

# Bedienungsanleitung für die statische Analyse physikalischer Einheiten in PEARL

© 2015

Christina K. Houben

Lehrgebiet Informationstechnik

FernUniversität in Hagen

## Inhaltsverzeichnis

<b>1</b>	<b>Bedienungsanleitung für den statischen Analysator</b>	<b>2</b>
<b>2</b>	<b>Syntax für die Angabe physikalischer Einheiten in PEARL-Dateien</b>	<b>2</b>
2.1	Ein- und Ausschalten der statischen Analyse . . . . .	2
2.2	Deklarationen, Spezifikationen und Typdefinitionen . . . . .	3
2.3	Literale . . . . .	4
2.4	Kontrollstrukturen . . . . .	5
2.5	Prozedur- und Operatorköpfe . . . . .	6
2.6	Felder . . . . .	6
<b>3</b>	<b>Syntax für die Definition physikalischer Einheiten in XML</b>	<b>7</b>
<b>4</b>	<b>Angaben zum statischen Analysator</b>	<b>7</b>
4.1	Bildnachweis . . . . .	7
4.2	Programmierungsumgebung . . . . .	7

# 1 Bedienungsanleitung für den statischen Analysator

## 2 Syntax für die Angabe physikalischer Einheiten in PEARL-Dateien

### 2.1 Ein- und Ausschalten der statischen Analyse

10. Um anzuzeigen, dass ein Modul Steuerkommentare zur Überprüfung physikalischer Einheiten enthält, kann optional `/*+U*/` hinter dem Semikolon der `MODULE`-Definition angegeben werden. Ansonsten arbeitet der Analysator wahlweise modul- oder prozedurorientiert:

```
1  MODULE(Wheatstone); /*+U*/
```

11. Die statische Analyse der physikalischen Einheiten wird mit `/*+U*/` eingeschalten und mit `/*-U*/` ausgeschalten:

```
1  /*-U*/
2  PUT 'Count of Successful Test Cases = ', CntSuccess
3  TO Terminal BY A, F(2), SKIP;
4  PUT 'Count of Erroneous Test Cases = ', CntError
5  TO Terminal BY A, F(2), SKIP;
6  /*+U*/
```

12. Die statische Analyse arbeitet nur mit physikalischen Einheiten von `FIXEDs` und `FLOATs`, sowohl atomar als auch in Form von Zeigern, Feldern und `STRUCTs`. Damit lässt sich auch die Korrektheit physikalischer Einheiten von komplexen Zahlen, Vektoren oder Matrizen etc. bestimmen. Zur Vereinfachung des Analysators werden physikalische Einheiten anderer Datentypen wie z.B. `CHAR` nicht betrachtet. Die Datentypen `DUR` bzw. `DURATION` und `CLOCK` werden nicht mit Einheiten versehen, weil sie es bereits sind und darüber hinaus allein für die Ablaufplanung von Prozessen genutzt werden sollen und nicht für Berechnungen in physikalischen Formeln.

13. Der Analysator unterscheidet drei verschiedene Arten von importierte Programmeinheiten, welche nicht notwendiger Weise physikalischen Einheiten verwenden.

Zum einen sind dies Ein- und Ausgabebefehle. Hier werden Standardperipherie auf der einen Seite und Sensoren und Aktuatoren auf der anderen Seite unterschieden. Bei Befehlen wie `PUT`, `GET` und `READ`, `WRITE` wird automatisch die Prüfung der physikalischen Einheiten abgeschalten. Hingegen ist es bei Sensoren und Aktuatoren sinnvoll, die physikalischen Einheiten bei den Befehlen `TAKE` und `SEND` zu prüfen. Stellen die entsprechenden Treiber keine Angaben zu physikalischen Einheiten zur Verfügung, muss die Analyse explizit ausgeschalten werden.

Zum anderen sind dies Standard-Operatoren und -Prozeduren wie `ABS` und `EXP`. Hier hat der Analysator intern vermerkt, welche physikalischen Einheiten für Argumente und Ergebnisse zugelassen sind.

Schließlich können globale Prozeduren für in Import spezifiziert werden. Hierbei sind physikalische Einheiten für die `SPC`-Anweisung so zu formulieren wie für Prozedurköpfe. Sollen die Einheiten hier nicht geprüft werden, muss die Analyse für benutzende Befehle explizit ausgeschalten werden.

## 2.2 Deklarationen, Spezifikationen und Typdefinitionen

20. Bei Deklarationen oder Spezifikationen mit einer oder mehreren Typangaben wird die physikalische Einheit jeweils direkt hinter jedem Typbezeichner angegeben:

```
1 DECLARE Resistances(1:4) FLOAT /**+U= Ohm */ ,
2      IdxUnknown          FIXED /**+U= 1 */ ,
3      Current              FLOAT /**+U= mA */ ;
```

21. Enthält ein Deklarationsblock von DCL bzw. DECLARE bis zum abschließenden Semikolon nur eine Variable oder mehrere Variablen gleichen Typs, so kann die physikalische Einheit auch nach dem Semikolon angegeben werden:

```
1 DECLARE Epsilon INV FLOAT INIT(0.05); /**+U=mA*/
```

22. Werden mehrere Variablen gleichen Typs mit unterschiedlichen physikalischen Einheiten zu einer Deklaration mit nur einem einzigen Typbezeichner zusammengefasst, so können die physikalischen Einheiten nach dem Semikolon getrennt durch Kommata in der Reihenfolge ihrer Nennung aufgelistet werden:

```
1 DCL (Inductance, Length, Resistance)
2     INV FLOAT INIT(2.5, 1.0, 5.0); /**+U=uH, cm, Ohm*/
```

23. Die Angabe der physikalischen Einheit ist obligatorisch. Zusätzlich darf auch der Dimensionsname angegeben werden. Dies fördert die Lesbarkeit, weil so komplex aufgebaute Einheiten in einem Wort zusammengefasst werden. Der Dimensionsname steht dann vor der Einheit und die Einheit wird in Klammern eingefasst:

```
1 DCL gMoon INV FLOAT INIT(1.62); /**+U=Acceleration(m/s2)*/
```

24. Eine physikalische Einheit besteht aus den optionalen Teilen Präfix, Einheitensymbol und Exponent. Präfixe werden ohne ein weiteres Zwischenzeichen direkt vor ein Einheitensymbol geschrieben. Exponenten werden ohne ein weiteres Zwischenzeichen direkt hinter ein Einheitensymbol geschrieben. Jede zusammengesetzte Einheit darf maximal einen Bruchstrich enthalten. Der Bruchstrich muss genutzt werden, damit keine negativen Exponenten entstehen. Der Einfachheit halber sind auch keine nicht-ganzen Exponenten zugelassen. Mehrere Basiseinheiten einer zusammengesetzten Einheit werden mit \* getrennt. Alle Basiseinheiten unterhalb eines Bruchstriches werden ebenso durch \* voneinander getrennt:

```
1 DCL RotatingTable FLOAT; /**+U=AngularMomentum(kg*m2/s)*/
```

25. Bei Typdefinitionen von Prozedurzeigern wird die physikalische Einheit direkt hinter die Typbezeichnungen FIXED oder FLOAT gesetzt:

```
1 TYPE TFunction REF PROC(FLOAT /**+U=dimA(unitA)*/)
2      RETURNS(FLOAT /**+U=dimB(unitB)*/);
```

26. Bei Typdefinitionen oder Deklarationen von Strukturen werden physikalische Einheiten von FIXED- oder FLOAT-Komponenten direkt hinter die jeweiligen Typ-Schlüsselwörter gesetzt:

```
1 DCL Patient STRUCT (/
2   Name          CHAR(32),
3   BloodCount    STRUCT (/
4     Leucocytes  FLOAT /**+U= 1/nL */ ,
5     Erythrocytes FLOAT /**+U= 1/pL */ /),
6   LiverValues   STRUCT (/
7     Albumin     FLOAT /**+U= g/dL */ ,
8     Biliburin   FLOAT /**+U= mg/dL */ /)
9 /);
```

27. Es gibt sowohl konkrete als auch generische Einheiten. Generische Dimensionen haben die Bezeichner `dimA` bis `dimZ`, während generische Einheiten die Dimensionen `unitA` bis `unitZ` haben. Die Gültigkeit generischer Dimensionen und Einheiten gilt entsprechend der Definitionsbereiche modul-global oder nur prozedur-weit:

```
1 ArithmeticMean: PROC(Numbers() FLOAT IDENT /*+U=unitA*/)
2                 RETURNS(FLOAT /*+U=unitA*/);
```

28. Momentan wird nicht die Möglichkeit geboten, generische Exponenten anzugeben. Ein Beispiel, wo man dies benötigen würde, ist die `SQRT`-Funktion, die den Exponenten `2*expA` auf `expA` abbildet. Neben einer komplizierteren Syntax führt dies im nächsten Schritt zu der Möglichkeit beliebig komplizierte Formeln zur Angabe des Ergebnis-Exponenten anzugeben. Diese Formeln selbst könnten teilweise nur durch Prozedur-artige Berechnungen ermittelt werden. So resultiert die Potenzierung mit einem `FIXED` zu `n*expA` als Ergebnis-Exponent, während die Potenzierung mit einem `FLOAT` sogar zu gebrochenen Exponenten führen kann. Ersatzweise dürfen daher für eine solche Prozedur Beispiele für physikalische Einheiten angegeben werden. Dafür wird vor der Prozedur `/*+U=Examples*/` angegeben. Physikalische Einheiten für verschiedene Beispiele werden durch Kommata getrennt angegeben:

```
1 /*+U=Examples*/
2 CumulativeProduct: PROC(F() FLOAT IDENT
3 /*+U=Temperature(degreeC),Length(m)*/)
4 RETURNS(FLOAT
5 /*+U=Temperature(degreeC5),Length(m5)*/);
```

## 2.3 Literale

30. Die physikalische Einheit von Literalen in Initialisierungen wird nicht explizit angegeben, da dies schon nach der Typangabe geschieht:

```
1 DECLARE gEarth INV FLOAT INIT(9.81); /*+U= m/s2 */
```

31. Treten Literale in Formeln auf, so werden ihre Einheiten nach dem zugehörigen Semikolon in der Reihenfolge ihres Auftretens mit physikalischen Einheiten versehen. Dabei werden die Werte zur Steigerung der Lesbarkeit wiederholt. Bei dimensionslose Literalen wird die Einheit 1 nicht explizit aufgeführt, sondern nur der Wert des Literals:

```
1 CoulombLaw := (Q1 * Q2) / (4.0 * Pi * 8.85 * 1.0003 * r * r);
2 /*+U= 4.0, 8.85 A*s/V*m, 1.0003 */
```

32. Werden Variablen oder Konstanten gleichzeitig deklariert und mit einer Formel initialisiert, so werden nacheinander die physikalische Einheit der deklarierten Variable und dann alle Einheiten der Literale durch Kommata getrennt nach dem Semikolon aufgelistet:

```
1 DCL Pi INV FLOAT INIT(3.141592654); /*+U=1*/
2 DCL VacuumPermeability INV FLOAT INIT(Pi * 4.0E+7);
3 /*+U=H/m,4.0E+7H/m*/
```

33. Konvertierungen von einer Einheit in eine andere werden über gewöhnliche Formeln realisiert, wobei der Konvertierungsfaktor ein Literal ist. Daher werden auch hier nach dem abschließenden Semikolon alle Werte und physikalischen Einheiten der verwendeten Literale in der Reihenfolge ihres Auftretens durch Kommata getrennt aufgelistet:

```
1 Length := Length_cm / 100.0; /*+U=100.0 cm/m*/
```

34. Sollen zwei dimensionslose Einheiten ineinander konvertiert werden, so muss für den Konvertierungsfaktor die entsprechende physikalische Einheit als Bruch zweier ganzer Zahlen angegeben werden:

```

1 DCL (EggCount, DozenCount) FIXED; /*+U= 1,12 */
2 DozenCount = 5; /*+U= 12 */
3 EggCount := DozenCount / 12; /*+U= 12/1 */

```

## 2.4 Kontrollstrukturen

40. Physikalische Einheiten von Literalen in IF-Bedingungen werden direkt vor oder direkt nach dem THEN in der Reihenfolge ihres Auftretens durch Kommata getrennt angegeben:

```

1 IF ABS(Result) - 1.0 < 0.05 /*+U=1.0mA,0.05Ohm*/
2 THEN
3   CntSuccess := CntSuccess + 1; /*+U=1*/
4 ELSE
5   CntError := CntError + 1; /*+U=1*/
6 FIN;

```

41. Hingegen muss bei CASE-Verzweigungen nach der Variable, in deren Abhängigkeit die Fallunterscheidungen getroffen werden, die gemeinsame physikalische Einheit aller Literale, die in den Alternativbedingungen auftauchen, angegeben werden:

```

1 CASE IdxUnknownResistance /*+U=1*/
2   ALT (1) RETURN(R(2) * R(3) / R(4));
3   ALT (2) RETURN(R(1) * R(4) / R(3));
4   ALT (3) RETURN(R(1) * R(4) / R(2));
5   ALT (4) RETURN(R(2) * R(3) / R(1));
6 FIN;

```

42. Physikalische Einheiten von Literalen in WHILE-Schleifen müssen direkt vor oder direkt nach dem Schlüsselwort REPEAT angegeben werden, und zwar in der Reihenfolge der Nennung der Literale durch Kommata getrennt:

```

1 WHILE (Distance > 0.1) AND (Duration < 20.0)
2 REPEAT /*+U=0.1mm,20.0s*/
3   ...
4 END;

```

43. Physikalische Einheiten für Zählvariablen und Literale in FOR-Schleifen werden nicht angegeben, denn hier nimmt der Analysator automatisch die dimensionslose Einheit 1 an:

```

1 FOR testcase FROM 1 TO 4 REPEAT
2   FOR idxunknown FROM 1 TO 4 REPEAT
3     ... TestCases(testcase).Resistances(idxunknown, 0.02) ...
4   END;
5 END;

```

44. Es kann auch passieren, dass sich die physikalische Einheit einer Variable während der Programmausführung bzw. während mehrerer Schleifendurchläufe ändert. Dann wird bei ihrer Deklaration die Zieleinheit angegeben und gleichzeitig mit -U die Überprüfung der physikalischen Einheiten nur für diese Variable unterdrückt. An dem Punkt im Programmablauf, an dem die endgültige Einheit erreicht ist, wird die Überprüfung wieder mit +U für diese Variable aktiviert:

```

1 DCL Width FLOAT; /*+U=dim1(unit1)*/
2 DCL Sum FLOAT; /*-U=unit1*unit2*/
3 Width := (B-A) / N;
4 Sum := Integrand(A) + Integrand(B);
5 FOR i FROM 2 TO N REPEAT
6   Sum := Sum + 2*Integrand(A + i*Width);
7 END;
8 Sum := Sum * (B-A) / (2*N); /*+U=unit1*unit2*/

```

## 2.5 Prozedur- und Operatorköpfe

50. Die physikalischen Einheiten von Argumenten in Prozedur- und Operatorköpfen werden direkt hinter jeder Typbezeichnung angegeben:

```
1 CalcUnknownResistance: PROCEDURE(  
2   Resistances(1:4)  FLOAT IDENT /*+U= Ohm */ ,  
3   IdxUnknown        FIXED      /*+U= 1   */ ,  
4   Current           FLOAT      /*+U= mA   */ )  
5   RETURNS(          FLOAT      /*+U= Ohm */ );
```

51. Werden Prozedurzeiger übergeben, so werden die physikalischen Einheiten der Argumente und des Ergebnisses des Prozedurzeigers in der Reihenfolge ihrer Nennung durch Kommata getrennt angegeben, also zuerst das erste Argument, ... und zuletzt das letzte Argument und schließlich die physikalische Einheit des Ergebnisses:

```
1 TYPE Function REF PROC(FLOAT /*+U=dimA(unitA)*/)   
2   RETURNS(FLOAT /*+U=dimB(unitB)*/);  
3  
4 TrapezoidalRule: PROC(  
5   Integrand Function /*+U=dim1(unit1),dim2(unit2)*/ ,  
6   (A, B)          FLOAT /*+U=dim1(unit1) */ ,  
7   N              FIXED  /*+U=1 */ ,  
8   RETURNS(       FLOAT  /*+U=unit1*unit2 */ );
```

52. Bei Strukturen wird ebenso wie bei Prozedurzeigern verfahren. Für alle Komponenten, die mit physikalischen Einheiten attribuiert sind, werden in der Reihenfolge ihrer Definition alle physikalischen Einheiten aufgelistet.

## 2.6 Felder

60. Die physikalische Einheit von Feld-Indizes braucht nicht angegeben zu werden, denn bei Ihnen wird automatisch angenommen und überprüft, dass sie dimensionslos sind:

```
1 DECLARE Resistances(1:4)  FLOAT; /*+U=Ohm*/  
2 Resistances(1) := 1000.0;      /*+U=Ohm*/
```

61. Physikalische Einheiten von Feldelementen kann man in sechs Kategorien unterteilen. Zum einen kann man konkrete Einheiten oder generische Einheiten angeben. Zum anderen können die physikalischen Einheiten der Feldelemente alle gleich sein, oder alle unterschiedliche, jedoch formalisierbar, oder alle unterschiedlich und nicht formalisierbar.

Der Analysator arbeitet nur mit Felder, bei denen alle Elemente die gleiche physikalische Einheit haben, egal ob konkret oder generisch.

Bei unterschiedlichen, jedoch formalisierbaren physikalischen Einheiten sollte die Überprüfung der Einheiten entsprechend dem zuvor genannten Beispiel mit der Variable, deren physikalische Einheit sich im Laufe des Programmdurchlaufs ändert, mit -U abgestellt und erst am Ende wieder mit +U angestellt werden.

Bei unterschiedlichen und nicht formalisierbaren physikalischen Einheiten sollte schon allein aus semantischen Gründen kein Feld gewählt werden, sondern mehrere unterschiedliche Variablen.

### **3 Syntax für die Definition physikalischer Einheiten in XML**

## **4 Angaben zum statischen Analysator**

### **4.1 Bildnachweis**

- 01. Recycling-Button: `klepas.deviantart.com` von A. Jay
- 02. Maßband-Button: `www.adpic.de` von M. Dietrich
- 03. Fragezeichen-Button: `paulabrown.net` von P. Brown

### **4.2 Programmierumgebung**

- 04. Programmiersprache: Java SE 1.7
- 05. Entwicklungsumgebung: eclipse Juno 4.2.1
- 06. Betriebssystem: Windows 8.1 Pro