pCOM

object-oriented compiler

promicon SYSTEMS

pCOM ⟺ pASM

Common features

- The program files created (*.po) contain the same executable instructions
- The same data types
- Source code can be distributed across several files (*.af)
- Generated code can be used in older systems

Differences	**pCOM**	**pASM**
Source code	similar to the high-level language C	instruction list
File extension	*.pmc	*.as
Upper/lower case	relevant	not relevant
Temporary variables	local variables	arithmetic registers

Language elements

Identifier

permitted characters: _ / a ... z / A ... Z / 0 ... 9

Example: _Abakus_2

Constants

integer values and floating-point values

Example:

```
-37.82          floating-point value

1234            integer dec. representation
0x5Ab4          integer hex. representation
0B10010101      integer bin. representation
‘S‘             integer asc. representation
```

Language elements

Monadic operators

A monadic operator refers to only one operator.

-	arithmetic negation
!	logical negation
~	complement

Example:

```
val = !(!4);   // val has the value 1
```

Language elements

Dyadic operators

Dyadic operators are used to link 2 operands together.

*	Multiplication	==	Comparison to equal to
/	Division	!=	Comparison to unequal to
%	Modulo division	&	Bitwise AND
+	Addition	^	Bitwise XOR
-	Subtraction	|	Bitwise OR
<<	Shift to the left	&&	Logical AND
>>	Shift to the right	||	Logical OR
<	Comparison to less than		
<=	Comparison to less than or equal to		
>	Comparison to greater than		
>=	Comparison to greater than or equal to		

Language elements

Block comments

Begin: /*
End: */

Example:

```
/*************************************
        Author:    Mr. Sample
*************************************/
```

Line comments

Begin: //
End: <End of line>

Example:

```
Valve = 1;          // turn on valve
Horn = 0;
```

Definitions

The keyword “static” before the definition means that the scope of the definition is within the source file.

Objects

System registers and digital In-/Outputs

```
object          Timer                   at      VT:100;
static object   Valve                   at      BO.3:4;
```

Variables

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

```
pool            Counter                 at      VI.34;
```

Definitions

One dimensional array

Pool register

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

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

Constants

```
const           Carl                    =       -127;
const           Jane                    =       0xFF;
const           Sum                     =       3.456;
const           Mix                     =       1.2 * 7 + Carl;
```

Control blocks

```
if (expression)
        statement1;
else if
        statement2;
else
        statement3;
```

```
switch (expression)
{
        case constant1:
                statement1;
                break;
        case constant2:
                statement2;
                break;
        default:
                statementn+1;
                break;
}
```

```
while (expression)
        statement;
```

```
repeat
        statement;
until (expression);
```

```
goto label;
label:
```

Control blocks if-selection

if (expression)
statement$_1$;
else if
statement$_2$;
else
statement$_3$;

Example:

```
if (selection == 1)
        output_1 = 1;
else if (selection == 2)
        output_2 = 1;
else
{
        output_1 = 0;
        output_2 = 0;
}
```

Control blocks switch-selection

```
switch (expression)
{
        case constant1:
                statement1;
                break;
        case constant2:
                statement2;
                break;
        default:
                statementn+1;
                break;
}
```

Example:

```
switch (input_1 * 2 + input_0)
{
        case 0b00:
        case 0b01:
                output_1 = 1;
                output_2 = 0;
                break;
        case 0b01 + 2:
                output_1 = 0;
                output_2 = 1;
                break;
        default:
                output_1 = 0;
                output_2 = 0;
                break;
}
```

Control blocks while-loop

while (expression)
    statement;

Example:

```
while (timer != 0.0);

while (timer != 0.0)
{
        valve = 1;

        if (disruption == 1)
                break;
        if (switch == 1)
                continue;

        valve = 0;
}
```

Control blocks repeat/until-loop

repeat
    statement;
until (expression);

Example:

```
repeat
{
        valve = 1;
        horn = 0;

        if (disruption == 1)
                break;
        if (switch == 1)
                continue;

        valve = 0;
        horn = 1;
}
until (timer == 0.0);
```

Control blocks goto-jump statement

goto – jump statement only within a function.

```
goto Termination;

        :
        :

Termination:
```

Functions

- First translated function is start or so called main function (any function name)
- Up to 2 arguments (Exception start function)
- Optional return value (Exception start function)
- Up to 4 local variables
- Data types for return values, arguments and local variables

  long integer value

  double floating-point value
- Keyword „void“ if no return value or no arguments
- Keyword „return“ for return values
- Keyword „static“ for the scope of the function within the source file

Example:

```
void cycle (void)
void sequence (long mode, double level)
static double analog (long pos)
```

Function example

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

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

Function example

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

double analog (long pos)
{
        double          value;

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

        return (value);
}
```

Built-in functions - mathematical functions

`double sqrt (double arg)`	Square root
`double sin (double arg)`	Sine
`double cos (double arg)`	Cosine
`double tan (double arg)`	Tangent
`double asin (double arg)`	Arcus sine
`double acos (double arg)`	Arcus cosine
`double atan (double arg)`	Arcus tangent
`double log (double arg)`	Natural logarithm
`double exp (double arg)`	Exponential function
`double pow (double bas, double exp)`	Power function

Built-in functions – event functions

`void event_halt (void)`	Execution program halt
`void event_end (void)`	Execution program end
`void event_stop (void)`	Program interruption
`void event_error (void)`	Occurrence of an alarm
`void event_halt_continue (void)`	Continue after program halt
`void event_stop_continue (void)`	Continue after interruption
`void event_error_continue (void)`	Continue after an alarm

Built-in functions - other functions

int	form integer part
frac	form fractional part
abs	form absolute value
dim	identify the array size
base	register number of objects and variables
width	width of a bit group

Example:

```
result = int (Value * 3.0);

result = frac (Value * 3.0);

result = abs (Value * 3.0);

pool Counter[32] at VI.44;
quantity = dim (Counter);

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

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

Projecting tool pPRO