pCOM

objektbasierter Compiler

promicon SYSTEMS

pCOM ⟺ pASM

Gemeinsamkeiten

- Die erstellten Programmdateien (*.po) enthalten die gleichen ausführbaren Anweisungen
- Die gleichen Datentypen
- Quell-Code kann auf mehrere Dateien (*.af) verteilt werden
- Erzeugter Code kann in älteren Systemen verwendet werden

Unterschiede	**pCOM**	**pASM**
Quell-Code	ähnlich der Hochsprache C	Anweisungsliste
Dateiendung	*.pmc	*.as
Groß-/Kleinschreibung	relevant	nicht relevant
Temporäre Variablen	lokale Variablen	Rechen-Register

Sprachelemente

Bezeichner

erlaubte Zeichen: _ / a ... z / A ... Z / 0 ... 9

Beispiel: _Abakus_2

Konstanten

ganzzahlige Werte und Gleitpunktwerte

Beispiel:

```
-37.82          Gleitpunktzahl

1234            Ganzzahl Dez-Darstellung
0x5Ab4          Ganzzahl Hex-Darstellung
0B10010101      Ganzzahl Dual-Darstellung
‘S‘             Ganzzahl ASCII-Darstellung
```

Monadische Operatoren

Ein monadischer Operator bezieht sich nur auf einen Operanden.

-	Arithmetische Negation
!	Logische Negation
~	Komplement

Beispiel:

```
val = !(!4);   // val hat den Wert 1
```

Sprachelemente

Dyadische Operatoren

Mit dyadischen Operatoren werden 2 Operanden miteinander verknüpft.

*	Multiplikation
/	Division
%	Modulo-Division
+	Addition
-	Subtraktion
<<	Schieben nach links
>>	Schieben nach rechts
<	Vergleich auf kleiner
<=	Vergleich auf kleiner oder gleich
>	Vergleich auf größer
>=	Vergleich auf größer oder gleich

==	Vergleich auf gleich
!=	Vergleich auf ungleich
&	Bitweises Und
^	Bitweises Exklusiv-Oder
|	Bitweises Oder
&&	Logisches Und
||	Logisches Oder

Sprachelemente

Blockkommentar

Beginn: /*
Ende: */

Beispiel:

```
/*************************************
        Author:   Hr. Mustermann
*************************************/
```

Zeilenkommentar

Beginn: //
Ende: <Zeilenende>

Beispiel:

```
Ventil = 1;          // Einschalten Ventil
Hupe = 0;
```

Definitionen

Durch das Schlüsselwort „static“ vor der Definition, ist der Geltungsbereich der Definition innerhalb der Quell-Datei.

Objekte

System-Register und digitale Ein-/Ausgänge

```
object          Zeitgeber          at      VT:100;
static object   Ventile            at      BO.3:4;
```

Variablen

Pool-Register (Px.n, Fx.n, Ax.n, VI.n, VF.n, VT.n, BF.n)

```
pool            Zaehler            at      VI.34;
```

Definitionen

Eindimensionale Felder Pool-Register

```
pool            Zaehler[20]          at      VI.34;

Zaehler[0]              -> VI.34
Zaehler[1]              -> VI.35
   :                       :
Zaehler[19]             -> VI.53
```

Konstanten

```
const           Karl                 =       -127;
const           Jane                 =       0xFF;
const           Summe                =       3.456;
const           Mix                  =       1.2 * 7 + Karl;
```

Steueranweisungen

```
if (ausdruck)
        anweisung1;
else if
        anweisung2;
else
        anweisung3;
```

```
switch (ausdruck)
{
        case konstante1:
                anweisung1;
                break;
        case konstante2:
                anweisung2;
                break;
        default:
                anweisungn+1;
                break;
}
```

```
while (ausdruck)
        anweisung;
```

```
repeat
        anweisung;
until (ausdruck);
```

```
goto marke;
marke:
```

Steueranweisung if-Auswahl

if (ausdruck)
anweisung$_1$;
else if
anweisung$_2$;
else
anweisung$_3$;

Beispiel:

```
if (Auswahl == 1)
        Ausgang_1 = 1;
else if (Auswahl == 2)
        Ausgang_2 = 1;
else
{
        Ausgang_1 = 0;
        Ausgang_2 = 0;
}
```

Steueranweisung switch-Auswahl

```
switch (ausdruck)
{
        case konstante1:
                anweisung1;
                break;
        case konstante2:
                anweisung2;
                break;
        default:
                anweisungn+1;
                break;
}
```

Beispiel:

```
switch (Eingang_1 * 2 + Eingang_0)
{
        case 0b00:
        case 0b01:
                Ausgang_1 = 1;
                Ausgang_2 = 0;
                break;
        case 0b01 + 2:
                Ausgang_1 = 0;
                Ausgang_2 = 1;
                break;
        default:
                Ausgang_1 = 0;
                Ausgang_2 = 0;
                break;
}
```

Steueranweisung while-Schleife

while (ausdruck)
    anweisung;

Beispiel:

```
while (Zeitgeber != 0.0);

while (Zeitgeber != 0.0)
{
        Ventil = 1;

        if (Stoerung == 1)
                break;
        if (Schalter == 1)
                continue;

        Ventil = 0;
}
```

Steueranweisung repeat/until-Schleife

repeat
    anweisung;
until (ausdruck);

Beispiel:

```
repeat
{
        Ventil = 1;
        Hupe = 0;

        if (Stoerung == 1)
                break;
        if (Schalter == 1)
                continue;

        Ventil = 0;
        Hupe = 1;
}
until (Zeitgeber == 0.0);
```

Steueranweisung goto-Sprunganweisung

goto – Sprunganweisung nur innerhalb einer Funktion.

```
goto Abschluss;

        :
        :

Abschluss:
```

Funktionen

- Erste übersetzte Funktion ist Start- bzw. Hauptfunktion (beliebiger Funktionsname)
- Bis zu 2 Argumente (Ausnahme Startfunktion)
- Optionaler Rückgabewert (Ausnahme Startfunktion)
- Bis zu 4 lokale Variablen
- Datentypen für Rückgabewerte, Argumente und lokale Variablen

long	Ganzzahlwert
double	Gleitpunktwert

- Schlüsselwort „void“ wenn kein Rückgabewert oder keine Argumente
- Schlüsselwort „return“ für Rückgabewerte
- Schlüsselwort „static“ für Geltungsbereich der Funktion innerhalb der Quell-Datei

Beispiele:

```
void zyklus (void)
void ablauf (long modus, double level)
static double analog (long pos)
```

Funktions-Beispiele

```
pool    Modus           at VI.0;
const   Hand            = 1;
const   Auto            = 2;

void zyklus (void)
{
        while (1)
        {
                if (Modus == Hand)
                        hand (3);
                else if (Modus == Auto)
                        auto (7);
                else
                        service (5, 6.3);
        }
}
```

Funktions-Beispiele

```
object  Analog1         at VF:900;
object  Analog2         at VF:901;

double analog (long pos)
{
        double          wert;

        if (pos == 1)
                wert = Analog1;
        else
                wert = Analog2;

        return (wert);
}
```

Eingebaute Funktionen - Mathematische Funktionen

`double sqrt (double arg)`	Quadratwurzel
`double sin (double arg)`	Sinus
`double cos (double arg)`	Cosinus
`double tan (double arg)`	Tangens
`double asin (double arg)`	Arcus-Sinus
`double acos (double arg)`	Arcus-Cosinus
`double atan (double arg)`	Arcus-Tangens
`double log (double arg)`	Natürlicher Logarithmus
`double exp (double arg)`	Exponential-Funktion
`double pow (double bas, double exp)`	Potenzierung

Eingebaute Funktionen - Ereignis-Funktionen

`void event_halt (void)`	Ausführung Programm-Halt
`void event_end (void)`	Ausführung Programm-Ende
`void event_stop (void)`	Anhalten des Programms
`void event_error (void)`	Auftreten eines Alarms
`void event_halt_continue (void)`	Fortsetzung nach Programm-Halt
`void event_stop_continue (void)`	Fortsetzung nach Anhalten
`void event_error_continue (void)`	Fortsetzung nach Alarm

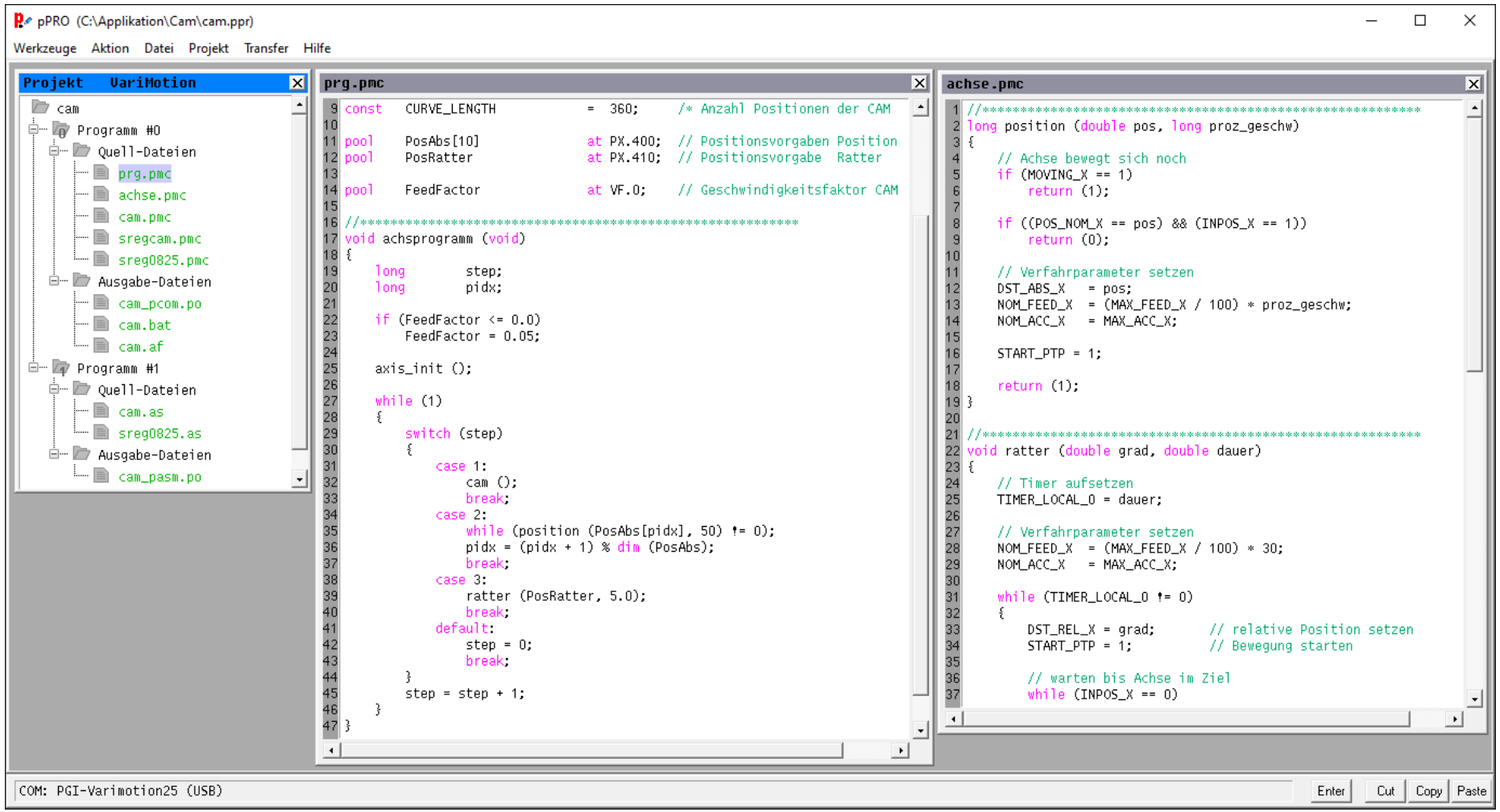
Eingebaute Funktionen - sonstige Funktionen

int	Ganzzahliger Anteil
frac	Gebrochener Anteil
abs	Bilden Betrag
dim	Ermitteln der Feldgröße
base	Registernummer von Objekten und Variablen
width	Breite einer Bitgruppe

Beispiele:

```
Ergebnis = int (Wert * 3.0);

Ergebnis = frac (Wert * 3.0);

Ergebnis = abs (Wert * 3.0);

pool Zaehler[32] at VI.44;
anzahl = dim (Zaehler);

object Timer at VT:112;
nummer = base (Timer);

object Code at BI.113:4;
nummer = width (Code);
```

Projektierungswerkzeug pPRO




```
const   CURVE_LENGTH            =  360;     /* Anzahl Positionen der CAM

pool    PosAbs[10]              at PX.400;  // Positionsvorgaben Position
pool    PosRatter               at PX.410;  // Positionsvorgabe  Ratter

pool    FeedFactor              at VF.0;    // Geschwindigkeitsfaktor CAM

//*************************************************************
void achsprogramm (void)
{
    long        step;
    long        pidx;

    if (FeedFactor <= 0.0)
        FeedFactor = 0.05;

    axis_init ();

    while (1)
    {
        switch (step)
        {
            case 1:
                cam ();
                break;
            case 2:
                while (position (PosAbs[pidx], 50) != 0);
                pidx = (pidx + 1) % dim (PosAbs);
                break;
            case 3:
                ratter (PosRatter, 5.0);
                break;
            default:
                step = 0;
                break;
        }
        step = step + 1;
    }
}
```

```
//*************************************************************
long position (double pos, long proz_geschw)
{
    // Achse bewegt sich noch
    if (MOVING_X == 1)
        return (1);

    if ((POS_NOM_X == pos) && (INPOS_X == 1))
        return (0);

    // Verfahrparameter setzen
    DST_ABS_X   = pos;
    NOM_FEED_X  = (MAX_FEED_X / 100) * proz_geschw;
    NOM_ACC_X   = MAX_ACC_X;

    START_PTP = 1;

    return (1);
}

//*************************************************************
void ratter (double grad, double dauer)
{
    // Timer aufsetzen
    TIMER_LOCAL_0 = dauer;

    // Verfahrparameter setzen
    NOM_FEED_X  = (MAX_FEED_X / 100) * 30;
    NOM_ACC_X   = MAX_ACC_X;

    while (TIMER_LOCAL_0 != 0)
    {
        DST_REL_X = grad;       // relative Position setzen
        START_PTP = 1;          // Bewegung starten

        // warten bis Achse im Ziel
        while (INPOS_X == 0)
```