daß das Übersetzungssystem (insbesondere der Code-Generator) mit möglichst geringem Aufwand an eine

neue Ziel-Maschine angepaßt werden kann. Die bereits erwähnte Portabilität der Basis-Software kann nur dann genutzt werden, wenn auch das Übersetzungssystem Code für die neue Ziel-Maschine generieren kann.

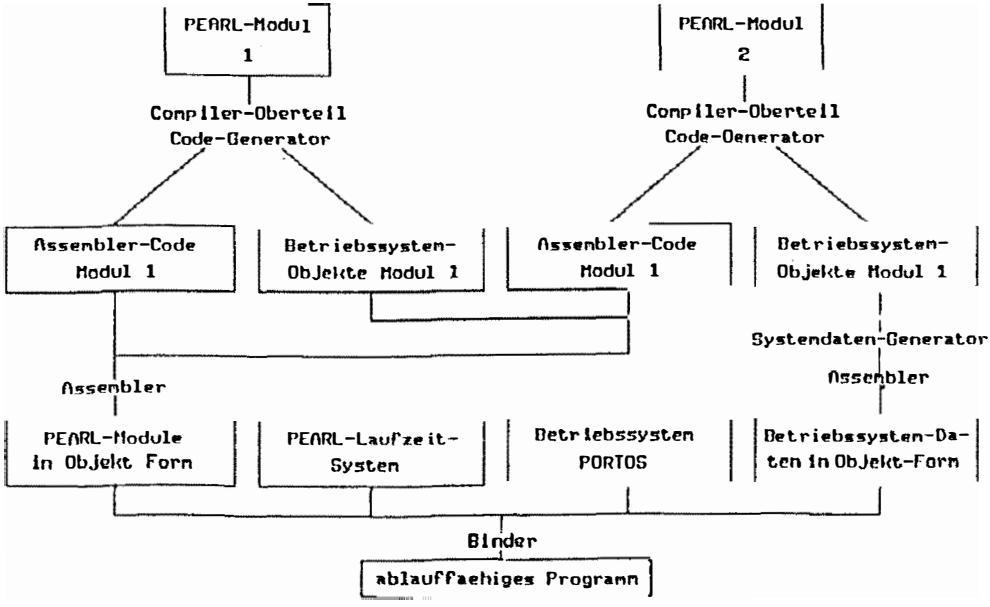
2.4 Zur Kommunikation zwischen den KR

Ein VS setzt Kommunikation zwischen den einzelnen KR unabdingbar voraus. Es ist zu beachten, daß es sich bei dedizierten VS im angegebenen Sinn nur um lokale Netze mit einer einfachen Topologie handelt. Daher ist es nicht notwendig, in einem KR etwa das gesamte ISO/OSI-Schichtenmodell zu realisieren. Vielmehr dürfte sich folgende Vorgehensweise anbieten:
Durch einen in die Basis-Software eingebrachten Treiber werden Datenblöcke mit einer bestimmten maximalen Länge (Pakete) an einen gewünschten KR gesandt. Zum Treiber gehört ein Fehler-Management, durch das das mit Fehlern behaftete reale Übertragungs-Medium virtuell fehlerfrei wird. Im Treiber werden damit die Ebenen 1 und 2 des ISO/OSI-Modells realisiert (z.B. HDLC-Prozedur). Statt eines Treibers ist etwa z. B. an den Anschluß eines ETHERNET-Moduls zu denken; diesem werden die Daten-Pakete mitsamt der Empfänger-Identifikation übergeben. Aufgrund der einfachen Netz-Topologie werden Daten-Pakete direkt beim Empfänger abgeliefert; damit entfällt die Netzwerks-Ebene (Routing) des ISO/OSI-Modells.
Besonders bei kleinen KR kann es aus Speicherplatz-Gründen ratsam sein, nicht mehr als die Übertragungs- und Leitungs-Ebene im Bereich der Basis-Software zu realisieren. Die Kommunikations-Struktur zwischen den einzelnen KR sollte so ausgelegt sein, daß eine zu übertragende Nachricht nicht größer als das zwischen den Rechnern übertragbare Daten-Paket ist, da damit auch die Transport-Ebene entfallen kann. In größeren KR-Systemen bzw. bei höheren Anforderungen an die Kommunikation sollte im Bereich der Basis-Software ein Botschaften-System vorliegen, das eine höhere Art der Inter-KR-Kommunikation erlaubt. Es ist aber auch durchaus möeglich, im Bereich der Anwender-Software auf der Basis der Übertragung einzelner Pakete ein komfortableres und auf die

jeweiligen Bedürfnisse abgestelltes Kommunikations-System zu realisieren. Dafür ist übrigens PEARL als prozess-orientierte Sprache sehr gut geeignet (Implementierung des Kommunikations-Prozesses im KR).
Schließlich sei noch bemerkt, daß eine weitere Vereinfachung und Vereinheitlichung der Kommunikation zwischen KR in einem "Mehrrechner-PEARL"-System erreicht werden könnte.

3. Überblick über das GPP-PEARL-System für die Intel-8086-Familie

Das GPP-PEARL-System für die Intel-8086-Familie ist für die Programmierung von Mikroprozessor-Systemen konzipiert, die höchstens mit einer Lade-Funktion ausgestattet sind. Derartige Mikroprozessor-Systeme können als KR in einem dedizierten VS verwendet werden. Das GPP-PEARL-System besitzt folgende Struktur:



Jeder PEARL-Modul wird durch den Compiler-Oberteil in eine maschinen-unabhängige Zwischensprache übersetzt, die wiederum durch den Code-Generator in einen Modul mit Assembler-Code der entsprechenden Ziel-Maschine (Intel-8086) überführt wird. Gleichzeitig erzeugt der Code-Generator zu jedem PEARL-Modul einen Hilfs-Modul, in dem die in dem PEARL-Modul deklarierten Betriebssystem-Objekte aufgelistet sind (Tasks, Semaphore, Dations etc.). Sind nun alle PEARL-Quellen übersetzt, so wird durch den Systemdaten-Generator auf der Basis der in den Hilfs-Moduln abgelegten Informationen ein Assembler-Modul mit den zu dem PEARL-Programm gehörigen Betriebssystem-Objekten erzeugt. Anschließend sind sämtliche Assembler-Moduln zu übersetzen und mit dem Betriebssystem PORTOS und dem PEARL-Laufzeitsystem zu einem ablauffähigen Programm zu binden. Der Mechanismus der Systemdaten-Generierung gestattet es, aus den PEARL-Quellen völlig automatisch ein ablauffähiges Programm zu erzeugen. Eine Änderung der Zahl der Betriebssystem-Objekte ist damit lediglich auf eine Änderung des PEARL-Quell-Programms beschränkt.

Abschließend seien einige Eigenschaften des GPP-PEARL-Systems in Bezug auf die in 2.2. und 2.3. aufgeführten Kriterien genannt:

- Das PEARL-Übersetzungssystem unterstützt die RAM/ROM-Aufteilung, wobei der PEARL-Anwendungs-Programmierer die Ablage von Daten in RAM- oder ROM-Bereich über das INV-Attribut in der PEARL-Quelle steuern kann. Ebenso ist auch die Basis-Software auf die RAM/ROM-Aufteilung hin ausgelegt.

- Das PEARL-Übersetzungssystem ist einerseits portabel in dem Sinn, das es auf verschiedenen Rechnern lauffähig ist. Andererseits muß wegen der Übersetzung in eine maschinenunabhängige Zwischen-Sprache bei einer Änderung der Ziel-Maschine lediglich der Code-Generator modifiziert werden. Zur Portabilität des Systems trägt auch bei, daß die Basis-Software portabel ist, da diese zu einem großen Teil ebenso wie das Übersetzungssystem in einer portablen höheren Sprache geschrieben ist.

- Der Code-Generator kann durch Optionen gesteuert werden, über die z.B. der 8086-Adressierungs-Mechanismus festgelegt wird.

- Das Betriebssystem PORTOS ist ein Echtzeit-Betriebssystem, das kurze Reaktionszeiten auf externe Ereignisse garantiert (maximale Interrupt-Latenzzeit des Betriebssystem-Kerns beim 8086-Prozessor ca. 50 µsec). Außerdem besitzt der Betriebssystem-Kern eine genau definierte Treiber-Schnittstelle, mithilfe derer neue Geräte-Treiber in das Basis-Software eingebracht werden können.

- Die durch das Betriebssystem angebotenen Dienste sowie andere Betriebssystem-Parameter (etwa die Zeitauflösung) können über einen Betriebssystem-Konfigurator bequem modifiziert werden.

- Die Basis-Software bietet Schnittstellen zum Lesen und Setzen der Uhrzeit. Beim Setzen der Uhrzeit werden außerdem laufende PEARL-Schedules derart berichtigt, daß relative Schedules (wie "AFTER 10 SEC RESUME") vom Setzen der Uhr unberührt bleiben, während absolute Schedules (wie "AT 24:00:00 ACTIVATE TASK0) hinsichtlich der neuen Uhrzeit berichtigt werden.