Introduction

The application selector is a useful utility that lets you load and run different applications stored in the SD Card, either through the LCD screen or via the integrated web server interface.

This document is intended to provide more details on the software design of the application selector.

Basic software structure

The Application Selector uses the MicroC/OS-II real-time operating system to provide multi-tasking and inter-task communication services and NicheStack TCIP/IP Stack to provide networking services to the application layer. Figure 1 shows the software layers implemented in the software structure of Application Selector.

Figure 1: Software Layers

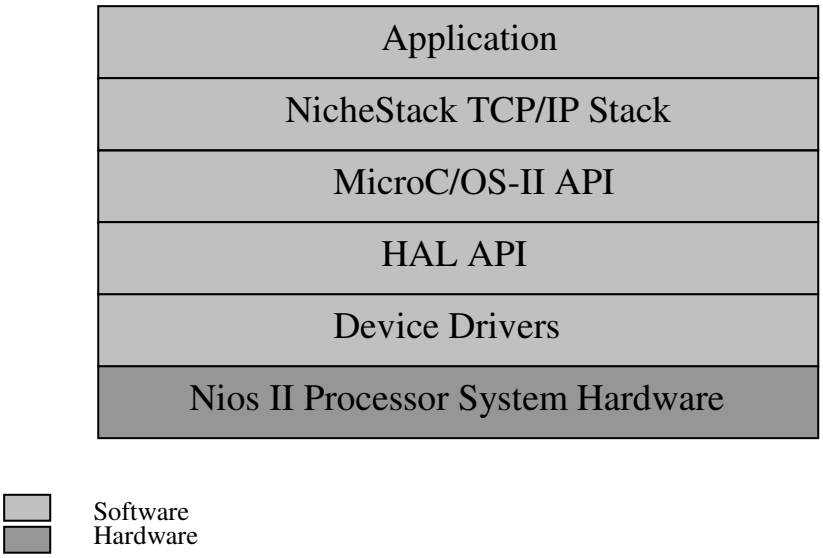
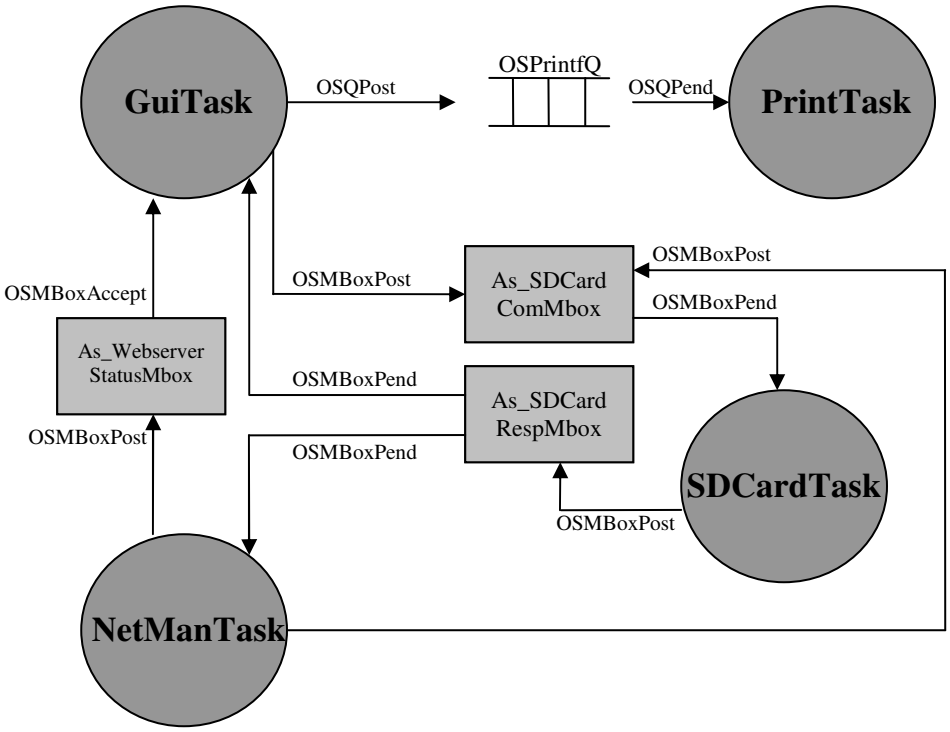


Figure 2: Tasks and resources



MicroC/OS-II Inter-Task Resources

The following resources are created to handle communication between OS tasks (refer Figure 2).

- OSPrintfQ

OSPrintfQ is a MicroC/OS-II message queue used to send messages to PrintTask() to print messages to STDOUT.

- As_WebserverStatusMbox

As_WebserverStatusMbox is the handle to the MicroC/OS-II Application Selector Web Server Status message box. The message box is used to send the web server status to be displayed on the LCD.

- As_SDCardComMbox

As_SDCardComMbox is the handle to the MicroC/OS-II Application Selector SD Card Com message box. The message box is used by SD Card functions to send commands to SDCardTask().

- As_SDCardRespMbox

As_SDCardRespMbox is the handle to the MicroC/OS-II Application Selector SD Card Response message box. The message box is used by SD Card functions to receive response from SDCardTask().

MicroC/OS-II tasks

MicroC/OS-II supports multi-tasking and this section describes the tasks used in the application selector (refer Figure 2).

Task	**Description**
ASInitialTask()	An initial task, this task initializes the NicheStack TCP/IP Stack, create operating system data structures and application tasks. Once this task is done, this task deletes itself.
GuiTask()	Draws the main menu of the application selector, senses input from the touch screen and takes the appropriate actions.
SDCardTask()	Receives and executes commands via As_SDCardComMBox and responds via As_SDCardRespMbox.
PrintTask()	Print messages to stdout by getting messages from OSPrintfQ.
LedTask()	Blinks the LED continuously.
NetManTask()	Manage and maintain the network connection and monitors the status of a single network PHY.
WSTask()	Manages the socket server connection.

Software Implementation Details

This section provides details about the Application Selector tasks and functions.

main() in main.c

- Calls OSTimeSet() to clear the OS Timer
- Calls ASInitialTask() via OSTaskCreateExt()
- Calls alt_uCOSIIErrorHandler() to check system call error codes
- Calls OSStart() to begin multithreading

The ASInitialTask() does the following:

- Creates data structures for the applications, including As_WebserverStatusMbox
- Creates non-NicheStack TCP/IP Stack dependent task, including GuiTask(), SDCardTask(), PrintfTask() and LedTask()
- Calls NETInit() which will create NetManTask()
- Calls OSTaskDel() to delete itself as a task because it is no longer needed

The GuiTask() does the following :

- Initializes the LCD Display by setting up the memory usage and display SG-DMA with the following function :

```
alt_video_display_init( ALT_VIDEO_DISPLAY_SGDMA_NAME,        // char* sgdma_name
                        ALT_VIDEO_DISPLAY_COLS,              // int width
                        ALT_VIDEO_DISPLAY_ROWS,              // int height
                        ALT_VIDEO_DISPLAY_COLOR_DEPTH,       // int color_depth
                        ALT_VIDEO_DISPLAY_USE_HEAP,          // int buffer_location
                        ALT_VIDEO_DISPLAY_USE_HEAP,          // int descriptor_location
                        2 );                                 // int num_buffers
```

- Initializes the LCD and touch screen peripherals and calibration.
- Gets the list of available applications from SD Card.
- Displays the Main Menu with the list of available applications.
- Checks for touch screen activity continuously and, if needed, executes the required actions such as loading an application from the SD Card or updating the touch screen display items.
- If there is no touch screen activity, will display a screen saver after time out.
- Get and display web server status via As_WebserverStatusMbox.

The SDCardTask() does the following :

- Creates message boxes As_SDCardComMbox and As_SDCardRespMbox.
- Checks the message box, As_SDCardComMbox for any of these SD Card commands: SDCARD_READ, SDCARD_WRITE, SDCARD_OPEN, SDCARD_CLOSE, SDCARD_FILELENGTH, SDCARD_LISTING, SDCARD_ERROR and execute the required actions.
- Post respond messages in As_SDCardRespMbox if required.

PrintfTask() does the following:

- Creates message queue OSPrintfQ with the following function :
  OSPrintfQ = OSQCreate(&OSPrintfMsg[0], 50)
- Up to 50 messages can be stored in the message queue.
- Continuously waits on OSPrintfQ for messages posted by the GUITask.
- Prints the messages to STDOUT.

LedTask() does the following:

- Toggle the LED on and off.
- Send the output to the LED IO port with the following macro :
  IOWR(LED_PIO_BASE, 0, led_value)
- Insert a delay with OSTimeDlyHMSM() to slow down the LED toggles.

NetManTask() does the following:

- Calls wait_on_phy() for PHY reset and detection.
- Calls net_init(), which calls alt_iniche_init() and netmain() to initialize NicheStack TCP/IP networking stack, and creates WSTask (via TK_NEWTASK) to start the networking task.
- Checks status of the network PHY continuously, and if disconnected, calls wait_on_phy() to reset and detect PHY again, and forces re-acquisition of the existing IP address.
- Posts web server status to GuiTask() via As_WebserverStatusMbox.

WSTask() does the following :

- Creates a socket to serve a TCP/IP connection, binds to the socket and listens for TCP/IP connection requests from a client.
- Calls http_handle_accept() for incoming TCP/IP connection.
- Calls http_handle_receive() and http_handle_transmit() to serve the TCP/IP connection.

Remote System Update

An important feature of Application Selector is the Remote System Update which allows you to update your system with a new FPGA image when the kit is connected to the network. From any PC, view the Board Update Portal web page by typing the kit's IP Address on any web browser. By following the instructions on the web page, load a design stored on the local PC to program it to the flash memory on the board. You can then reset the FPGA to reconfigure from the newly downloaded flash image.

Remote System Update Software Implementation Details

Once a network connection is established, the WSTask will constantly monitor the TCP/IP connection. Any incoming requests are handled by http_handle_receive().

When the Upload button in Board Update Portal web page is pressed, HTTP will send POST command with "/reset_system.html" as the next web page to be loaded. In http_handle_post(), if "/reset_system.html" is received, memory will be allocated for the upload buffer, and conn->file_upload flag will be set to 1. This will cause upload_field.func() to call function file_upload().

In file_upload(), the function lookup_flash_offset() will check whether a hardware or software flash file is being uploaded and set the flash offset accordingly. The flash offset for hardware flash file is USER_HW_IMAGE_OFFSET and for software flash file is USER_SW_IMAGE_OFFSET. In process_uploaded_data(), the data from upload buffer will be written into flash memory.

When the Reset System button in Board Update Portal web page is pressed, HTTP will send POST command with "/RESET_SYSTEM". In http_handle_post(), if "/RESET_SYSTEM" is received, reset_field.func() will call reconfig_fpga(). In reconfig_fpga(), AsReconfigFPGA() will be called. In AsReconfigFPGA(), the ALT_REMOTE_UPDATE megafunction is used set reconfigure from offset address in flash, RECONFIG_ADDRESS.

Task priorities

MicroC/OS-II operating system requires all tasks running on the system to be assigned a unique priority number. Tasks assigned with lower priority numbers are treated as higher priority tasks.

Task priorities	Task type
#define TK_NETMAIN_TPRIO 2 *#define TK_NETTICK_TPRIO 3*	NicheStack TCP/IP Stack internal tasks (defined in ipport.h)
#define AS_INITIAL_TASK_PRIO 0 *#define AS_SDCARD_PRIO 6* *#define AS_LED_PRIO 16* *#define AS_PRINTF_PRIO 5* *#define AS_GUI_PRIO 7* *#define AS_NETMAN_PRIO 10* *#define AS_HTTP_PRIO 11*	Application Selector tasks (defined in os_utils.h)

AS_INITIAL_TASK_PRIO is assigned the highest priority value of 0 because this is the first task that MicroC/OS-II runs to create other tasks and resources. Priority for the NicheStack TCP/IP Stack task, TK_NETMAIN_TPRIO, is set to 2, because this task should be higher priority than other application tasks that use the NicheStack TCP/IP Stack. Altera recommends that the time-keeping task for NicheStack TCP/IP Stack, TK_NETTICK_TPRIO to be set to one priority level lower than TK_NETMAIN_TPRIO.

For Application Selector tasks, AS_PRINTF_PRIO is given higher priority to push data out to STDOUT as soon as possible. AS_LED_PRIO is given the lowest priority because this is a non-critical task. Priorities for other Application Selector tasks can be assigned accordingly between AS_PRINTF_PRIO and AS_LED_PRIO.

Application Selector Files

The following list the main files in Application Selector.

Filename/Folder	Description
main.c	Contains functions to creates OS data structures and OS tasks for the application selector
app_selector_gui.c app_selector_gui.h	Contains functions to draw the graphic display, send the display data to the touchscreen , sense touchscreen input and take the appropriate actions according to touchscreen input
app_selector.c app_selector.h	Contains functions to load the hardware and software of an application from the SD Card into flash, then reconfigures the FPGA to run the application
webserver.c webserver.h	Contains functions to initialize and run the NicheStack, including getting the IP address and maintaining the network connection
sd_card.c sd_card.h	Contains functions to perform tasks related to the SD Card
os_utils.c os_utils.h	Contains functions for printf task and MicroC-OS/II error handlers. os_utils.h contains definition for OS task stack size and task priority
alt_touchscreen	Contains the touch screen software API
alt_tpo_lcd	Contains LCD software API
alt_video_display	Contains the video pipeline software API
alt_eeprom	Contains functions that interface with the EEPROM
gimp_bmp	Contains tables for gimp image
graphic_lib	Contains graphic functions
images	Contains tables for image data
srec	Contains S-Record routines

Referenced Documents

- Using the NicheStack TCP/IP Stack (http://www.altera.com/literature/tt/tt_nios2_tcpip.pdf)
- Implementing an LCD Controller (http://www.altera.com/literature/an/an527.pdf)