Module example

```
Module negout
Title 'Functional example of active low output signals'

Declarations

      clock pin;
      in1   pin;
      in2   pin;

      !out1 pin istype 'reg'; " active low signal
      out2  pin istype 'reg'; " active high signal
      !out3 pin istype 'reg'; " active low signal
      out4  pin istype 'reg'; " active high signal

      x, c, H, L = .x., .c., 1, 0 ;        " Constant assignment
      outputs = [out1,out2,out3,out4];     " Set Declaration

Equations

      out1 := in1  &  in2 ;
      out2 := in1  &  in2 ;
      out3 := in1  &  in2 ;
      out4 := in1  &  in2 ;

      outputs.clk  =  clock ;

Test_vectors ([ clock, in1, in2 ] -> [ out1, out2, out3, out4 ])
      [ 0 , x , x ] -> [ x , x , x , x ];
      [ c , 0 , 0 ] -> [ L , L , L , L ];
      [ c , 1 , 1 ] -> [ H , H , H , H ];
      [ c , 0 , 1 ] -> [ L , L , L , L ];

End negout
```

Equations

```
Module comp4
Title '4-bit look-ahead comparator'

Declarations

      A3..A0,B3..B0                  pin;
      E3..E0                         pin istype 'com';
      A_EQ_B,A_NE_B,A_GT_B,A_LT_B    pin istype 'com';

      A = [A3,A2,A1,A0];
      B = [B3,B2,B1,B0];
      E = [E3,E2,E1,E0];

Equations

      E = A !$ B;              " intermediate An = Bn

      A_EQ_B = E3 & E2 & E1 & E0;    " A=B
      A_NE_B = !A_EQ_B;        " A!=B
      A_GT_B = (A3>B3) #             " A>B
               E3 & (A2>B2) #
               E3 & E2 & (A1>B1) #
               E3 & E2 & E1 & (A0>B0);
      A_LT_B = !A_GT_B & !A_EQ_B;    " A<B

End comp4
```

Truth tables

```
Module bcd2led
Title '7-segment display decoder'

"          a
"         ---       BCD-to-seven-segment decoder similar to the 7449
"       f| g |b
"         ---           segment identification
"       e| d |c
"         ---

Declarations

      D3,D2,D1,D0      pin;
      a,b,c,d,e,f,g    pin istype 'com';

      bcd      = [D3,D2,D1,D0];
      led      = [a,b,c,d,e,f,g];

      ON,OFF   = 0,1;                   " for common anode LEDs
      L,H,X,Z  = 0,1,.X.,.Z.;

@dcset
Truth_table (bcd -> [ a ,  b ,  c ,  d ,  e ,  f ,  g ])
               0   -> [ ON,  ON,  ON,  ON,  ON,  ON, OFF];
               1   -> [OFF,  ON,  ON, OFF, OFF, OFF, OFF];
               2   -> [ ON,  ON, OFF,  ON,  ON, OFF,  ON];
               3   -> [ ON,  ON,  ON,  ON, OFF, OFF,  ON];
               4   -> [OFF,  ON,  ON, OFF, OFF,  ON,  ON];
               5   -> [ ON, OFF,  ON,  ON, OFF,  ON,  ON];
               6   -> [ ON, OFF,  ON,  ON,  ON,  ON,  ON];
               7   -> [ ON,  ON,  ON, OFF, OFF, OFF, OFF];
               8   -> [ ON,  ON,  ON,  ON,  ON,  ON,  ON];
               9   -> [ ON,  ON,  ON,  ON, OFF,  ON,  ON];

End bcd2led;
```

Multi-way conditional assignments

```
Module MUX41
Title '4 to 1 multiplexer design with when then else construct'

Declarations
      SEL0..SEL1 pin;
      A,B,C,D    pin;
      MUX_OUT    pin istype 'com';

      SEL = [SEL1,SEL0];

Equations
      when SEL==0 then MUX_OUT = A; else
      when SEL==1 then MUX_OUT = B; else
      when SEL==2 then MUX_OUT = C; else
      when SEL==3 then MUX_OUT = D;

End MUX41
```

```
Module Counter
Title '4-bit synchronous counter with count enable, asynchronous reset and synchronous
load'

Declarations
      CLK             pin;
      RESET           pin;
      CE, LOAD, DIR   pin;
      DIN3..DIN0      pin;
      COUNT3..COUNT0 pin istype 'reg';

      DIN = [DIN3..DIN0];
      COUNT = [COUNT3..COUNT0];

Equations
      COUNT.CLK = CLK;
      COUNT.ACLR = RESET;

      when LOAD then
            COUNT := DIN;
      else when !CE then
            COUNT := COUNT;
      else when DIR then
            COUNT := COUNT + 1;
      else
            COUNT := COUNT - 1;

End Counter
```

State diagrams

```
Module counter
Title 'Decimal Up Counter'

Declarations

      Clk         pin;
      Q3..Q0      pin istype 'reg';


      " Counter States
      S0 = ^b0000; S4 = ^b0100; S8 = ^b1000;
      S1 = ^b0001; S5 = ^b0101; S9 = ^b1001;
      S2 = ^b0010; S6 = ^b0110;
      S3 = ^b0011; S7 = ^b0111;

      CNT = [Q3,Q2,Q1,Q0];

Equations

      CNT.CLK = Clk;

State_diagram [Q3,Q2,Q1,Q0]

      State S0:   goto S1;
      State S1:   goto S2;
      State S2:   goto S3;
      State S3:   goto S4;
      State S4:   goto S5;
      State S5:   goto S6;
      State S6:   goto S7;
      State S7:   goto S8;
      State S8:   goto S9;
      State S9:   goto S0;

End counter
```

State diagrams with conditional transitions

```
Module counter
Title 'Decimal Up/Down Counter with Synchronous Clear'

Declarations

      Clk         pin;
      Clr,Dir     pin;
      Q3..Q0      pin istype 'reg';


      " Counter States
      S0 = ^b0000; S4 = ^b0100; S8 = ^b1000;
      S1 = ^b0001; S5 = ^b0101; S9 = ^b1001;
      S2 = ^b0010; S6 = ^b0110;
      S3 = ^b0011; S7 = ^b0111;

      CNT = [Q3,Q2,Q1,Q0];

Equations

      CNT.CLK = Clk;

State_diagram [Q3,Q2,Q1,Q0]

      State S0:   if Clr then S0
                  else if Dir then S1
                  else S9;
      State S1:   if Clr then S0
                  else if Dir then S2
                  else S0;
      State S2:   if Clr then S0
                  else if Dir then S3
                  else S1;
      State S3:   if Clr then S0
                  else if Dir then S4
                  else S2;
      State S4:   if Clr then S0
                  else if Dir then S5
                  else S3;
      State S5:   if Clr then S0
                  else if Dir then S6
                  else S4;
      State S6:   if Clr then S0
                  else if Dir then S7
                  else S5;
      State S7:   if Clr then S0
                  else if Dir then S8
                  else S6;
      State S8:   if Clr then S0
                  else if Dir then S9
                  else S7;
      State S9:   if Clr then S0
                  else if Dir then S0
                  else S8;

End counter
```

State diagrams with state transition-triggered actions

```
Module counter
Title 'Decimal Up/Down Counter with Synchronous Clear and Overflow Indicator'

Declarations
      Clk, Clr,Dir      pin;
      Q3..Q0, Ov        pin istype 'reg';

      " Counter States
      S0 = ^b0000; S4 = ^b0100; S8 = ^b1000;
      S1 = ^b0001; S5 = ^b0101; S9 = ^b1001;
      S2 = ^b0010; S6 = ^b0110;
      S3 = ^b0011; S7 = ^b0111;
      CNT = [Q3,Q2,Q1,Q0];

Equations
      CNT.CLK = Clk;
      Ov.CLK = Clk;
      when !Clr then Ov := 0;

State_diagram [Q3,Q2,Q1,Q0]
      State S0:   if Clr then S0
                  else if Dir then S1 with
                        Ov := 0;
                  endwith;
                  else S9 with
                        Ov := 1;
                  endwith;
      State S1:   if Clr then S0
                  else if Dir then S2
                  else S0;
      State S2:   if Clr then S0
                  else if Dir then S3
                  else S1;
      State S3:   if Clr then S0
                  else if Dir then S4
                  else S2;
      State S4:   if Clr then S0
                  else if Dir then S5
                  else S3;
      State S5:   if Clr then S0
                  else if Dir then S6
                  else S4;
      State S6:   if Clr then S0
                  else if Dir then S7
                  else S5;
      State S7:   if Clr then S0
                  else if Dir then S8
                  else S6;
      State S8:   if Clr then S0
                  else if Dir then S9
                  else S7;
      State S9:   if Clr then S0
                  else if Dir then S0 with
                        Ov := 1;
                  endwith;
                  else S8 with
                        Ov := 0;
                  endwith;

End counter
```

Hierarchical design example

```
" ---- top-level module ----

Module ANDORINV
Title 'And - Or - Invert gate'

Declarations
      AND interface ( A,B -> Y );
      NOR interface ( A,B -> Y );

      A1,A2        functional_block AND;
      N            functional_block NOR;

      I1..I4       pin;
      Y            pin istype 'com';
      N1,N2        node istype 'com';

Equations
      A1.A = I1;
      A1.B = I2;
      N1 = A1.Y;

      A2.A = I3;
      A2.B = I4;
      N2 = A2.Y;

      N.A = N1;
      N.B = N2;
      Y = N.Y;

End ANDORINV

" ---- lower-level modules ----

Module AND
Interface ( A,B -> Y );

Title 'Two input AND gate'

Declarations
      A,B pin;
      Y   pin istype 'com';

Equations
      Y = A & B;

End AND

Module NOR
Interface ( A,B -> Y );

Title 'Two input NOR gate'

Declarations
      A,B pin;
      Y   pin istype 'com';

Equations
      Y = !( A # B );

End NOR
```

Test vectors

```
Module COMP4
Title '4-bit comparator'

 No,Yes = 0,1;

Declarations
      A3..A0,B3..B0                pin;
      A_EQ_B,A_NE_B,A_GT_B,A_LT_B   pin istype 'com';

      A = [A3..A0];
      B = [B3..B0];

Equations                       " 4-bit comparator
 A_EQ_B =   A == B;
 A_NE_B = !(A == B);
 A_GT_B =   A > B;
 A_LT_B = !((A > B) # (A == B));

Test_vectors  'test for A = B'
       ([ A, B] -> [A_EQ_B, A_GT_B, A_LT_B, A_NE_B])
        [ 0, 0] -> [ Yes  ,   No  ,   No  ,   No  ];
        [ 2, 2] -> [ Yes  ,   No  ,   No  ,   No  ];
        [15,15] -> [ Yes  ,   No  ,   No  ,   No  ];

Test_vectors  'test for A > B'
       ([ A, B] -> [A_EQ_B, A_GT_B, A_LT_B, A_NE_B])
        [ 2, 1] -> [  No  ,  Yes  ,   No  ,  Yes  ];
        [ 8, 7] -> [  No  ,  Yes  ,   No  ,  Yes  ];
        [ 5, 0] -> [  No  ,  Yes  ,   No  ,  Yes  ];

Test_vectors  'test for A < B'
       ([ A, B] -> [A_EQ_B, A_GT_B, A_LT_B, A_NE_B])
        [14,15] -> [  No  ,   No  ,  Yes  ,  Yes  ];
        [ 7, 8] -> [  No  ,   No  ,  Yes  ,  Yes  ];
        [ 2, 8] -> [  No  ,   No  ,  Yes  ,  Yes  ];

End COMP4
```

```
Module Counter
Title '4-bit counter with synchronous reset'

Declarations
      Clk,Clear   pin;
      Q3..Q0      pin istype 'reg';

      Count = [Q3..Q0];

Equations
     Count.clk = Clk;
     Count := (Count.fb + 1) & !Clear;

Test_vectors
    ( [ Clk, Clear ] -> [Count] )
      [ .C.,   1   ] -> [ 0   ];  "clear the counter
      [ .U.,   0   ] -> [ 1   ];  "trigger the counter to next state
      [ .D.,   0   ] -> [ 1   ];  "transition down

End Counter
```

Other examples

```
Module DFFAR
Title 'D F/F with asynchronous Reset'
        CLK    pin;
        RESET  pin;
        DIN    pin;
        DOUT   pin istype 'reg';

Equations
      DOUT.CLK = CLK;
      DOUT.ACLR = RESET;
      DOUT := DIN;

End DFFAR
```

```
Module DFFSR
Title 'D F/F with synchronous Reset'
        CLK    pin;
        RESET  pin;
        DIN    pin;
        DOUT   pin istype 'reg';

Equations
      DOUT.CLK = CLK;
      DOUT.CLR = RESET;
      DOUT := DIN;

End DFFSR
```

```
Module BUF3
Title 'Tristate Buffer'
        DENABLE pin;
        DIN     pin;
        DOUT    pin istype 'com';

Equations
      DOUT.OE = DENABLE;
      DOUT = DIN;

End BUF3
```

```
Module SHREG
Title '4-bit serial-in and serial-out shift register'

Declarations
        CLK    pin;
        DIN    pin;
        DOUT   pin istype 'reg';

        REG2..REG0 node istype 'reg';
        REG = [REG2..REG0,DOUT];

Equations
      REG.CLK = CLK;
      REG := [DIN,REG2..REG0];

End SHREG
```