

**Application Note: AN10060** 

## How to dynamically change the type of a port

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to dynamically change the type of a port.

## 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 dynamically change the type of a port

When you declare a port, you declare it as an in or out port or possibly as buffered.

```
out port p = XS1_PORT_1A;
```

Sometimes you need to change the port type dynamically during the execution of a program. This can be done with the reconfigure\_port function.

```
int main() {
```

To reconfigure a port you need to create movable pointers for the port you want to reconfigure and the reconfigured port.

```
out port * movable pp = &p;
buffered out port:32 * movable buffered_p;
```

The reconfigure\_port function works be transferring the ownership of the port from one movable pointer to another. This means that after this function is called you cannot use the original port configuration.

```
buffered_p = reconfigure_port(move(pp), buffered out port:32);
```

After reconfiguring the port, the port can be accessed via the pointer.

```
*buffered_p <: 0xa0a0a0a0;</pre>
```

After the program has finished using the reconfigured port it needs to pass ownership back to the original port pointer (by reconfiguring it back to the original type.

```
pp = reconfigure_port(move(buffered_p), out port);
```



Copyright © 2016, All Rights Reserved.

Xmos Ltd. is the owner or licensee of this design, code, or Information (collectively, the "Information") and is providing it to you "AS IS" with no warranty of any kind, express or implied and shall have no liability in relation to its use. Xmos Ltd. makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.