MAX32600 UART printf() Demonstration

April 3, 2015

1 Abstract

The MAX32600 has two standard serial UART interfaces. In this application, we will demonstrate how to use one of the UART interfaces as an ASCII text output device via the standard C library function printf().

2 Requirements

- MAX32600B EvKit
- Olimex JTAG adapter and USB cable to connect the JTAG adapter to the PC
- Sample code for this application located in `Firmware/Applications/UARTPrintfDemo`
- GNU ARM toolchain with newlib libc
- USB Full-size A to B cable
- PC or Workstation with USB and terminal emulator software

3 Setup

On the MAX32600 EvKit, the UART0 device can be configured to connect to an on-board FTDI USB to serial converter for easier connectivity to a standard PC. The onboard FTDI converter is powered from the PC, not the EvKit itself, therefore, the USB to serial connectivity is alway alive without power requirements from the EvKit.

- Connect your PC to the full-size USB-B port on the EvKit
- Depending on your operating system and drivers you may need to install the FTDI USB to serial driver.
- Open a serial terminal emulator program. On MS Windows, you may have or need to install one of several options including but not limited to; TeraTerm, PuTTY, RealTerm.
- Find the PC serial port. On MS Windows, the USB to serial converter typically enumerates itself to COM6 or higher.
- Set your terminal emulator program for the following serial options: BAUD => 115200; No parity; No flow control;
- Compile and load the application

4 Observation

Observe the text string on the terminal emulator. The code also has a character echo feature, you may type in the emulator program and the UART code will push the character back to the emulator.

5 Source Code Overview

5.1 Drivers In Use

- Instruction Cache
- Clock Manager
- Power Manager
- IO Manager
- GPIO
- SysTick
- UART
- Newlib / libc

5.2 Interrupts Enabled

- UART0

5.3 Code Operation

- Enable Instruction Cache
- Setup Clocks (trim ring oscillator)
- Set UART pin mapping
- Set UART configuration (baud rate and serial controls)
- Setup stdout buffer for libc printf()
- Register UART input handler
- Demonstrate "`printf()`" which will call the provided libc `_write()`
- Wait for interrupts; service incoming UART characters