Programmentwicklung für RTOS-UH mit PE

Version 1.6-G vom 11. Dezember 2006

Das Programm PE dient der komfortableren Erstellung von PEARL- oder Assembler-Programmen für das Echtzeitbetriebssystem RTOS-UH insbesondere bei der Crossentwicklung. Anstelle des Compilers oder Assemblers wird vom Anwender das Programm PE aufgerufen. Dieses generiert mit Hilfe verschiedener Kommandozeilenparameter und Umgebungsvariablen den Aufruf des PEARL-Compilers oder des Assemblers. Sollten bei der Compilation Fehler auftreten, so werden die Fehlermeldungen aus dem erzeugten Listing in ein benutzerdefiniertes, einzeiliges Format umgewandelt und auf StdOut oder in einer Datei ausgegeben. Die Ausgaben des aufgerufenen Programms werden dabei je nach Kommandozeilenparameter unterdrückt. Mit Hilfe von Umgebungsvariablen lassen sich die Default-Ein-/Ausgabepfade auch individuell für einzelne Projekte festlegen.

IRT Institut für Regelungstechnik
Leibniz Universität Hannover

Inhaltsverzeichnis

1 Eigenschaften

1.1 Einzeilige Fehlermeldungen

PE konvertiert nach dem automatischen Aufruf des gewünschten Compilers, Assemblers oder, bei Cross-Entwicklung, des VCPs (virtueller Code-Prozessor) die mehrzeiligen Fehlermeldungen in einzeilige Ausgaben auf StdOut oder in eine Datei, um insbesondere bei der Crossentwicklung eine Anpassung an komfortable Editoren und Entwicklungsumgebungen mit automatischer Cursorpositionierung auf den Compilerfehlern zu ermöglichen. Da sich die von den Editoren verwendeten Fehlerformate unterscheiden, bietet PE die Möglichkeit zur freien Konfiguration des gewünschten Formats (s. Abschnitt 2).

Die normale Fehlerausgabe des PEARL Compilers mit der Option `lo=no` bei Fehlern in der Datei `C:/Daten/Rtos-UH/regler.pq` und in weiteren Include-Dateien hat beispielsweise die Form:

```
68xxx-Maxi-RTOS-PEARL90-15.9-J  <c>1998 W.Gerth   19-02-2000  15:22:11

a    5 D=X;
<ERROR>   *
/Undefined/
=   12 D=D***3;
<ERROR>     *
/Syntax violation/
a   10 SEND D TO AOUT;
<ERROR>              *
/Undefined/


MISSING PROC/TASK:
(***)PARAM

VAR(RAM):0000-001F  CODE(RAM):0020-009C

$009C BYTES  (FOR 68000) (SIZE-LIMIT-ERROR!)   TOTAL:    5 ERRORS.
```

Diese Fehlerliste ist aufgrund der mehrzeiligen Fehlermeldungen schwer von komfortablen Editoren, die bei der Crossentwicklung zur Verfügung stehen, auszuwerten. Eine automatische Cursorpositionierung auf den nächsten Programmfehler ist daher nicht möglich. Insbesondere fehlen die Namen der Include-Dateien, in denen ein Fehler aufgetreten ist. Dieser Umstand erschwert die Analyse der Fehlermeldungen auch bei der Programmentwicklung unter RTOS-UH erheblich. Die Ausgabe des PE lautet hingegen (abhängig von der Konfiguration):

```
68xxx-Maxi-RTOS-PEARL90-15.9-J  <c>1998 W.Gerth   19-02-2000  15:35:02
C:/Daten/Rtos-UH/MESS.PQ(5,4): Undefined
C:/Daten/Rtos-UH/regler.pq(12,6): Syntax violation
C:/Daten/Rtos-UH/STELL.PQ(10,15): Undefined
C:/Daten/Rtos-UH/regler.pq(1,1): MISSING PROC/TASK: (***)PARAM
C:/Daten/Rtos-UH/regler.pq(1,1): SIZE-LIMIT-ERROR ($009C Bytes)
$009C BYTES  (FOR 68000) (SIZE-LIMIT-ERROR!)   TOTAL:    5 ERRORS.
```

Die einzelnen Fehlermeldungen enthalten den Dateinamen, die Zeilen- und die Spaltennummer sowie die Fehlerbeschreibung. Das Format ist frei konfigurierbar. Eine Auswertung zur automatischen

Cursorpositionierung in entsprechenden Editoren stellt somit kein Problem dar. Das `SETLINE`-Kommando des PEARL-Compilers wird dabei berücksichtigt: Die Zeilennummer in der Fehlerbeschreibung bezieht sich daher stets auf die Quelldatei und nicht auf die Compilerliste.

1.2 Prüfung der Notwendigkeit des gewünschten Compilerlaufs

PE legt (mit der Option `-lif`, s. Abschnitt 3) bei der Auswertung der erzeugten Compilerliste die Datei Quelldateiname`.lif` mit einer Liste aller Include-Dateien und ihrer Verschachtelungen an. Als Beispiel ist die LIF-Datei (LIF, *L*ist of *I*nclude *F*iles) der Quelldatei `d:/rtos-uh/pq/TEST.PQ` nachstehend dargestellt.

```
<d:/rtos-uh/pq/TEST.PQ>
 |
 +--<TEST2.PQ> d:/rtos-uh/pq/TEST2.PQ
 |   |
 |   +--<TEST3.PQ> d:/rtos-uh/pq/TEST3.PQ
 |   |   |
 |   |   +--<TEST4.PQ> d:/rtos-uh/pq/TEST4.PQ
 |
 +--<TEST3.PQ> d:/rtos-uh/pq/TEST3.PQ
 |   |
 |   +--<TEST4.PQ> d:/rtos-uh/pq/TEST4.PQ
 |
 +--<TEST4.PQ> d:/rtos-uh/pq/TEST4.PQ
```

In der ersten Zeile ist die Quelldatei eingetragen, in den weiteren die Include-Dateien. Bei den Include-Dateien steht jeweils der im PEARL- oder Assemblerprogramm angegebene Pfad und/oder Name zwischen den Klammern $<$ und $>$. Dahinter folgt der absolute Pfad der beim Compilerlauf gefundenen Include-Datei.

Anhand dieser Datei überprüft PE (mit Option `-lif`, s. Abschnitt 3) vor dem Aufruf des Compilers/Assemblers die Quelldatei und alle Include-Dateien. Die Quelldatei wird anschließend nur übersetzt, falls

- die Quelldatei oder eine Include-Datei ein jüngeres oder das gleiche[1] Datum wie die Datei mit dem S-Record aufweist,
- der S-Record noch nicht existiert (nach einem Compilerlauf mit Fehlern wird die S-Record-Datei von PE gelöscht),
- die zwischen den Klammern $<$ und $>$ angegeben Include-Dateien beim Überprüfen zu anderen absoluten Angaben führen als in der LIF-Datei angegeben,
- die Quellatei mit absoluter Pfadangabe nicht mit dem Eintrag in der ersten Zeile der LIF-Datei übereinstimmt,
- noch keine LIF-Datei vorhanden ist.

Es zu beachten, dass z. B. der Austausch einer älteren mit einer neueren Include-Datei , die aber ein älteres Datum als der zuletzt erzeugte S-Record aufweist, **nicht** zu einem neuen Übersetzungslauf beim Aufruf des PE führt.

Die Abhängigkeiten der Quelldatei von Include-Dateien wird bei Verwendung der Option `-lif` zudem im Makefile-Format in die Datei Quelldateiname`.dep` geschrieben. Diese Datei lautet für obiges Beispiel:

[1] Neu-Übersetzungen bei gleichem Datum sind insbesondere bei Dateisystemen mit grober Zeitauflösung (z. B. 1 Minute) unerlässlich.

```
TEST.PQ: \
    d:/rtos-uh/pq/TEST2.PQ \
    d:/rtos-uh/pq/TEST3.PQ \
    d:/rtos-uh/pq/TEST4.PQ \
    d:/rtos-uh/pq/TEST3.PQ \
    d:/rtos-uh/pq/TEST4.PQ \
    d:/rtos-uh/pq/TEST4.PQ
```

1.3 Umgebungsvariablen

PE unterstützt zudem einige Umgebungsvariablen mit Standardpfaden für Quell-, Code- und List-Dateien, die für den Aufruf des gewünschten Compilers ausgewertet werden. Diese lassen sich für einzelne Projekte individuell anpassen (s. Abschnitt 4).

1.4 Einschränkungen

Da PE stets eine vollständige, mit der Quelldatei übereinstimmende Liste zur Auswertung der Fehlerbeschreibungen benötigt, führt die Verwendung der Compileranweisungen `/*-L*/` und `/*+P*/` sowie der Assemblerdirektive `PRINT 0` zu einer fehlerhaften Arbeitsweise des PE aufgrund von Unterschieden zwischen Liste und Quelldatei. Insbesondere sind in Include-Dateien, die z.B. allgemeine Deklarationen enthalten und für viele verschiedene Programme eingesetzt werden, häufig `/*-L*/` bzw. `PRINT 0` enthalten. Diese Include-Dateien sollten daher vor der Verwendung des PE nach den genannten Anweisungen durchsucht werden.

Die Compileranweisung `SETLINE` muss in einer Zeile angegeben werden. Eine für den PEARL90-Compiler ebenfalls mögliche `SETLINE`-Anweisung über zwei Zeilen wird vom `PE` nicht erkannt.

2 Installation

2.1 Inhalt des Archivs

2.1.1 Für Cross-Entwicklung

`pe(.exe)`	PE Programmdatei.
`pe_setup(.exe)`	Programm zur Erstellung der Konfiguration.
`cvcp(.exe)`	Virtueller Code-Prozessor (VCP), auf dem die RTOS-Compiler laufen.
`pd_ppc.vbi`	Auf dem VCP lauffähiger PEARL90-Compiler für PowerPC-Zielsysteme.
`pd_68k.vbi`	Auf dem VCP lauffähiger PEARL90-Compiler für 68K-Zielsysteme.
`readme.txt`	Kurzanleitung zur Installation sowie Lizenzinformationen.
`pedoc.pdf`	Diese Dokumentation.

Hinweis: Die im Archiv enthaltenen Compiler `pd_ppc.vbi` und `pd_68k.vbi` sind Demo-Versionen, deren Nutzung **ausschließlich für den privaten Gebrauch** gestattet ist. Die Demo-Versionen unterliegen folgenden Einschränkungen gegenüber den Vollversionen:

- Der erzeugte Code ist voll lauffähig, lediglich etwas länger und als Ergebnis eines Demo-Compilers gekennzeichnet.
- Die Compileranweisung `/*+P*/` zur Ausgabe vom erzeugten Code im Listing wird nicht unterstützt.
- Die Option `/*+M*/` funktioniert nur im normalen Modus, also ohne `MODE=NOLSTOP`.

2.1.2 Für RTOS-UH

`PE`	Transientes Kommando PE.
`PE_SETUP`	Transientes Kommando zur Erstellung der Konfiguration.
`readme.txt`	Kurzanleitung zur Installation sowie Lizenzinformationen.
`pedoc.pdf`	Diese Dokumentation.

2.2 Vorgehensweise bei der Installation

2.2.1 Windows 9x/2000/NT/ME/XP

Zur Installation unter Windows 9x/2000/NT/ME/XP[1] sind die im Archiv entahltenen Programme `pe.exe`, `pe_setup.exe` und `cvcp.exe` sowie die binären Compiler-Dateien `pd_ppc.vbi` und `pd_68k.vbi` in das gewünschte Verzeichnis zu kopieren. Das Programm `cvcp.exe` ist ein virtueller Code-Prozessor (VCP), auf dem die binären Compiler `*.vbi` laufen.

Anschließend ist `pe_setup.exe` zu starten. Dieses Programm fragt die in Abschnitt 2.3 näher erläuterten Konfigurationen von PE ab und trägt sie in die Konfigurationsdatei `pe.cfg` ein. Diese Datei wird, falls möglich, dort angelegt, wo sich auch das Programm `pe_setup.exe` befindet.

[1] Die in dieser Dokumentation genannten Markennamen und Produktbezeichnungen sind eingetragenen Warenzeichen ihrer Inhaber.

Sollte dieses nicht möglich sein, wird das aktuelle Arbeitsverzeichnis gewählt. Beim späteren Aufruf von PE wird die Datei dann im gleichen Verzeichnis wie `pe.exe` und anschließend im aktuellen Arbeitsverzeichnis gesucht.

Die erforderlichen Einstellungen können auch manuell in `pe.cfg` eingetragen bzw. dort geändert werden. Alternativ kann die Konfiguration auch in eine andere Datei eingetragen werden. Diese Datei ist beim Aufruf von PE mit dem Parameter `-cfg` anzugeben (s. Kapitel 3). Die Angabe erfolgt absolut oder relativ zum aktuellen Arbeitsverzeichnis. Die Verwendung mehrerer Konfigurationsdateien kann dann sinnvoll sein, wenn PE in verschiedenen Entwicklungsumgebungen eingesetzt wird.

2.2.2 Unix-Derivate

Die Installation unter Unix erfolgt durch Kopieren der Programme `pe`, `pe_setup` und `cvcp` sowie die binären Compiler-Dateien `pd_ppc.vbi` und `pd_68k.vbi` in das gewünschte Verzeichnis. Das Programm `cvcp` ist ein virtueller Code-Prozessor (VCP), auf dem die binären Compiler `*.vbi` laufen. Anschließend ist `pe_setup` zu starten, um die Datei `pe.cfg` anzulegen. Weitere Einzelheiten sind identisch zur Vorgehensweise unter Windows (s. Unterabschnitt 2.2.1).

2.2.3 RTOS-UH

Die S-Records `PE` bzw. `PE_SETUP` entpacken und in das gewünschte Verzeichnis kopieren. Dort stehen dann die neuen transienten Bedienbefehle `PE` bzw. `PE_SETUP` zur Verfügung. Alternativ können die S-Records auch einmalig geladen werden. Bei wiederholten Aufrufen erspart dieses das erneute Laden. Der Befehl `PE_SETUP` legt die Konfigurationsdatei `PE.CFG` im aktuellen Arbeitsverzeichnis an. Beim Aufruf von `PE` wird `PE.CFG` zuerst im aktuell gesetzten Arbeitsverzeichnis und dann in `/ED` gesucht.

Zur manuellen Erstellung von Konfigurationsdateien, auch unter anderem Namen, gelten ebenfalls die Ausführungen in Unterabschnitt 2.2.1.

2.3 Setzen des verwendeten Compilers, Assemblers oder C-VCPs

Die im folgenden angegebenen Programmnamen gelten für Windows-Systeme. Für UNIX-Systeme ist die Endung `.exe` wegzulassen.

PE benötigt den Namen des aufzurufenden Compilers. Dieser wird in der ersten Zeile der Konfigurationsdatei `pe.cfg` eingetragen. Bei Crossentwicklung ist hier der virtuelle Code-Prozessor `cvcp.exe` ggf. mit Pfadangabe anzugeben. Die benötigte Binärdatei, z.B. `pd_ppc.vbi` (mit Pfad, falls erforderlich) kann ebenfalls angegeben oder als Kommandozeilenargument übergeben werden (s. Abschnitt 3). Bei Verwendung des Tools unter RTOS-UH ist der Name des Compilers (`P`, `QP`, `AS` oder `QAS`) einzutragen. Er kann jedoch auch als erster Kommandozeilenparameter beim Aufruf von PE angegeben werden, um PE wahlwiese für einen dieser Compiler/Assembler einzusetzen. In diesem Fall muss die erste Zeile der Konfigurationsdatei leer bleiben.

- *Beispiel:*

  Folgende drei Konfigurationen führen mit den entsprechenden Kommandozeilen zum gleichen Ergebnis:

1. Die erste Zeile der Datei `pe.cfg` lautet
   ```
   c:\rtosbin\cvcp.exe
   ```
   Dann bewirkt die Kommandozeile
   ```
   pe c:\rtosbin\pd\_ppc.vbi.vbi -si=test.pq
   ```
   einen Aufruf des VCP mit der Binärdatei `pd_ppc.vbi`.

2. Der Eintrag
   ```
   c:\rtosbin\cvcp.exe  c:\rtosbin\pd\_ppc.vbi
   ```
   in der Datei `pe.cfg` in Verbindung mit der Kommandozeile
   ```
   pe -si=test.pq
   ```
   führt auf das gleiche Resultat.

3. Bleibt die erste Zeile der Datei `pe.cfg` leer, muss PE wie folgt aufgerufen werden:
   ```
   pe c:\rtosbin\cvcp.exe c:\rtosbin\pd\_ppc.vbi -si=test.pq
   ```

In allen drei Fällen wird von PE der Aufruf
```
c:\rtosbin\cvcp.exe c:\rtosbin\pd\_ppc.vbi -si=test.pq
```
erzeugt.

Soll PE bei der Crossentwicklung wahlweise für den PowerPC- oder 68k-PEARL-Compiler eingesetzt werden, so ist die Variante 1 der Konfigurationsdatei aus obigem Beispiel zu wählen und die gewünschte Binärdatei in der Kommandozeile anzugeben. Die Variante 3 wäre ebenfalls möglich, führt jedoch zu einem Mehraufwand beim Aufruf. Die Verwendung des Assemblers (`as68.vbi`) oder des Transfer-Assemblers `tapp.vbi` ist mit dem PE ebenfalls möglich. Diese sind, wie auch die Vollversionen der PEARL-Compiler jedoch nicht im freien PE-Paket enthalten.

Bei Verwendung des PE unter RTOS-UH entfällt die Angabe eines VCP. Soll wahlweise der (Quick-)Compiler oder (Quick-)Assembler verwendet werden, ist die Variante 3 der Konfigurationsdatei zu wählen und der Compiler oder Assembler in der Kommandozeile des PE anzugeben.

2.4 Wahl des gewünschten Fehlerformats

Die zweite Zeile der Datei `pe.cfg` enthält das Format der einzeiligen Fehlermeldungen. Das Setup-Programm bietet die Möglichkeit, vordefinierte Formate auszuwählen oder ein eigenes Format zu erstellen. Der Formatstring darf alle ASCII-Zeichen außer # sowie folgende Variablen enthalten.

Variable	Bedeutung
`#fname#`	Name der Datei, in der der Fehler aufgetreten ist
`#line#`	Zeilennummer des Fehlers
`#column#`	Spaltennummer des Fehlers
`#error#`	Fehlerbeschreibung (z.B. undefined)
`#code#`	Quelltextzeile, in der der Fehler aufgetreten ist

Das Format muss mindestens eine der Variablen, aber insgesamt nicht mehr als fünf Variablen enthalten.

- *Beispiel:*

  Das im Setup angegebene oder manuell in die Datei `pe.cfg` eingetragene Format lautet
  ```
  #fname#(#line#,#column#) : #error#
  ```
  Dann gibt PE bei einem Syntaxfehler in Zeile 203, Spalte 16 der Datei `test.pq` die Fehlermeldung

```
C:/RTOS-UH/Project1/test.pq(203,16) :  Syntax violation
```

auf StdOut aus.

Folgende vordefinierte Formate für den Einsatz von gängigen Editoren[2] und Entwicklungsumgebungen stehen im Setup-Programm zur Vefügung:

Nr	Format	Bemerkung
1	`#fname#:#line#: #error#`	wie gcc Ausgabe, zu verwenden für zahlreiche Editoren wie Emacs, UltraEdit etc.
2	`#fname#(#line#,#column#) : #error#`	z. B. zu verwenden für TextPad mit `^\([^(]+\)(\([0-9]+\),\([0-9]+\))` als regulärem Ausdruck für das Sprungziel.
3	`#fname#(#line#) : #error# #code#`	z. B. zu verwenden für Visual Studio 6.0.

Zu einigen Fehlermeldungen, wie z. B. `SIZE-LIMIT-ERROR`, `LABELING-ERROR` oder `MISSING PROC/TASK`, gibt der Compiler bzw. der Assembler keine Zeilennummer aus. In diesen Fällen setzt PE in der einzeiligen Fehlermeldung sowohl die Zeilen- als auch die Spaltennummer auf 1, um unmittelbar an den Anfang der Datei springen zu können.

[2] Die in dieser Dokumentation genannten Markennamen und Produktbezeichnungen sind eingetragenen Warenzeichen ihrer Inhaber.

3 Aufruf und Parameter

Aufrufsyntax

`pe [comp-prog] [comp-args] -si=<source-input> [-co=<code-output>]`
`[-lo=<list-output>][-cfg=<pe-cfg-file>][-o=<outfile>]`
`[-lo=no][-lif][-e0][-e1][-e2][-h]`

Optionen und Parameter	
`comp-prog`	Programm (VCP, Compiler, Assembler, etc.), das von PE aufgerufen werden soll, falls es nicht während des PE-Setups konfiguriert wurde (d.h., die erste Zeile der Datei `pe.cfg` ist leer). **In diesem Fall muss `comp-prog` der erste Kommandozeilenparameter sein**.
`comp-args`	Parameter für das von PE aufgerufene Programm (VCP, Compiler, Assembler etc.). Diese Parameter werden direkt an das Programm weitergegeben. Ausnahme: Die Optionen `-l0`, `-l1` und `-I` des VCP, da diese auch von PE ausgewertet und berücksichtigt werden.
`<source-input>`	Quelldatei, relativ zum aktuellen Arbeitsverzeichnis oder, falls dort nicht vorhanden, zum Inhalt der Umgebungsvariablen `PE_SOURCE` (falls gesetzt).
`<code-output>`	S-Record, relativ zum aktuellen Arbeitsverzeichnis oder, falls gesetzt, zum Inhalt der Umgebungsvariablen `PE_CODE`. Default: Quelldateiname`.sr` oder Quelldateiname`.SR`, falls Quelldateiname nur aus Großbuchstaben besteht. Der Pfad der Quelldatei wird nicht übernommen. Falls `code-output=no` gilt, wird kein S-Record erzeugt.
`<list-output>`	Listdatei, relativ zum aktuellen Arbeitsverzeichnis oder, falls gesetzt, zum Inhalt der Umgebungsvariablen `PE_LIST`. Default: Quelldateiname`.lst` oder Quelldateiname`.LST`, falls Quelldateiname nur aus Großbuchstaben besteht. Der Pfad der Quelldatei wird nicht übernommen.
`<pe-cfg-file>`	Datei mit der PE-Konfiguration absolut oder relativ zum aktuellen Arbeitsverzeichnis. Default: `pe.cfg` oder `PE.CFG` im PE Programmverzeichnis (bei RTOS-UH im Arbeitsverzeichnis oder im `/ED`).
`<outfile>`	Datei, in die PE und das von PE aufgerufene Programm ihre normalen Ausgaben schreiben. Default: `StdOut`. (Fehlermeldungen werden immer nach `StdErr` geschrieben.)
`-lo=no`	Die erzeugte Listdatei wird nach ihrer Auswertung wieder gelöscht.
`-lif`	Die Datei Quelldateiname`.lif` im gleichen Verzeichnis wie die Quelldatei wird verwendet, um die Notwendigkeit des gewünschten Compilerlaufs zu prüfen. Die Datei wird bei einem Compilerlauf neu erzeugt. (S. Abschnitt 1.2) und enthält die Struktur der Include-Dateien. In der Datei Quelldateiname`.dep` werden die Abhängigkeiten der Quelldatei von Include-Dateien zudem im Makefile-Format abgespeichert.
`-e0`	Die Kopfzeile des Compiler-/Assemblerlistings wird ausgegeben.
`-e1`	Die Ausgaben des aufgerufenen Programms nach StdOut werden nicht unterdrückt und PE gibt eine eigene Kopfzeile aus.
`-e2`	Vor dem ersten Fehler innerhalb eines Includelevels wird der Includelevel selbst ausgegeben.
`-h`	Ausgabe einer Hilfe.

Wird keine Konfigurationsdatei mit `-cfg` angegeben, muss sich die Default-Konfigurationsdatei `pe.cfg` oder `PE.CFG` beim Aufruf von PE (s. Abschnitt 2) im Programmverzeichnis befinden (eine Ausnahme stellt die Anwendung von PE unter RTOS-UH dar: Hier wird die Datei `pe.cfg` oder `PE.CFG` entweder im aktuell gesetzten Arbetisverzeichnis oder in `/ED` gesucht).

Standardmäßig gibt PE lediglich die einzeiligen Fehlerbeschreibungen und die Schlusszeile des Compilers oder Assemblers nach StdOut aus. Die Ausgaben des von PE aufgerufenen Programms nach StdOut werden unterdrückt, die Ausgaben nach StdErr jedoch nicht. Mit dem Parameter `-e1` können auch die Ausgaben nach StdOut wieder eingeschaltet werden. Außerdem gibt PE dann seine eigene Kopfzeile aus. Mit `-e0` wird die Kopfzeile des erzeugten Listings mit ausgegeben.

Der Parameter `-lo=no` hat eine etwas andere Bedeutung als gewohnt. PE benötigt immer ein vollständiges Listing, um auch den Namen der Include-Datei, in der ein Fehler auftrat, korrekt ermitteln zu können. Somit wird, auch wenn `-lo=no` angegeben ist, immer eine Liste erzeugt (ohne weiteren `lo`-Parameter in der Datei Quelldateiname`.sr` relativ zum aktuellen Arbeitsverzeichnis oder der Umgebungsvariablen `PE_LIST`). Die Angabe von `-lo=no` bewirkt dann lediglich, dass die erzeugte Listdatei wieder gelöscht wird.

Auch unter Windows wird eine Pfadangabe einer Include-Datei der Form `/H0/...` als absoluter Pfad auf dem Laufwerk des aktuellen Arbeitsverzeichnisses (also z.B. `C:\H0\...`)interpretiert. Auf diese Weise können die selben Dateien mit absoluten Pfadangaben für Include-Dateien sowohl unter Windows als auch unter RTOS-UH und Linux verwendet werden.

Wird die Umlenkung der Ausgabe mit `-o` zusammen mit der Option `-e1` (Ausgaben des von PE aufgerufenen Programms nicht unterdrücken) verwendet, wird im Arbeitsverzeichnis temporär die Datei `pe_redir.tmp` angelegt. Vor dem Ende des PE-Laufs wird diese Datei wieder gelöscht.

Die Arbeitsweise des PE setzt sowohl die Liste als auch den Quellcode in einer Datei voraus. Aus diesem Grund werden `StdIn` bzw. `StdOut` als Pfadangaben für `-si`, `-lo` und `-co` nicht unterstützt.

- *Beispiel für einen PE-Aufruf:*

  Als aufzurufendes Programm ist `cvcp.exe` in `pe.cfg` eingetragen. Ein möglicher Aufruf des PE lautet dann

  ```
  pe -si=PIRegler.pq -lo=no pd_ppc.vbi -sz=10000 -lif -e0
  ```

  mit

`PIRegler.pq`	Quelltext
`PIList.lst`	Listdatei (Default: Quelldateiname`.lst`)
`PIRegler.sr`	S-Record (Default: Quelldateiname`.sr`)

  und den Parametern

`-lo=no`	Die Listdatei `PIList.lst` wird am Ende wieder gelöscht
`-lif`	Prüfung auf Notwendigkeit des Compilerlaufs (LIF-Datei)
`-e0`	Ausgabe der Kopfzeile des Listings
`pd_ppc.vbi`	PEARL90 Compiler
`-sz=10000`	Size-Parameter für den C-VCP

  Die Parameter `pd_ppc.vbi` und `-sz=10000` gehören `comp-args` an. Sie werden vom PE direkt an den VCP und den darauf laufenden Compiler weitergereicht.

4 Umgebungsvariablen für Ein- und Ausgabepfade

Die nachstehend beschriebenen Umgebungsvariablen können verwendet werden, um dem Compiler oder Assembler Pfade für Ein- und Ausgabedateien vorzugeben. Sind eine oder mehrere dieser Variablen nicht gesetzt, so wird das aktuelle Arbeitsverzeichnis verwendet.

Umgebungsvariable	**Bedeutung**
PE_SOURCE	Verzeichnis der Quelldateien
PE_LIST	Verzeichnis der Listdateien
PE_CODE	Verzeichnis der S-Records
PE_PROJECT	Name des aktuellen Projekts

Sollte PE_PROJECT gesetzt sein, so werden zusätzlich folgende Variablen überprüft:

Umgebungsvariable	**Bedeutung**
<pname>_SOURCE	Verzeichnis der Quelldateien des Projekts
<pname>_LIST	Verzeichnis der Listdateien des Projekts
<pname>_CODE	Verzeichnis der S-Records des Projekts

Dabei ist <pname> ein frei wählbarer Name des Projekts. Diese Variablen übersteuern, falls sie gesetzt sind und PE_PROJECT=<pname> gilt, die Werte der Variablen PE_SOURCE, PE_LIST und PE_CODE.

- *Beispiel:*

  Die Umgebungsvariablen sind wie folgt gesetzt:

```
PE_PROJECT   = HTTPD
PE_SOURCE    = C:\Rtos-UH\source
PE_CODE      = C:\Rtos-UH\srec
HTTPD_SOURCE = C:\Rtos-UH\HTTPDaem\source
SIM_SOURCE   = C:\Rtos-UH\SimuTool\source
```

  Diese Definition der Umgebungsvariablen hat zur Folge, dass die Suche der Quelltextdateien relativ zu C:\Rtos-UH\HTTPDaem\source erfolgt. PE_SOURCE bzw. SIM_SOURCE werden ignoriert, denn PE_PROJECT und HTTPD_SOURCE sind gesetzt bzw. PE_PROJECT=HTTPD. Da jedoch HTTPD_CODE nicht gesetzt ist, wird auf PE_CODE zurückgegriffen und S-Records relativ zu C:\Rtos-UH\srec geschrieben. Für die Listdateien ist keine Umgebungsvariable gesetzt, diese werden daher relativ zum aktuellen Arbeitsverzeichnis abgelegt.

5 Download, Versionsliste und Ansprechpartner

5.1 Download

Das Programm PE ist Freeware[1]. Die Binaries für Windows9x/2000/NT/ME/XP, Linux, HP-UX und RTOS-UH inklusive Dokumentation befinden sich im Verzeichnis

`http://www.irt.uni-hannover.de/pub/rtos-uh/pe/`

Eine HTML-Version dieser Dokumentation ist unter

`http://www.irt.uni-hannover.de/rtos/freeware/pedoc.html`

verfügbar.

5.2 Versionsliste

1.0-A	04.05.1999	Erste verfügbare Version.
1.0-B	14.05.1999	Option `-e` für die Zusatzmeldung, in welcher Datei sich die folgenden Fehler befinden. Ausgabe von Fehlermeldungen, die zum Abbruch des Compilers/Assemblers führen. Ausgabe des Size-Limit-Errors.
1.0-C	21.05.1999	Fehler bei `#IFDEF` und `#INCLUDE` behoben.
1.0-D	26.08.1999	Option `-q`: Keine Ausgaben auf `StdOut` ausser Fehlermeldungen des Compilers.
1.0-E	30.08.1999	Warnung bei `-L` und `SETLINE`.
1.0-F	14.01.2000	Anpassung an neuen C-VCP: `-SZ` Option. Keine Abfrage der Parameter mehr über die Konsole.
1.0-G	21.01.2000	Parameter des PE: `-si`, `-co`, `-lo`, `-q`, `-e`; andere Parameter werden direkt an das zu aktivierende Proramm weitergegeben; `-I`, `-l0` und `-l1` für den C-VCP werden mit ausgewertet. Fehlermeldungen des PE auf `StdErr` statt auf `StdOut`.
1.1-A	21.02.2000	Ausgabe auch der PEARL `WARNING` im angegebenen Fehlerformat. Optionen `-e0`, `-e1`, `-e2` hinzugefügt, `-q` statt dessen entfernt. Neuer Switch `-as` für Assemblerprogramme. Test auf `SdtIn` und `StdOut`, da nicht als Quell- oder Listing-Pfade erlaubt. Erweiterung der Hilfe zu `-lo` und `-co`. Datei-Erweiterungen von Code und Listing auch in Großbuchstaben. Test, ob überhaupt ein Programm zum Aufruf angegeben ist.
1.1-B	02.04.2000	Fehler unter RTOS-UH behoben. Verwendung von `SETLINE` ermöglicht. Größenangabe in der `SIZE_LIMIT_ERROR`-Meldung.
1.1-C	12.04.2000	Fehler beim Compilieren von Shell-Modulen behoben.
1.1-D	14.04.2000	Fehler unter RTOS-UH auf PowerPCs behoben.
1.1-E	19.04.2000	Fehler unter RTOS-UH auf PowerPCs behoben.
1.1-F	26.04.2000	Test auf `+N` oder `PAGE`. Hinweis auf `-L`, `+P` und `PRINT 0`, falls Include fehlschlug.
1.1-G	19.05.2000	Test auf `.fin` und `.else` beim Assembler.
1.1-H	27.06.2000	Fehler beim Include in Assemblerprogrammen behoben. Mehrzeilige Listings einer Assemblerzeile werden berücksichtigt. Quelldatei wird zuerst im aktuellen Arbeitsverzeichnis gesucht, dann in `PE_SOURCE`, falls gesetzt.
1.1-I	20.07.2000	PEARL Module Summary wird auf extra devices untersucht.
1.1-J	23.08.2000	Option `-o` zur Umleitung der Ausgabe des PE in eine Datei.

[1] Bitte beachten Sie jedoch unbedingt die Datei `readme.txt` im Download-Verzeichnis oder im heruntergeladenen Archiv.

1.1-K	29.08.2000	Switch `-as` abgeschafft, statt dessen Test der Kopfzeile auf Assembler. Erste für RTOS-UH verfügbare Version.
1.1-L	16.11.2000	Fehler beim Einsatz unter RTOS-UH und WinSTon behoben. Fehler behoben, der auftrat, wenn die erste Zeile eine Include-Datei wieder ein Include-Statement war.
1.2-A	01.03.2001	Option `-lif` zur Verwendung der Datei `<sourcefile>.lif` und Prüfung, ob der S-Record noch neuer als alle Quell- und Include-Dateien ist.
1.2-B	30.03.2001	Anpassung an das Format der Kopfzeile des Compilers P-16.1-A und -B.
1.3-A	06.07.2001	Anpassung an mingw32-gcc, Programme jetzt als MS-Windows Applikation (benutzten System-Dlls).
1.3-B	22.11.2001	Fehler bei Abbruch der Übersetzung z.B. durch nicht gefundene Include-Datei und Verwendung der Option `-lif` behoben.
1.3-C	30.11.2001	Fehler, wenn Include-Datei absolut und mit \ (Windows) angegeben wurde, behoben.
1.4-A	10.04.2002	Vorgabe von Pfad und Name der Konfigurationsdatei mit der Option `-cfg` (Default: `pe.cfg` oder `PE.CFG`); die Default-Datei wird jetzt unter allen Windows-Versionen im Verzeichnis von `pe.exe` und nicht im Arbeitsverzeichnis gesucht. Deutlichere Fehlermeldung, wenn aufgerufener Compiler nicht ausgeführt wird und PE das Programmlisting nicht öffnen kann. Option `-lif`: Neu-Übersetzung, auch wenn Source und S-Record gleich alt sind (für Dateisysteme mit grober zeitl. Auflösung); Eintrag der Quelldatei in der lif-Datei immer mit absoluter Pfadangabe.
1.5-A	05.08.2002	`/H0/...` wird auch unter Windows als absoluter Pfad interpretiert: So kann mit `C:\H0\...` ein RTOS-Dateisystem emuliert werden. Auch auf UNIX-Systemen wird `pe.cfg` zuerst im Programmverzeichnis gesucht (danach im Arbeitsverzichnis). Fehlermeldung statt Absturz, falls Inlcude-Datei nicht aus der Quelldatei bestimmt werden konnte. Berücksichtigung von Shellmodulen mit `USABLE_ALSO_INSIDE...`-Meldung. Robusteres Verhalten bei Quellzeilen, die nicht im Listing erscheinen (z.B. `#IF..#FIN`).
1.5-B	07.10.2002	Geschwindigkeitsverbesserung.
1.5-C	07.11.2002	Berücksichtigung des Listing-Formates des TAPP, das sich vom AS68 unterschiedet.
1.5-D	21.11.2002	Erhöhung der Robustheit bei fehlerhafter `pe.cfg`. PE_SETUP 1.3-A: `pe.cfg` wird, falls möglich, im Programmverzeichnis angelegt, sonst im Arbeitsverzeichnis; das benutzerdefinierte Fehlerformat wird nach der Eingabe überprüft.
1.6-A	14.01.2003	Reentrant unter RTOS-UH. Neues Verfahren zur Ermittlung der aktuellen Programmzeile. Erste verfügbare Version für HP-UX und Linux.
1.6-B	15.01.2003	Fehler beim Test auf `SETLINE` behoben.
1.6-C	05.05.2003	Anpassung an neuen `SETLINE` mit nachfolgender Modulnummer.
1.6-D	18.02.2004	Erkennung der PEARL90-Demo- Compiler.
1.6-E	28.04.2004	Fehlerzeile zeigte Dateien der Root-Ebene in der Form `c://test.pq` an.
1.6-F	11.11.2004	Schreiben der Abhängigkeiten in `<sourcefile>.dep` im Makefile-Format.
1.6-G	11.12.2006	Unterstützung des Linker-Outputs.

5.3 Ansprechpartner

Bei Problemen, Kritik und Anregungen wenden Sie sich bitte an

Torsten Lilge , Institut für Regelungstechnik
Appelstr. 11
D-30167 Hannover
E-Mail : lilge@irt.uni-hannover.de