Application Note: AN10013

How to examine the contents of memory

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to examine the contents of memory.

Required tools and libraries

This application note is based on the following components:

- xTIMEcomposer Tools - Version 14.0.0

Required hardware

Programming how-tos are generally not specific to any particular hardware and can usually run on all XMOS devices. See the contents of the note for full details.

1 How to examine the contents of memory

XGDB can be used to examine the contents of memory at a given point in time. For example, compile the following code ensuring that debug is enabled (-g):

```
int global_variable[5] = {0, 1, 2, 3, 4};

int f(int index) {
  return global_variable[index];
}

int main() {
  f(2);
  return 0;
}
```

2 From within xTIMEcomposer Studio

Create a new debug configuration via *Run->debug Configurations->xCORE Applications*. Set a breakpoint at the start of *f* then start debugging. Execution will now break when *f* is reached. The memory contents occupied by the global variable *global_variable* can be seen in the *Memory* view. Click on the *add memory monitor* button on the view toolbar. In the resulting dialog, input 'global_variable' then press *ok*. This will show the address of *global_variable* and the contents of this memory location.

3 From the command line

On the command line, memory contents can examined using the *x* (examine) command. For example, start XGDB, connect to the simulator and set a breakpoint on *f*. When run, execution will break at the start of *f*. You can now display the contents of the memory occupied by *global_variable* using the *x* command as follows:

```
> xgdb a.xe
...etc...
(gdb) connect -s
0xffffc04e in ?? ()
(gdb) break f
Breakpoint 1 at 0x100b2: file examining_memory.xc, line 11.
(gdb) run
...etc...
Breakpoint 1, f (index=2) at examining_memory.xc:11
11    return global_variable[index];
(gdb) x global_variable
0x10274 <global_variable>:  0x00000000
```

Note: The print command accepts an argument specifying the desired format. For example, *x/d* will display the memory contents as decimal instead of the default of hex.

Also, to see the content of an offset other than 0 in the *global_variable* array, (for example, index = 1), the following command can be used:

```
x &global_variable[1]
```