Building NEEK Superloop Simple Socket Server Plus

Last Updated

June 12, 2009

The Superloop Simple Socket Server Plus example is a Superloop implementation of the Simple Socket Server Plus (SSS Plus) example that is also published on the Nios Wiki site. This example has all of the features and capabilities of the SSS Plus example however it is implemented with the Superloop option that is built into the InterNiche TCP/IP stack such that it does not require an RTOS for the application to run. This is a basic capability that is available in the InterNiche TCP/IP stack as it ships with the Altera Nios II development environment.

This example is built on the Software Build Tools flow as described in the Nios II Software Developers Handbook. The software project directory contains an app subdirectory, a bsp subdirectory, a superloop_iniche_lib subdirectory and a superloop_tse_lib subdirectory, each of which contain a shell script that creates the project makefiles, bsp settings, library settings, etc. To create and build the project you should refer to the “readme.txt” file located in the “app” subdirectory of the software project directory. The “readme.txt” file will provide you with the specific command line to execute to build the application.

There are substantial modifications that this example employs from the default Simple Socket Server Plus application and associated default uCOS-II BSP. These differences can be seen in the BSP configuration, BSP patches, Library configuration, Library patches and application sources.

For more information on the way that the original Simple Socket Server Plus application is built, please refer to the “building_NEEK_simple_socket_server_plus.doc” document.

BSP configuration changes

If you look at the “create-this-bsp” script for this application, you can see that we are building a “hal” based BSP which is substantially different from the “ucosii” aware BSP that the SSS Plus application requires. We do not enable the “altera_iniche” software package in the BSP since this is created as a separate library in this example, but we still enable the “altera_ro_zipfs” software package.

The rozipfs package is configured in pretty typical fashion with the rozipfs expected to be located at offset 0x002C0000 of the CFI flash, which is the high portion of reserved space for the NEEK software application storage. This means that you could store up to 3.5MB of rozipfs data in this flash area.

The sysclk and timestamp timer services are assigned to timers in pretty standard fashion.

The TSE MAC driver is disabled in the BSP to prevent the default instantiation and initialization from happening in the alt_sys_init.c file. This example provides an

externally configured library for this driver and it is then manually instantiated and initialized by the main application.

The compiler optimization level of the BSP is set to –O2.

The program and data sections for the application are all configured to run out of the DDR RAM that is provided in the hardware system.

BSP Patches

There is one subtle patch applied to the default HAL BSP that we create. The file alt_syscall.h that is used by the HAL BSP is replaced by the same header out of the ucosii BSP sources instead so that we can pick up certain driver calls that will actually be implemented in the iniche library. This subtle change is implemented by one line in the "create-this-bsp" script prior to compiling the library.

iniche Library Configuration

The "create-this-lib" script provided for the iniche library is rather long and boring. The bulk of the work that this script performs is to copy all of the source files for the InterNiche Stack out of the installation directory tree and into the local library directory. Once that is accomplished, the library patches are applied and the library is compiled.

The library is compiled with optimization level –O2.

The iniche package in this example is configured to NOT use the dhcp client for IP address acquisition. The software application statically assigns an IP address and reads the MAC address out of the NEEK eeprom, and these addresses are assumed to be used by default. If you wish to use the DHCP client service for IP address acquisition, you will need to change the library sources a bit. The best way to enable the DHCP client for this library is to enable its definition in the ipport.h file.

iniche Library Patches

The patches that are applied to the iniche library by the "create-this-lib" script are essentially the same patches that were applied in the SSS Plus example. You can get more detailed information on the patches applied to SSS Plus in the "building_NEEK_simple_socket_server_plus.doc" document.

Beyond the initial patches applied to SSS Plus, the Superloop SSS Plus example must apply additional code patches where necessary to compensate for the Superloop nature of the library, where no RTOS support requires additional implementation details to be deployed. None of the Superloop specific patches are substantial alterations to the structure of the library.

TSE Library Configuration

The "create-this-lib" script provided for the TSE library is rather brief and boring. The bulk of the work that this script performs is to copy all of the source files for the TSE driver out of the installation directory tree and into the local library directory. Once that is accomplished, the library patches are applied and the library is compiled.

The library is compiled with optimization level –O2.

TSE Library Patches

The patches that are applied to the TSE library by the “create-this-lib” script are essentially the same patches that were applied in the SSS Plus example. You can get more detailed information on the patches applied to SSS Plus in the “building_NEEK_simple_socket_server_plus.doc” document.

Beyond the initial patches applied to SSS Plus, the Superloop SSS Plus example must apply additional code patches where necessary to compensate for the Superloop nature of the library, where no RTOS support requires additional implementation details to be deployed. None of the Superloop specific patches are substantial alterations to the structure of the library.

Application Sources

Please refer to the “create-this-app” script for details of the specific makefile settings that are directed by that script when generating the application makefiles. The application source files for Superloop Simple Socket Server Plus are derived from the original Simple Socket Server Plus application; however these substantial changes have been applied:

network_utilities.c – just as in SSS Plus, the get_mac_addr() and get_ip_addr() functions that are required to provide implementation specific results when called by the InterNiche TCP/IP stack have been coded to return a hard coded static IP address and the NEEK MAC address programmed into the eeprom of the NEEK board. All of the flash manipulation code that was part of the original SSS implementation has been commented out. For users who wish to make a deployable implementation out of this example, attention will need to be paid here and some deployable implementation of get_mac_addr() and get_ip_addr() will need to be implemented here.

simple_socket_server.h – just as in SSS Plus, this file contains the hard coded IP address settings.

demo_control.c – just as in SSS Plus, this file has been added to the project to provide the additional commands that are accessed thru the InterNiche menu system.

led.c – has been removed and all of its functionality is unavailable in the Superloop SSS Plus application

alt_error_handler.c – has been removed. The functionality provided by this file was primarily in support of the ucosii environment which is not provided in Superloop SSS Plus.

simple_socket_server.c – has been substantially modified to remove the ucosii specific task functionality of the original SSS Plus implementation. This routine does still provide the main loop for the superloop however.

iniche_init.c – has been substantially modified. All of the multi tasking support has been removed. The TSE driver instantiation and initialization is provided in this file since we are linking in the new customized TSE driver for Superloop mode. The tk_yield() routine that the InterNiche stack requires is implemented here and the SignalPktDemux() routine is stubbed out in this file. There is an Altera HAL alarm started to provide the ctick functionality required by the InterNiche Stack, since this was originally provided as a ucosii service in SSS Plus. The way that the InterNiche Stack is initialized and started has been modified slightly, however, we still fall into the remnants of the SSStask just like the original SSS Plus example.

neek_support – the sources in this directory have been added to the project to provide access to the LCD display on the NEEK board.

lrozipfs.zip and srozipfs.zip – just as in SSS Plus, these files are provided with the project sources as example ROZIPFS file systems that can be programmed into flash using this application. The lrozipfs.zip file provides a lager multi level hierarchy directory structure in it's file system, whereas the srozipfs.zip file provides a simple small flat directory structure in it's file system.