Application Note: AN10001

How to pass alias pointer arguments

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to pass alias pointer arguments.

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 pass alias pointer arguments

By default, pointer arguments are *restricted* pointers, which cannot alias any other pointer. However, it is possible to pass alias pointers by explicitly marking the parameter type as an alias pointer:

```
void func1(int * alias x) {
  printintln(*x);
}

void func2() {
  int i = 55;
  int *p = &i; // This is implicitly an alias pointer
  func1(p);
}
```

If a function parameter is not explicitly marked as alias then it is a non-aliasing pointer. However, it is OK to pass alias pointers to these functions provided that:

- None of the pointers passed in alias each other (if they do, the function call will trap at the point of the call).
- The alias pointer being passed in is of local scope and hasn't been assigned (possibly through several assignments) from a pointer with non-local scope.

For example the following is not allowed:

```
void func3(int *x);

void func4(int * alias x) {
  int *y = x;
  func3(y);  // y gets it's value from x which has non-local scope
}
```

However the following is allowed:

```
void func3(int *x) {
  printintln(*x);
}

void func4() {
  int i = 77;
  int *x = &i;
  func3(x);
}
```

In this case, the local alias pointer is reinterpreted as a non-aliasing pointer for the duration of the function call.