Estimo C Language Reference Manual®

The Embedded Motion Estimation Language.

George Vafiadis [vafiadis@ieee.org]

Comments

```
// this is a single line comment

/* this is a multiline

   comment written is estimo C

*/
```

Variables Assignments

Variables should be initialized with a value. You are able to assign to them expressions or numeric constants.

```
a = 122;

b = 12.10;

c = (a+b)/2.0;
```

Conditional Statements

Conditional Statements targeting the interpreter (running at PC)

a > b, a < c, a >= b, a <= b, a == b, a && b, a || b, a != b, !a

```
if( a > 12 && b < 10)
{

}

if( a <= 12 )
{

}
else
{

}
```

Loop Statements

break; command could be used to exit from a loop. (Exit from the first loop).

If you need to break a loop (go after the loop) in FPGA level then you should use

#break; command. All the classical statements exist “for”, “while”, “do-while”. All are running on FPGA level.

```
for(i = 0 to 10 step 1)
{

}

while( k < 10 )
{

}

do
{

} while( var >= 0 );
```

Define Pattern

```
Pattern(hexagon)
{
 check(1,2)
 check(2,0)
 check(1,-2)
 check(-1,-2)
 check(-2,0)
 check(-1,2)
}
```

Check Pattern

With predefined static pattern:

```
check(hexagon);
```

Pattern defined using single checks, full scriptable support (variables can be used instead of values):

```
 check(1,2);
 check(2,0);
 check(1,-2);
 check(-1,-2);
 check(-2,0);
 check(-1,2);
 update;                /// collects all the checks, form a new pattern and check it
```

Conditional based on Winning Point

```
/// Example 15: Dr Jose sample code
iter = 4;
Pattern(hexagon)
{
check(1,2)
check(2,0)
check(1,-2)
check(-1,-2)
check(-2,0)
check(-1,2)
}
check(0,0);
update;
for(loop = 1 to iter step 1)
{
  check(hexagon);
    #if( WINID == 0 )
        #break;
}
// Fractional refinement
check(0.5,0);
check(-0.5,0);check(1.5,0);
check(0,0.5);check(0,-0.5);
check(0.25,0);
update;
```

Check the WINID register if equals a point execute the block of code [running on FPGA level] (you can also use #else)

The #break command exits the loop.

This command is running on FPGA level (is different from 'break' without the sharp, running at interpreter level). The #break is like jump to the end of the loop

Caution: No Optimization is performed. Use check(){case } if you need optimization.

Check Pattern and Refine Based on the Winning Point [Optimized]

Check Pattern and perform operation based on the result of the check. After a check pattern command the WINID register is updated with the winning point ID. Inside the cases, **the first** check pattern will be modified (optimization) based on the winning point in order to avoid re-checking points already checked.

NOTE: The case should not be "**case 0:" If you need to check it you should use it after the**

Case like #if(WINID == 0) {}

```
check(hexagon)
{
  case 1:
    check(hexagon);    /// <- optimizer will modify this pattern based on #1
    check(refine2);    ///  another check pattern

  case 2:
    check(hexagon);    /// <- optimizer will modify this pattern based on #2
    check(refine2);    ///  another check pattern


  case 3:
    check(hexagon);    /// <- optimizer will modify this pattern based on #3
    check(refine2);    ///  another check pattern

}
```

SharpEye Studio

The SharpEye Studio has been modified to show a useful dialog after the compilation with the output. In order to show the result the IDE is running the sharpinfo.exe, this little utility can be used outside of the IDE in order to decompile the result of point memory and program memory.

SharpInfo: Decompiler for Estimo C v2.5

USAGE: sharpinfo -pro|-mem <input-file>

-pro Output Program Info

-mem Output Point Memory Info