```
**********************************************************************
* IMPORTANT NOTE:                                                    *
*                                                                    *
* To build this example you will need PSYLINK version 2.27 or later *
*                                                                    *
* To debug this example you need DBUGPSX version 4.70 or later.      *
**********************************************************************
```

This example consists of a main program and 3 overlayed levels.
The example is only a shell to demonstrate how to achieve this.

There is a make file. To build the example execute psymake.

Examine the map file produced (MAIN.MAP). This will show you
that the address for the 3 level's code and data all overlap
in memory. It will also show you the addresses of the global
symbols demonstrating that they are in the same memory area.

Warning : There is no automatic system for loading levels when
a routine is called. It is up to you to make sure that the
level's code and data is in memory before you call it.

Have a look at the files MAKEFILE.MAK and MAIN.LNK.

In the make file, each level is compiled with a parameter like
-Wa,sl1. The -Wa, part of this parameter tells ccpsx to pass the
rest of the parameter to the assembler aspsx. In this instance
specifying -Wa,sl1 causes the parameter -sl1 to be passed to
aspsx. (Add the -v option to the ccpsx command line to see this
being done).

The -s parameter to aspsx tells it to put the code and data into a
special group which is specified after the -s. e.g. -sl1 tells aspsx
to put the code and data into a group called l1.

Notice that the -Wa,s... parameter specifies a different name for each
level.

In MAIN.LNK you can see the declarations for groups l1, l2 and l3.
l2 and l3 are specified as being over l1. This tells the linker that
all these groups should be placed at the same memory address. The
linker allocates enough space in memory to hold the largest of the
groups. When a smaller group is in memory the extra space will be
wasted.

Each group also has a file parameter. This tells the linker not to
write the code for the group into the standard output file. Instead
the linker writes the code as a block of pure binary data into the
file name specified.

If you examine the contents of your directory you will find the output
file MAIN.CPE and also the files L1.BIN, L2.BIN and L3.BIN which
contain the data for each of the levels. These are the files that the
main program should load into memory in the load_level routine.

One further point to note is the use of the -G0 parameter when the levels

are compiled. This stops the compiler from using the global pointer in the level code since the global pointer will be set up to point to the main programs small globals rather than the level code's.

There is one additional module - address.s

This is a short piece of assembly language used to get the address of the load area. This is needed because this address can't be found in C code.