Application Note: AN00175

A startKIT LED demo

This application shows a very simple program running on the XMOS startKIT development board. It displays an animated pattern on the LEDS on the board by directly writing to the port which is connected to the LEDs.

Required tools and libraries

- xTIMEcomposer Tools - Version 14.0

Required hardware

This application note is designed to run on any XMOS multicore microcontroller.

The example code provided with the application has been implemented and tested on the XMOS startKIT. It depends on the specific mapping of ports to hardware but is quite general in its nature.

Prerequisites

- This document assumes familiarity with the XMOS xCORE architecture, the XMOS GPIO library, the XMOS tool chain and the xC language. Documentation related to these aspects which are not specific to this application note are linked to in the references appendix.
- For descriptions of XMOS related terms found in this document please see the XMOS Glossary[1].

[1] http://www.xmos.com/published/glossary

1 Overview

1.1 Introduction

startKIT is a low-cost development board for the configurable xCORE multicore microcontroller products from XMOS. It's easy to use and provides lots of advanced features on a small, extremely low cost platform.

xCORE lets you software-configure the interfaces that you need for your system; so with startKIT you can configure the board to your match your exact requirements. Its 500MIPS xCORE multicore microcontroller has eight 32bit logical cores that perform deterministically, making startKIT an ideal platform for functions ranging from robotics and motion control to networking and digital audio.

startKIT also connects easily to your Raspberry Pi, allowing you to add real-time I/O and communication features to this popular computing platform, and to try out advanced applications for xCORE.

2 A simple LED example

The example in this application note is one of the simplest programs you can write for the startKIT. It simple drives an moving pattern onto the LEDS on the board.

On the startKIT, the 32-bit port connects to multiple I/O including the LEDs. Figure 1 shows how the LEDs are connected to the port.

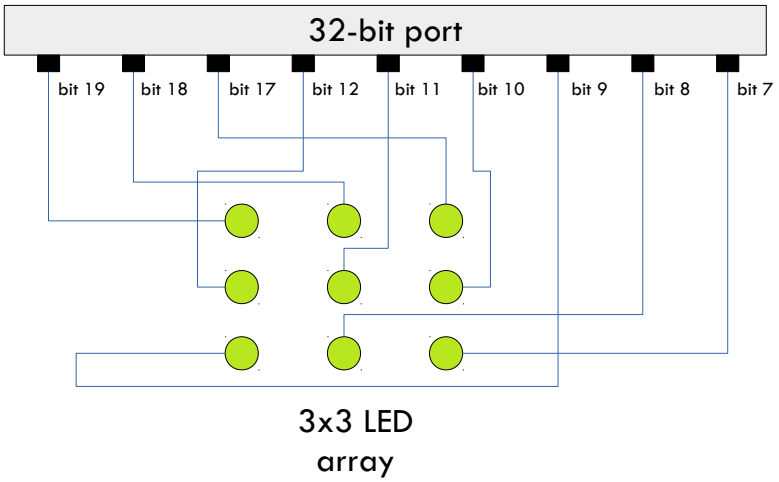

Figure 1: startKIT LED port connection

The LEDs are turned on by driving the port low and turned off by driving the port high.

2.1 The Makefile

The Makefile needs to target the startKIT. So has the line:

```
TARGET = STARTKIT
```

All other parts of the Makefile are just the standard build template used by XMOS projects.

2.2 Application resource declaration

The only resource used by this application is the 32-bit, this needs to be declared in the code:

```
/* This the port where the leds reside */
port p32 = XS1_PORT_32A;
```

2.3 The application main() function

By driving different 32-bit values onto the 32-bit port, different patterns will be shown on the LED. The patterns array in the code defines 4 different patterns that can be driven:

```
#define NUM_PATTERNS 4

int patterns[NUM_PATTERNS] = {
    0xE0380,
    0x61700,
    0xA1680,
    0xC1580
};
```

The main() function of the program justs performs an infinite loop that drives this value onto the port.

```
int main(void) {
    int delay = 50;      // initial delay 50 ms
    int counter = 0; // A counter to count through the patterns array
    while(1) {
        delay_milliseconds(delay);        // Wait
        delay += 1;                       // Gradually increase the delay
        p32 <: patterns[counter];         // Drive the next led pattern
        counter++;                     // Pick the next pattern
        if (counter == NUM_PATTERNS) {       // If we are at the last pattern
            counter = 0;              // then wrap around.
        }
    }
    return 0;
}
```

Looking at this in a more detail you can see the following:

- The <: operator is a port output. So p32 <: patterns[counter] will drive a pattern onto the port.
- The delay_milliseconds function is defined in timer.h and pauses for the specified number of milliseconds.

APPENDIX A - Launching the demo application

Once the demo example has been built either from the command line using xmake or via the build mechanism of xTIMEcomposer studio we can execute the application on the startKIT.

Once built there will be a `bin` directory within the project which contains the binary for the xCORE device. The xCORE binary has a XMOS standard .xe extension.

A.1 Launching from the command line

From the command line we use the `xrun` tool to download code to both the xCORE devices. If we change into the bin directory of the project we can execute the code on the xCORE microcontroller as follows:

```
> xrun AN00175_startKIT_LED_demo.xe          <-- Download and execute the xCORE code
```

Once this command has executed the application will be running on the startKIT and the LEDs should flash.

A.2 Launching from xTIMEcomposer Studio

From xTIMEcomposer Studio we use the run mechanism to download code to xCORE device. Select the xCORE binary from the bin directory, right click and go to Run `Configurations`. Double click on xCORE application to create a new run configuration and then select Run.

Once this command has executed the application will be running on the startKIT. and the LEDs should flash.

APPENDIX B - References

XMOS Tools User Guide

http://www.xmos.com/published/xtimecomposer-user-guide

XMOS xCORE Programming Guide

http://www.xmos.com/published/xmos-programming-guide

APPENDIX C - Full source code listing

C.1 Source code for main.xc

```
// Copyright (c) 2016, XMOS Ltd, All rights reserved
#include <xs1.h>
#include <timer.h>

/* This the port where the leds reside */
port p32 = XS1_PORT_32A;

/*
 * the patterns for each bit are:
 *   0x80000 0x40000 0x20000
 *   0x01000 0x00800 0x00400
 *   0x00200 0x00100 0x00080
 *
 * As the leds go to 3V3, 0x00000 drives all 9 leds on, and 0xE1F80 drives
 * all nine leds off.
 * The four patterns below drive a dash, backslash, pipe, and slash.
 */

#define NUM_PATTERNS 4

int patterns[NUM_PATTERNS] = {
    0xE0380,
    0x61700,
    0xA1680,
    0xC1580
};

int main(void) {
    int delay = 50;      // initial delay 50 ms
    int counter = 0; // A counter to count through the patterns array
    while(1) {
        delay_milliseconds(delay);        // Wait
        delay += 1;                       // Gradually increase the delay
        p32 <: patterns[counter];         // Drive the next led pattern
        counter++;                      // Pick the next pattern
        if (counter == NUM_PATTERNS) {       // If we are at the last pattern
            counter = 0;              // then wrap around.
        }
    }
    return 0;
}
```