

Application Note: AN10006

A button handling example

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows a button handling example.

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 A button handling example

To handle buttons a task needs to event when a pin changes value. This can be done using the `select` construct and the `pinsneq` predicate on the select case:

```
// This function is combinable - it can run on a logical core with other tasks.
[[combinable]]
void task1(port p_button)
{
  // The last read value off the port.
  int current_val = 0;
  while (1) {
    select {
      // event when the button changes value
      case p_button when pinsneq(current_val) :=> int new_val:
        if (new_val == 1) {
          printf("Button up\n");
        } else {
          printf("Button down\n");
        }
        current_val = new_val;
        break;
    }
  }
}
```

This code will react when the I/O pins change value. However, due to the button bouncing up and down, after a button is pressed the I/O pin will change value many times, very quickly. To avoid reacting to each of these changes you can add a debouncing period.

To do this, add a guard to the select case. This guard says do not react to the button unless the variable `is_stable` evaluates to true (i.e. non-zero). When a button is pressed `is_stable` is set to 0 and a timeout is setup. A separate case handles this timeout expiring (using a timer) at which point `is_stable` is set back to 1.

```
[[combinable]]
void task1a(port p_button)
{
    int current_val = 0;
    int is_stable = 1;
    timer tmr;
    const unsigned debounce_delay_ms = 50;
    unsigned debounce_timeout;
    while (1) {
        select {
            // If the button is "stable", react when the I/O pin changes value
            case is_stable => p_button when pinsreq(current_val) :> current_val:
                if (current_val == 1) {
                    printf("Button up\n");
                } else {
                    printf("Button down\n");
                }
                is_stable = 0;
                int current_time;
                tmr :> current_time;
                // Calculate time to event after debounce period
                // note that XS1_TIMER_HZ is defined in timer.h
                debounce_timeout = current_time + (debounce_delay_ms * XS1_TIMER_HZ);
                break;

            // If the button is not stable (i.e. bouncing around) then select
            // when we the timer reaches the timeout to reenter a stable period
            case !is_stable => tmr when timerafter(debounce_timeout) :> void:
                is_stable = 1;
                break;
        }
    }
}
```