

Additional Quick Start information regarding use of the PL/i Programming Language in i.CanDoIt

The compiler embedded within the web server of i.CanDoIt is useful for compiling smaller programs. However, due to the compiler's liberal use of the memory pool, it is likely to run out of memory when compiling larger programs. In addition, as the same program is compiled repeatedly (like when debugging), the memory becomes fragmented. This means that while the total amount of available memory is sufficient, due to fragmentation, the system no longer sees that memory as being available in a contiguous chunk.

If you have successfully compiled a program one or more times, then get an error message referring to "heap full" or some other memory related issue, you can recover the fragmented memory by simply restarting the system. If you get a memory related error message immediately after restart, your program may have outgrown the internal capacity of the compiler.

A PC based stand alone compiler is available. When you run this on your PC, you will generally have plenty of memory available. Furthermore, each time you run the compiler on the PC, you are effectively "restarting the system" and reclaiming any fragmented memory.

Even on the PC, it is still possible to get a stack overflow. Unfortunately the command line compiler is not graceful about telling you this. It simply stops and does not produce any output. The PC compiler's stack will overflow if you attempt to compile a program that is several hundred lines of straight in-line code. By this we mean hundreds of lines of code not broken into subroutines (called procedures). Since this is bad practice anyway, we are not particularly worried about the compiler choking on such code. We have found that the exact same 600 lines of code that break the compiler when compiled in-line will compile just fine when broken into as few as 4 subroutines.

Approach to Debugging

We are working on an interactive debugger. In the mean time, since most of the program activity involves doing things with registers, you can get reasonably far just looking at register contents with the browser. You can also place "write" commands to send tags and values to the virtual terminal in the browser window. This amounts to the browser version of the old fashioned approach to debugging through the liberal use of "print" statements. In some cases that is still the better choice since most interactive debuggers slow down the program being debugged, and we don't expect ours to be any exception in that regard.

What does PL/i look like?

The program listed to the right is an example program that compiles without errors even though it doesn't really do anything useful. It's purpose is only to show some of the features of the language.

PL/i language help, including full syntax definition, is available on-line in the web pages served by the device itself. If you do not have access to the device, a demo copy of the device's web site may be found at www.csimn.com (after 4/1/2007).

How to use the PL/i Command Line Compiler

The compiler is available at <ftp://ftp.csimn.com>. Download "icandoit_compiler.zip".

Refer to the example screen shot below. Simply type the command "icandoit" followed by the program source code file. Use any text editor (e.g. Notepad) to create your file. It must have a .pli suffix before the compiler will create a code file by the same name with a .plx suffix. Once you have successfully compiled your program and created a "plx" file, upload it into the i.CanDoIt server, select it, and click Run.

Support

Support is still free, and we intend to keep it that way. But we would appreciate it if you help us help you. For the most efficient handling of your inquiry, please address technical and "how to" questions via email to support@csimn.com. If you have questions about a particular programming issue, it will be especially helpful if you attach a sample of your code to the email.

A growing number of help pages, including demo programs, will be appearing at www.csimn.com. Look for the "Support" and "Downloads" buttons in the left-hand column on our updated web site after April 1, 2007.

```
program test

declare
  type myRec =
    record
      ival1: int;
      ival2: int;
      bval1: bit;
      bval2: bit;
      fval1: float;
      fval2: float;
    end;
  type myRecordSet = array [1..20] of myRec;

myVar: myRecordSet;
var1: int;
var2: int;
var3: float;
var4: boolean;
var5: int;
i: int;
stuff: array [1..10] of int;

#RoomTemp101 = 223;
#RoomTemp22 = 224;
#RoomTemp120 = 225;

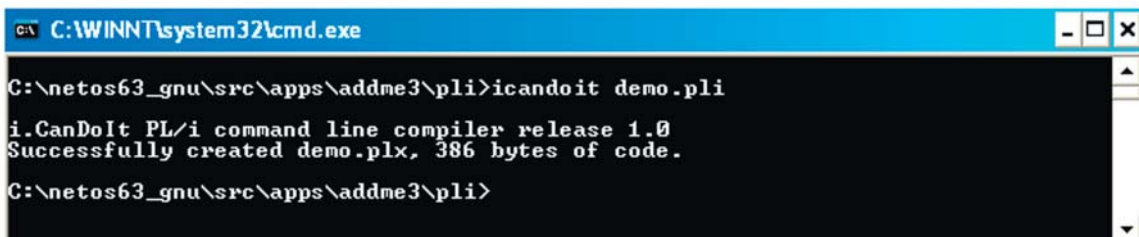
procedure update_me (var n: myRec)

declare
  j: int;

  procedure update_you (var k: int)
  declare
  begin
    seti (102, k);
  end;

begin
  n.ival1 = geti (104);
  n.fval1 = getf (1015);
  update_you (n.ival1);
end;

begin
  var1 = geti (#RoomTemp101);
  var2 = geti (#RoomTemp22);
  for i = 1 to 10
    begin
      myVar[i].ival1 = i * 10;
      var3 = i;
      myVar[i].fval1 = var3 / 2.0;
    end;
  for i = 10 down to 1
    update_me (myVar[i]);
  select
    when (var1 = var2) and (var1 = var5)
      begin
        seti (98, 3);
        seti (99, 3);
      end;
    when var1 = var3 seti (98, 0);
    when var2 = var3 seti (99, 0);
    otherwise seti (97, 2);
  end;
end
```



```
C:\WINNT\system32\cmd.exe
C:\netos63_gnu\src\apps\addme3\pli>icandoit demo.pli
i.CanDoIt PL/i command line compiler release 1.0
Successfully created demo.plx, 386 bytes of code.
C:\netos63_gnu\src\apps\addme3\pli>
```

Note: We are working on an interactive debugger, but didn't want to prevent people from at least getting started with a command line compiler.