XK XS1-G Development System Quick Start Guide

Version 1.1

XMOS

Publication Date: 2010/05/10

1 Introduction

The XDK XS1-G Development System contains everything you need to start exploring the XMOS event-driven processor technology.

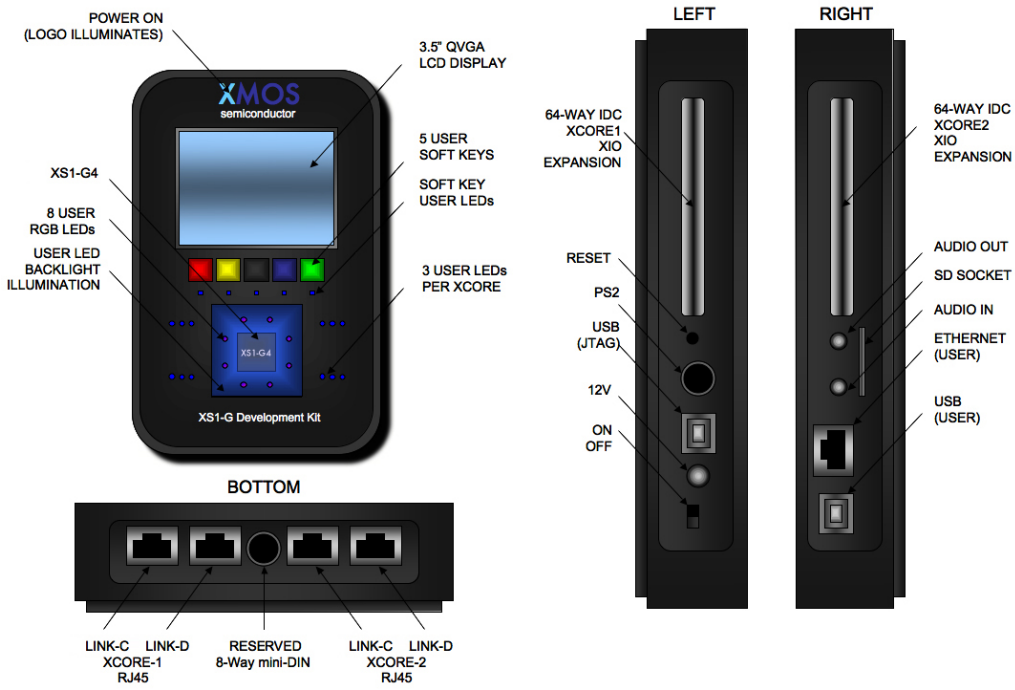
The XDK Development System includes:

- XS1-G Development Kit (XDK)—the black box with all the electronics and hardware.
- 12V power supply unit and local AC power cable.
- USB cable—connects your system to the XDK.
- XMOS Link cable (x 2)—connects the XDK to other devices using XMOS Link sockets. XMOS Link cables have the same form factor as Ethernet cables, but are crossover cables that can only be used for linking to XDK XMOS Link sockets.

2 XDK Demos

A set of demonstrations is pre-installed on the XDK which you can use to test the hardware. Additional demonstrations can be downloaded from the XMOS web site, including an XDK tutorial which introduces the key concepts you need to understand when programming the XK-1 board.

1. Plug in the XDK using the supplied power cable.
2. Slide the **ON/OFF** button (below the power socket) up to switch the XDK on and wait for the splash screen to be displayed.
3. Press the **MENU** (black/middle) button to go to the Menu page.
4. Press the **UP** (red/left) and **DOWN** (green/right) buttons to scroll through the demos.
5. Highlight the Pong demo (PONG.XB) and press the **RUN** (blue) button to play the demo.
6. Press **MENU** to go back to the Menu screen.


POWER ON
(LOGO ILLUMINATES)
3.5" QVGA
LCD DISPLAY
5 USER
SOFT KEYS
SOFT KEY
USER LEDs
3 USER LEDs
PER XCORE
XS1-G4
8 USER
RGB LEDs
USER LED
BACKLIGHT
ILLUMINATION
XMOS
semiconductor
XS1-G4
XS1-G Development Kit
LEFT
RIGHT
64-WAY IDC
XCORE1
XIO
EXPANSION
RESET
PS2
USB
(JTAG)
12V
ON
OFF
64-WAY IDC
XCORE2
XIO
EXPANSION
AUDIO OUT
SD SOCKET
AUDIO IN
ETHERNET
(USER)
USB
(USER)
BOTTOM
LINK-C
XCORE-1
RJ45
LINK-D
RESERVED
8-Way mini-DIN
LINK-C
XCORE-2
RJ45
LINK-D

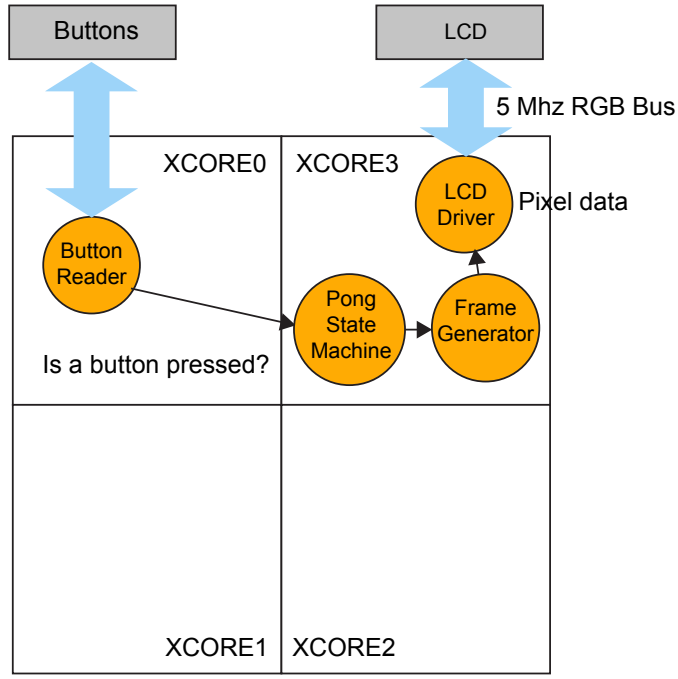
2.1 Pong (PONG)

Pong is a simple application that uses four threads. The state machine polls the button thread to see if a button is currently pressed, makes the main game calculations, updates the positions of the pads and ball and passes them to the frame generator.

The frame generator draws the raw pixel data, and passes it through a channel to the LCD driver thread to output to the screen.

The demonstration is divided across four XCores due to the device mapping on the XDK.

Use the two left buttons to move the left paddle up and down, and the two right buttons to move the right paddle.

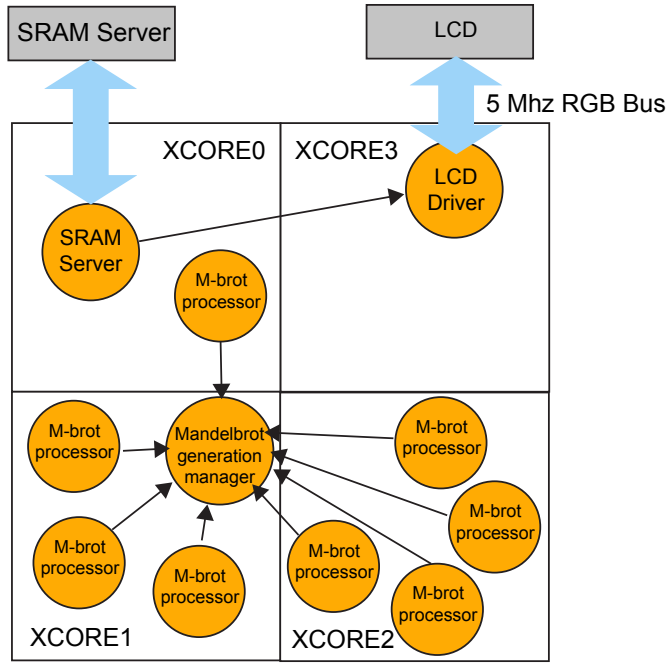


2.2 Mandelbrot (MBROT)

The Mandelbrot demonstration uses the concurrent feature of the XMOS architecture to farm out the processing to multiple threads running across the XCores.

The generation manager requests Mandelbrot calculations from eight threads running the escape time algorithm. It then passes the resulting pixel data to the SRAM server for storage.

Concurrently, the SRAM server pipes the RGB data to the LCD driver, which displays the image on the LCD.

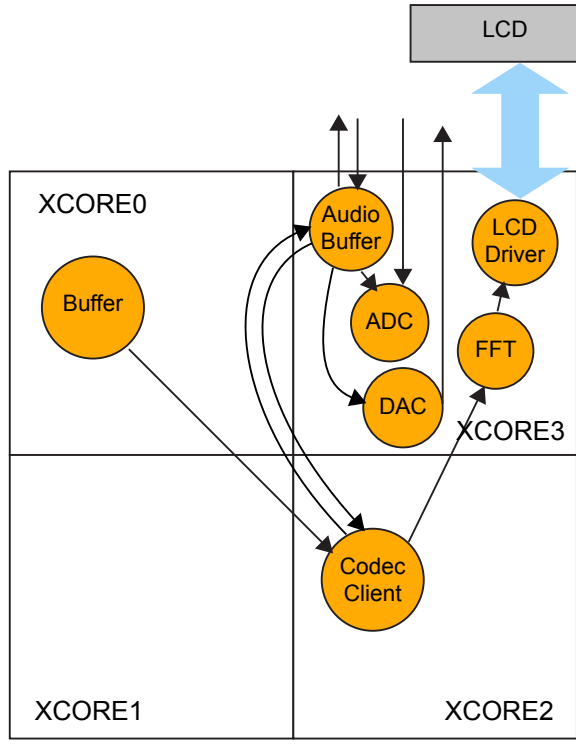


2.3 Audio Frequency Analyser (FREQ)

The FREQ program reads audio input, and visualizes the power in each frequency band in real-time on the screen. Connect a microphone or MP3 player to the line input to run the program.

The horizontal axis of the screen is time, and the vertical axis represents the frequency bands. The sound intensity is represented by colorâĂŤdark colors indicate little power in the band, and bright colors indicate high power. The power spectrum is updated in real-time using a sliding window, which represents about a second worth of audio.

Internally, the program computes a 512-sample FFT, and uses a lookup table to map the magnitude of each frequency onto a color. The FFT is computed 200 times per second with a 75% overlap—128 new samples are appended to 384 of the previous samples and the FFT is computed over these 512 samples. The real and imaginary components are squared and summed, the sine and cosine parts are then added, and the resulting value is used to lookup a color.



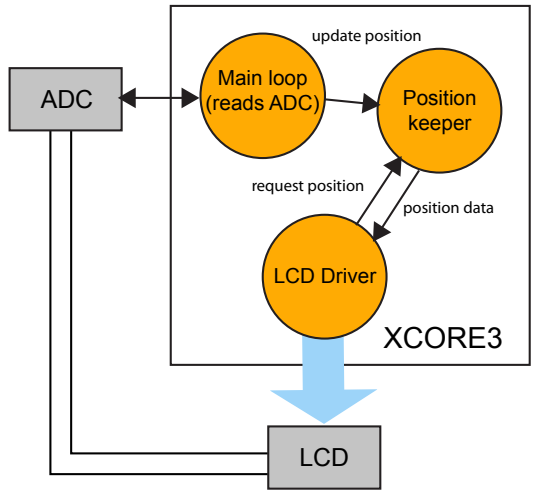
2.4 Touch Screen Finger Follow

The touch-screen demonstration has the following behavior:

When the application starts, a large red cross is displayed on the LCD screen. When a press is detected, the color of the cross is changed to green, and the cross follows the user's finger. On release, the color of the cross returns to red.

To achieve this functionality the application is broken down to three threads:

1. **Position Keeper**: Effectively this thread allows state to be safely shared between the LCD driver thread and the main program loop. This thread stores the color and position of the cross. It takes updates to this shared state from the main program loop and responds to requests for data from the LCD driver.
2. **Main program loop**: This thread is paused waiting for the PENIRQ line from the touch screen digitizer to go low, indicating the screen is pressed. When this event is fired, it sends a command (via a channel) to the *position keeper* to update the color of the cross, then reads the position on the press and issues another command to the *position keeper* to update the location of the cross. This thread also lights associated *core* LEDs dependent on the position read. Once done the color of the displayed cross is reset to red.
3. **LCD Driver**: This thread is responsible for providing the physical interface to the screen. It requests an update for the position/color of the cross once per frame. This thread uses timers to produce the appropriate control signal and horizontal/vertical blanking delays.



2.5 Menu system, loader and LCD screen (XDK)

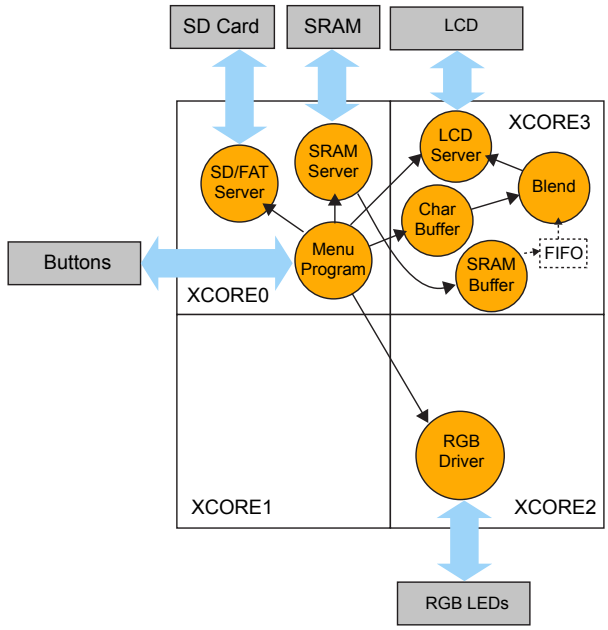
The XDK menu program is provided for users who want to change the menu for their own use. If you run XDK.XB, the XDK will appear to reset itself, as it re-runs the menu; no other behavior is expected.

The program uses SRAM, LCD, and SD/FAT16 files system software components to display the menu system.

The program uses the SD/FAT16 component to load images from the SD card into frame buffers in the SRAM.

The LCD component includes a thread, which reads frame data from the SRAM and buffers it locally. A *character buffer* thread calculates pixel values using a local character buffer. For each pixel in a frame, these two threads output data through a channel to a *blender thread* that does a simple blending operation. The *blender thread* outputs the blended pixel data to the LCD server thread, which provides the interface to the LCD.

The main menu program thread uses two channels to communicate with the LCD component, one for text and the other for graphics. The text channel adds characters to, and removes them from, the character buffer. The graphics channel is used for frame update commands and other system requirements.



3 XMOS Tools Support

The XMOS tools are provide in a single platform-specific downloadable file from

http://www.xmos.com/tools

Instructions on installing and using the XMOS Tools can be found in the XMOS Tools User Guide **http://www.xmos.com/published/xtools_en**. The following table provides a summary tools support for the XDK.

Minimum tools version required	9.7
Board support file	XDK.xn
USB-to-JTAG adaptor	Internal
USB-to-JTAG driver	FTDI
Board Tutorial	Integrated into XDE and available as PDF

4 Document History

Date	Release	Comment
2009-07-08	1.0	First release
2010-05-10	1.1	Added tools support section

Disclaimer