XK-1A Development Board Tutorial

Document Revision 11.2

An interactive version of this tutorial is available in the XMOS Development Environment. To open, choose **Help ▶ Welcome** and select the XK-1A tutorial.

Publication Date: 2011/03/29

1 Introduction

The XK-1A is a development board based on the XMOS XS1-L1 device. It comprises a single XS1-L1, four LEDs, two press-buttons, SPI flash memory, JTAG and serial interface, and two 16-way IDC connectors for connecting external components.

The XS1-L1 consists of a single XCore that comprises an event-driven processor with tightly integrated general purpose I/O. The processor provides up to eight threads, 400 MIPS and 64 KBytes of RAM. The I/O pins are connected to board components and can be directly controlled by software (see Section 7).

Programs are written using a combination of XC, C and C++. XC provides extensions to C that simplify the control over concurrency, I/O and time. These extensions map directly onto XCore hardware resources such as threads and ports, and are *efficient*—compiling into short instruction sequences, and *safe*—free from many sources of deadlock, race conditions and memory violations. This makes programs easy to write, understand and debug.

This tutorial provides an introduction to writing XC programs using the components integrated on your XK-1A. It assumes that you are familiar with C [?]. This tutorial shows you how to:

- illuminate an LED on the board
- flash an LED at a fixed rate
- flash an LED while cycling it along a 4-LED line in sequence
- create concurrent threads that flash LEDs and respond to button presses

The example programs show you how XC simplifies the creation of programs for event-driven processors. Sections of the tutorial that introduce a new language feature include the new keywords or operators in their title.

The example programs have been tested with version 11.2 of the tools. Information on downloading, installing and using these tools is provided in a separate user guide [?].

2 Illuminate an LED: `port`, `<:`

This part of the tutorial shows you how to illuminate an LED on your XK-1A, using an XC port and an output statement.

A port connects a processor to one or more physical pins and is used to interface with the board components. The port logic can drive its pins high or low, or it can sample the value on its pins.

The program below illuminates a single LED on your XK-1A:

```
#include <platform.h>

out port led = PORT_LED;

int main() {
  led <: 0b0001;
  while(1)
    ;
  return 0;
}
```

The declaration

```
out port led = PORT_LED;
```

declares an output port named `led`, which refers to the 4-bit port identifier `PORT_LED` (see Section 7). This identifier is defined in the board description file “XK-1A.xn” and is exported to the header file `platform.h` during compilation.

Integrated input and output statements make it easy to express I/O operations on ports. In `main`, the statement

```
led <: 0b0001;
```

outputs the value 0b0001 to the port `led`, causing the port to drive one of its four pins high. This in turn causes the LED connected to the pin to illuminate.

The port continues to drive the pins until instructed otherwise or until the program terminates. The infinite loop that follows the output statement prevents the latter condition.

✎ Compile and run this program on your XK-1A. The LED labeled LED0 should illuminate.

✎ Modify the value output to the port so that all four LEDs are illuminated.

3 Flash an LED: `timer`, `:>`

This part of the tutorial shows you how to flash an LED at a fixed rate, using an XC timer and an input statement.

A timer is a hardware resource used to determine when an event happens or to delay execution until a particular time. Each timer contains a 32-bit counter that is incremented at 100MHz and whose value can be input at any time.

The program below flashes an LED on your XK-1A:

```
#include <platform.h>

#define FLASH_PERIOD 20000000

out port led = PORT_LED;

int main(void) {
  timer tmr;
  unsigned isOn = 1;
  unsigned t;
  tmr :> t;
  while (1) {
    led <:  isOn;
    t += FLASH_PERIOD;
    tmr when timerafter(t) :> void;
    isOn = !isOn;
  }
  return 0;
}
```

The statement

```
tmr :> t;
```

inputs the value of the timer `tmr`'s counter into the variable `t`. Having recorded the current time, the statement

```
t += FLASH_PERIOD;
```

increments this value by the required delay, and the statement

```
tmr when timerafter(t) :> void;
```

delays inputting a value until the specified time is reached. The processor must complete an input operation once a condition is met, even if the input value is not required. This is expressed in XC as an input to `void`.

✎ Compile and run this program on your XK-1A. The LED labeled LED0 should flash at the rate defined by the flash period.

4 Flash and cycle LEDs at different rates: `select`

This part of the tutorial shows you how to flash an LED while cycling it between the LEDs on your XK-1A, using the XC `select` statement.

The `select` statement is used to respond to one of a set of inputs, depending on which becomes ready first. If more than one input becomes ready at the same time, only one is executed.

The `select` statement below waits for one of two timeouts to occur:

```
select {
  case tmrF when timerafter(timeF) :> void :
    /* add code to respond to timeout */
    break;
  case tmrC when timerafter(timeC) :> void :
    /* add code to respond to timeout */
    break;
}
```

The program below uses a `select` statement to update the state of the LED flash and cycle algorithms at different rates:

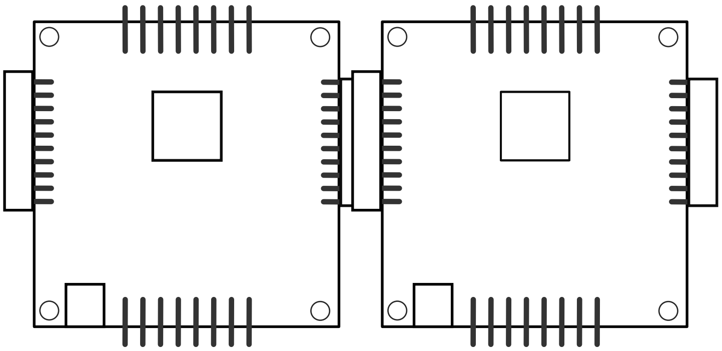
```
#include <platform.h>

#define FLASH_PERIOD 10000000
#define CYCLE_PERIOD 50000000

out port led = PORT_LED;

int main(void) {
  unsigned ledOn = 1;
  unsigned ledVal = 1;
  timer tmrF, tmrC;
  unsigned timeF, timeC;
  tmrF :> timeF;
  tmrC :> timeC;
  while (1) {
    select {
      /* respond to timeouts */
    }
  }
  return 0;
}
```

✎ Complete this program so that on each flash timeout the program changes whether or not the LED is illuminated and on each cycle timeout the program changes which

LED is being flashed. Compile and run this program on your XK-1A. The LEDs should continually flash in sequence.

5 Running concurrent tasks on multiple boards: `par`, `on`

This part of the tutorial shows you how to connect multiple XK-1A boards together and run tasks on them concurrently, using the XC `par` and `on` statements.



The `par` statement provides a simple way to create *concurrent threads* that run independently of one another. The `on` statement instructs the compiler on which processor each port is connected and each thread is executed. The program below creates two concurrent threads, each running an instance of a function that flashes an LED:

```
#include <platform.h>

#define PERIOD 20000000

out port led0 = PORT_LED_0;
out port led1 = PORT_LED_1;

void flashLED(out port led, int period);

int main(void) {
  par {
    on stdcore[0]: flashLED(led0, PERIOD);
    on stdcore[1]: flashLED(led1, PERIOD);
  }
  return 0;
}
```

The header file `platform.h` provides a declaration of the global variable `stdcore`, which can be used to specify the placement of port declarations and threads.

An`on` statement may only be used with threads created by `main`, in which case `main` may contain only channel declarations, a single `par` statement and an optional `return` statement.

The project includes a link to the read-only system file XK-1A.xn, which is the board design file for the XK-1A. By extending this file, you can specify a target platform consisting of two XK-1As.

✎ Right-click on the XN file in the *Project Explorer* and select **Import XN into Project.** Open the XN file in the editor and update it to specify two L1 devices connected together using XMOS links. Information on writing XN files can be found in the tools user guide [**?**, Chapter 8]. Note that the port names used for the LED in the XC are PORT_LED_0 and PORT_LED_1.

Implement the function flashLED, then build and run the project. The LEDs labeled LED0 on each board should flash at the rate defined by the flash period.

✎ Experiment with different period values for each of the threads so that the threads can be seen to be operating independently of one another.

6 What to read next

This tutorial provides only a basic introduction the XC language and XMOS architecture. The following documents provide more information on designing with XC on your XK-1A board:

- **Programming XC on XMOS Devices** [**?**]: Provides an in-depth tutorial on how to write XC programs, including case studies of real-world designs, along with the official XC language specification and details of its implementation on XS1 devices.
- **The XMOS Tools User Guide** [**?**]: Explains how to use the XMOS tools.
- **XK-1A Hardware Manual** [**?**]: Provides a functional description of the XK-1A board including the port-to-pin mappings of the components.

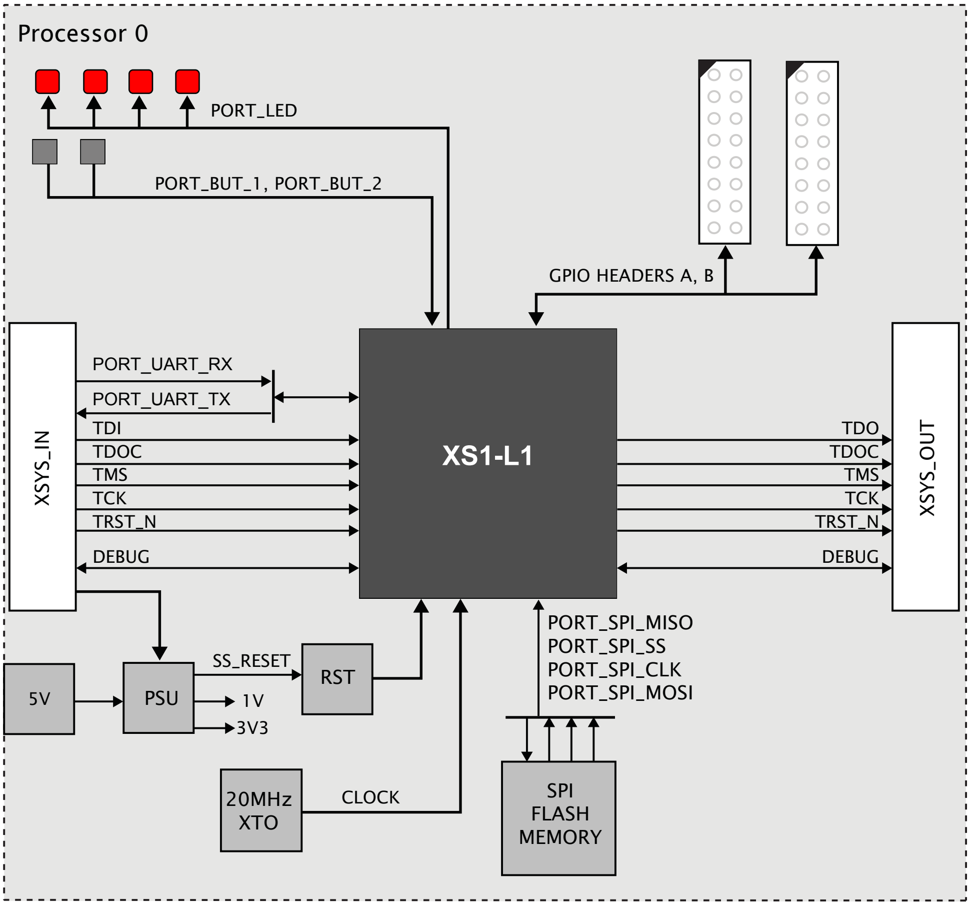
You may also find information from the following online resources useful:

- `http://www.xmos.com/`

- `http://www.xcore.com/`

7 XK-1A port-to-component map

The diagram below shows how components on your XK-1A are connected to the processors on the XS1-L1, and gives the port names used in software.



1 Related Documents

[1] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1988.

[2] Douglas Watt and Huw Geddes. *The XMOS Tools User Guide*. XMOS Limited, 2009. http://www.xmos.com/published/xtools_en.

[3] Douglas Watt. *Programming XC on XMOS Devices*. XMOS Limited, Sep 2009. http://www.xmos.com/published/xc_en.