

Ausführliche Liste der Befehle und Funktionen von PAKMA Version 2

Stand: Juli 00

Im Editor des PAKMA 2.x eingegebene Texte werden in einem Parser so umgesetzt, daß sie dann von Turbopascal für Windows kompiliert werden können. Damit der Editor auch ohne PASCAL-Kenntnisse benutzt werden kann, ergänzt er manche Formelemente selbständig (z.B. wird bei Zuweisungen das '=' durch ':=' ersetzt, oder die Zeilen mit einem ';' abgeschlossen). Daher müssen bei PAKMA auch die Variablen nicht eigens deklariert werden, sondern werden, wenn sie benutzt werden, automatisch deklariert und mit '0' initialisiert. Zwischen Groß- und Kleinschrift wird nicht unterschieden.

Variablen

<code>var</code>	Realvariable. Dies sind die üblichen Variablen in PAKMA. Die Rechengenauigkeit beträgt 11 bis 12 Stellen
<code>var%</code>	Integervariable. Diese sind 16 Bit Ganzzahlen im Wertebereich von -32768 bis 32767.
<code>var\$</code>	Stringvariable. Diese Variablen dienen zur Verwaltung von Zeichenketten, können aber nicht mit ausgabe () übergeben werden.
<code>var[nr]</code>	Arrays werden definiert durch: <code>dim var[feldgr.]</code> Dabei werden <code>feldgr.+1</code> Felder angelegt (<code>var[0]</code> bis <code>var[feldgr.]</code>).

Alle Variablen, die im PAKMA-Kernprogramm benutzt werden, werden vor Programmstart mit 0 initialisiert, außer, sie erhalten durch Startwerte einen anderen Wert.

Deklaration von Variablen

Wichtig :

Wenn Variablen nicht durch eine Anweisung in der folgenden Art deklariert werden, werden sie automatisch als Real-Variablen bzw. - wenn sie auf '%' oder '\$' enden - als Integer- oder String- Variable deklariert.

<code>var i,j : Integer;</code>	Deklariert i und j als Integervariable
<code>var r,x : Real;</code>	Deklariert r und x als Realvariable
<code>var s,t : String;</code>	Deklariert s und t als Stringvariable
<code>dim a[feldgr.];</code>	Definiert a als Array von Realvariablen

Variablen, die durch eine derartige Anweisung deklariert worden sind, dürfen zur Verdeutlichung mit einem '?' am Ende versehen werden.

Es existieren noch weitere Variablentypen, hierfür lesen Sie bitte ein PASCAL-Handbuch.

Befehle für Schleifen und bedingte Ausführung von Anweisungen

Anweisungen in den folgenden Schleifen werden vom PAKMA-Editor automatisch eingerückt.

begin [Anweisungen] end;	Anweisungen und Funktionen lassen sich mit begin und end zu Blöcken verbinden, so ist es möglich, mehrere Anweisungen in einer Schleife, bzw. Abfrage unterzubringen.
----------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```

for var%:=startw.
to (bzw. downto)
endw. do
    [Anweisung]

```

Mit der **for**-Schleife läßt sich die Anweisung mehrfach hintereinander durchführen, wobei die Integervariable **var%** die Werte von **startw.** bis **endw.** in aufsteigender (bzw absteigender (**downto**)) Reihenfolge annimmt. Sollen mehrere Anweisungen in der Schleife durchgeführt werden, müssen diese mit **begin** und **end;** geklammert werden.

Wird die Laufvariable nicht als Integervariable eingeführt, so ergänzt der Editor das '%' automatisch beim Verlassen der Zeile.

Wird eine for -Schleife in einer proceduere oder function verwendet, muß die Laufvariable mit var deklariert werden.

Repeat-Schleife. Diese Schleife wird solange durchgeführt, bis die Bedingung hinter **until wahr** ist.

```

repeat
    [Anweisung1]
    [Anweisung2]
    .
    .
    .
until Bedingung;

```

```

if Bedingung then
    [Anweisung1]
[else]
    [Anweisung2]
case var% of
    wert_1:
        [Anweisung 1];
    wert_2,wert_3:
        [Anweisung 2]
    wert_4..wert_5:
        [Anweisung 4]
    wert_m:
        [Anweisung n]
[else]
    [Anweisung n+1]
end;

```

If-Abfrage, ist die Bedingung *wahr*, wird die Anweisung1 ausgeführt, sonst wird, wenn vorhanden die Anweisung2 ausgeführt.

Sollen mehrere Anweisungen in den einzelnen Fällen durchgeführt werden, müssen diese mit **begin** und **end;** geklammert werden.

Nimmt die Integervariable **var%** einen der Werte **wert_1 ... wert_m** an, so wird die dahinterstehende Anweisung ausgeführt; sonst wird, wenn vorhanden, die Anweisung **n+1** ausgeführt. Sollen mehrere Anweisungen in der Schleife durchgeführt werden, müssen diese mit **begin** und **end** geklammert werden. **var** darf z.B. auch die Zeichenvariable **taste** sein, jedoch keine String- oder Realvariable.

Rechenoperationen

```

+, -, *, /
:=
( )

```

Grundrechenarten

Zuweisung (z.B. **x:=x+1**)

Klammern in Rechenausdrücken

```

sin(var)
cos(var)
arctan(var)

```

Sinus (var : Winkel im Bogenmaß)

Kosinus (var : Winkel im Bogenmaß)

Arkustangens (Ergebnis im Bogenmaß)

pi

ist mit maximaler Genauigkeit bereits vordefiniert

ln (var)

Logarithmus (var > 0)

sqr (var)

Quadrat

Weitere Potenzen durch Multiplikation (z.B. $v^3=v*v*v$) oder $v^n=\exp(n*\log(v))$.

sqrt (var)

Quadratwurzel

exp (var)

Exponentialfunktion

round (var)

Rundet var auf nächsten Integerwert

trunc (var)

Schneidet bei var die Nachkommastellen ab

abs (var)

Liefert den Betrag von var

random

Erzeugt Zufallszahlen im Bereich von 0 bis 1 (0 inklusive, 1 exklusiv)

random(n)

Erzeugt ganze Zufallszahlen im Bereich 0 bis n-1.

Logische Operationen

Um komplexere Bedingungen formulieren zu können, können einzelne Bedingungen mit folgenden logischen Operationen zu einer Bedingung verknüpft werden.

<code>((Bedingung1)and(Bedingung2))</code>	Bedingung erfüllt wenn beide Bedingungen erfüllt sind
<code>((Bedingung1)or(Bedingung2))</code>	Bedingung erfüllt wenn mind. eine der Bedingungen erfüllt ist

Sonstige Funktionen

<code>n\$:=taste</code>	Tastenabfrage bei laufendem Programm. Die Variable <code>taste</code> wird nur beim Aufruf des Befehls <code>ausgabe</code> neu gesetzt. Ggf. gedrückte Umschalttasten haben keinen Einfluß auf den Wert von <code>taste</code> . Mit <code>taste</code> können direkt nur alphanumerische Zeichen abgefragt werden. Um die übrigen Zeichen abzufragen, müssen die ASCII-Codes verwendet werden.
<code>// Kommentar</code> <code>{ Kommentar }</code>	Um ein Kernprogramm besser zu erklären, können Kommentare, d.h. Text der keinen Einfluß auf den Programmablauf hat, verwendet werden. Jeder Text, der nach <code>'/'</code> bis zum Zeilenende oder zwischen <code>'{'</code> und <code>'}'</code> steht, wird als Kommentar angesehen.

Kern-`Programm´ als Procedure kern

Wenn mit selbstdefinierten Prozeduren, Funktionen und externen Prozeduren gearbeitet werden soll, ist es nötig, daß das Kernprogramm die Form einer Prozedur kern erhält, siehe das Beispiel unter Prozeduren und Funktionen. Dann erfolgt auch keine automatische Initialisierung aller Variablen auf Null. Diese muß genauso wie die Zuordnung der Startwerte zu den Variablen durch die Aufrufe `nullwerte` und `startwerte` erfolgen. (s. Beispiel unten)

Prozeduren und Funktionen

<code>uses unitname;</code> <code>bzw.</code> <code>use unitname;</code>	Einbinden einer Pascal Unit. Dieser Befehl muß, wenn Units benutzt werden sollen, am Anfang des Programms vor <code>procedure kern</code> , stehen. Es ist dann auch notwendig, daß das Hauptprogramm die Form einer Prozedur mit Namen <code>kern</code> hat, s.o.
<code>procedure dings</code> <code>procedure dings (a,b,c)</code> <code>procedure dings (a,b,c :real)</code> <code>function dings[(a,b,c)]</code> <code>function dings[(a,b,c:real)]:real</code>	Wie in Pascal ist es im PAKMA auch möglich Unterroutinen als (<code>procedure</code> und <code>funktion</code>) zu schreiben, die auch Übergabe von Parametern (z.B.: <code>a,b,c :Real</code>) erlauben. Funktionen geben auch einen Wert zurück; wenn nicht anderes angegeben ist, ist der Rückgabewert eine Realvariable. Sollen Unterroutinen benutzt werden, muß das Hauptprogramm die Form einer Prozedur haben mit dem Namen <code>kern</code> .

Beispiel:

<code>procedure kern</code>	Anfang des PAKMA-Hauptprogramms
<code>begin</code>	
<code>Nullwerte;</code>	In diesem Fall notwendig, siehe oben.
<code>Startwerte;</code>	In diesem Fall notwendig, siehe oben.
<code>[Anweisungen]</code>	Beliebige Anweisungen
<code>b:=Doppelt(a);</code>	Aufruf der Funktion <code>Doppelt</code>
<code>[Anweisungen]</code>	Weitere Anweisungen
<code>do_it;</code>	Aufruf der Prozedur <code>do_it</code> .
<code>end;</code>	Ende des Hauptprogramms

```

funktion Doppelt(x: Real):Real  Deklaration der Funktion Doppelt
begin
    return 2*x;                  Übergabe des Rückgabewertes
end;                             Ende der Funktion Doppelt

procedure do_it                  Deklaration der Prozedur do_it
begin
    [Anweisungen]
end;

```

Für genauere Informationen und weitergehende Möglichkeiten bei der Verwendung von Units, Prozeduren und Funktionen lesen Sie bitte ein Pascal- Handbuch.

Abweichend vom Pascal-Standard gilt:

Unterprogramme, die im Hauptprogramm verwendet werden, dürfen auch hinter dem Hauptprogramm stehen. Jedoch dürfen Unterprogramme, die vor dem Hauptprogramm stehen, nicht in Unterprogrammen hinter dem Hauptprogramm verwendet werden.

Variablen-, Konstanten-, oder Typendeklarationen dürfen nicht länger als eine Zeile mit 128 Zeichen sein. Mehrere jeweils durch VAR,CONST oder TYPE eingeleitete Zeilen sind jedoch zulässig.

PAKMA-Spezifische Befehle

Anweisungen zur Initialisierung bzw. Rücksetzen

Nullwerte	Setzt alle Variablen auf Null
Startwerte	Setzt alle Variablen für die Startwerte existieren auf diese Startwerte.

Messtypen

zählen	Zählen von Impulsen in dt
mauszählen	Zählen von PC-Maus Impulsen in dt Bei Verwendung von mauszählen wird automatisch 2-kanaliges Messen und vor_rück aktiviert.
u_meter	Spannung messen
bit_aus	Bitmuster ausgeben
bit_ein	Bitmuster abfragen

Einstellungen für alle Messtypen

vorb(abl)	Ablegen der Messwerte eines Durchgangs in der internen Ablage abl Es existieren 16 solche Ablagen. Werden allerdings Parallelmessungen in n-Kanälen durchgeführt, belegt ein Messdurchgang n Spalten, entsprechend muß dann $abl \geq 16/n$ sein. Wird diese Zahl überschritten, wird ein Fehler gemeldet.
s_folg('mode',dt)	Messmodus und Zeitschritt festlegen mode = i intermittierend (gleiches Intervall dt zwischen 2 Messungen) Die Messwerte werden sofort im Kernprogramm bearbeitet und gegebenenfalls ausgegeben(Echtzeit). Trifft das Kernprogramm vor Ablauf des Zeitintervalls dt auf einen Messaufruf, so wird noch solange gewartet, bis seit der letzten Messung dt s vergangen sind. mode = f fortlaufend beim ersten Messaufruf werden alle Messungen durchgeführt und anschließend verarbeitet und ausgegeben. Der Vorteil gegenüber dem intermittierenden Messen ist, daß in kleineren Zeitschritten dt gemessen werden kann. Die Minimalwerte für dt hängen vom jeweiligen Rechner ab. mode = e Einzelmessung.

Bei jedem Messaufruf wird eine Messung durchgeführt.

f_länge(n) Nur wenn mehr als 1024 Messungen in einem Durchgang aufgezeichnet werden sollen, ist diese Anweisung erforderlich.
Anzahl der dann zu speichernden Messwerte ist ($n \cdot 1024$) mit $n=1,2,3,4$

Einstellungen für zählen

vor_rück Messen in beide Richtungen (ohne diesen Aufruf : nur in eine Richtung)

sonar(i) Statt Zählrad Sonarmessung auf i Kanälen durchführen ($i=1$ oder 2). Die übrigen Kanäle bleiben Zählkanäle.

Einstellungen für mauszählen

mausunsichtbar Schaltet den Mauszeiger während der Messung aus.

mausort(x,y) ???

Einstellungen für u_meter

m_bereich(u_max, kanal) Stellt den Messverstärker für den Kanal **kanal** auf die angegebenen Messbereich **u_max** ein. Vorangestellt ist **u_max = 10 Volt** für alle Kanäle. Mit **kanal=0** werden alle Kanäle auf den angegebenen Wert eingestellt.
Folgende Messbereiche sind möglich :
u_max = 10V; 1V; 0.1V; 0.01V.
Bei Anschluss der C64-u_meter-Karte:
u_max = 8V; 2V; 0.5V; 0.125V

mes_warte(mms) Erhöht die Wartezeit zum Einschwingen des Verstärkers (größeres **mms** => größere Zeit). Nur nötig, wenn **u_max > 1V**. Die Werte für **t** werden in Microsekunden angegeben. Üblich sind Werte zwischen 10 und 1000 für 486'er bzw. Pentium Rechner

n_auf1 Stellt den 8-Bit Modus ein. Gilt nur für die Spannungsmessung mit alten C64-Modulen.
Hier gelten auch für **m_bereich** andere Grenzen und zwar :

Befehle zum Messen

mes Durchführen einer einkanaligen Messung mit dem aktuell aufgerufenen Messtyp
Ausnahme : Bei **mauszählen** wird eine 2-Kanal Messung durchgeführt

mes_p(n) Durchführen einer Parallelmessung auf n Kanälen

mes_r(n) Durchführen einer einkanaligen Messung im Kanal n. Dies sollte nur im s_folg-Modus 'e' (Einzelmessung) verwendet werden. Der Befehl kann sinnvoll sein, um für verschiedene Zählkanäle unterschiedlich lange Messzeiten vorzugeben.

Übergabe der Messwerte an Variablen

v:= mes_w Übergabe des Messwerte der letzten Messung an die Variable v

v:= mes_wp(n) Übergabe des Messwerte des n-ten Kanals der letzten Parallelmessung an die Variable v

v:= sp_w(r,i) Übergabe des Messwertes der r-ten Messung des i-ten Kanals der letzten Parallelmessung an die Variable v.
Wurde **vorb(k)** benutzt muß man statt **i**, **$n \cdot k + i$** benutzen, wobei **n** die Anzahl der Kanäle ist.

Einlesen von Eingabewerten während des Programmlaufes

y:=schieber(x) Während des Programmlaufs können über das Eingabeelement: „Schieber x“ Werte eingelesen und einer Variablen y zugeordnet werden.
Hinweis: Vor dem Programstart muß das Ausgabefenster aktiviert

sein, in dem der zu verändernde Schieber liegt. Zum Einlesen muß über Eingaben/Schieber ein Schieberelement im Ausgabefenster erzeugt werden.

y:=schalter(x) Entsprechend schieber(x); das Schalterelement kann hier jedoch nur zwei Werte annehmen.

y:=taster(x) Entsprechend schalter(x); das Schalterelement wechselt hier zwischen zwei Werten.

Bereitstellen von Variablen für die Ausgabeelemente

ausgabe(var1,var2,...) Bereitstellen der Variablen var1,var2,.. für Ausgabeelemente. Es können bis zu 32 Variablen übergeben werden.

Warte - Routine

warte_sek(t) Der Rechner wartet an dieser Anweisung t Sekunden in Vielfachen von 0,01s, unabhängig von der Taktfrequenz des Rechners, bevor er zur nächsten Anweisung weitergeht.

init_zykluszeit Initialisiert die Verwendung von zykluszeit.

zykluszeit(t) Der Rechner wartet an dieser Anweisung t Sekunden seit dem letzten Aufruf von zykluszeit oder init_zykluszeit.

Befehle zum Steuern über die Interfacebox

relais(a) Schaltet das Relais (a=1 : ein und a=0 :aus)
Grundeinstellung ohne Aufruf : ausgeschaltetes Relais.
Bei Programmabbruch oder -ende wird das Relais ausgeschaltet

ausg(bits) Gibt den Wert von bits als Bitkombination aus. Voraussetzung hierfür ist, daß vorher der Messtyp **bit_aus** gewählt wurde.

Funktionen zum Steuern und Abfragen des Programm-Modus

original Unabhängig vom gewählten Modus des Programmablaufs werden die folgenden Anweisungen als Originalmessungen ausgeführt.

repro Unabhängig vom gewählten Modus des Programmablaufs werden die folgenden Anweisungen als Reproduktion ausgeführt.

m=reprowert Gibt den Modus an, in dem das Kernprogramm zuletzt kompiliert wurde :
m = 0 : Original
m = 1 : Modell
m = 2 : Reproduktion
m = 3 : wird bei Original und Reproduktion ausgegeben, falls im Rechner keine PAKMA-Interfacekarte eingebaut ist.

Zeitverzögerung

warte_sek(t) Wartet für t Sekunden (t ≥ 0.01)

Graphensteuerung

grafен_löschen Löscht alle Graphen in allen Graphendarstellungen

neu_graf Unterbricht die Verbindungslinien aller Graphen für einen Ausgabebefehl

Neue externe Funktionen und Prozeduren in der Unit Pak_math

y:=sign(x) Liefert in y das Vorzeichen von x, d.h. -1 wenn x negativ, 0 wenn x Null und 1, wenn x positiv ist

mittelung(x,t,n,x_m,t_m) Mittelt über jeweils n Werte der Real-Variablen x und gibt den Zeitpunkt t_m aus, x_m ist der Mittelwert

den Zeitpunkt t_m aus. x_m ist der Mittelwert

lin_naehung (y,x,x_min,x_max)	Bildet die lineare Näherung von y zu x im Intervall $[x_{\min}, x_{\max}]$ mit der Methode des kleinsten Quadrates
lin_faktoren (y,a,b)	Die Real-Variablen a und b werden auf die Koeffizienten der linearen Näherung (s.o.) gesetzt
y_n:=lin_kurve (y,x)	Setzt y_n auf die Werte der linearen Näherungskurve
quad_naehung (y,x,x_min,x_max)	Bildet die quadratische Näherung von y zu x im Intervall $[x_{\min}, x_{\max}]$ mit der Methode des kleinsten Quadrates
quad_faktoren (y,a,b,c)	Die Real-Variablen a, b und c werden auf die Koeffizienten der quadratischen Näherung (s.o.) gesetzt
y_n:=quad_kurve (y,x)	Setzt y_n auf die Werte der quadratischen Näherungskurve
diffq_op (x,t,v,t_v);	Bildet den Differenzquotienten $v=dx/dt$ zum Zeitpunkt t_v .
y:=integral (x,t);	Berechnet das Integral von x über t
x_alt:= alt_wert (x);	Gibt den vorherigen Wert von x
x:= sum (dx);	Erhöht x um dx
dx:= diff (x);	Weist dx den Wert der Änderung der Variablen x zu