MP3 Player

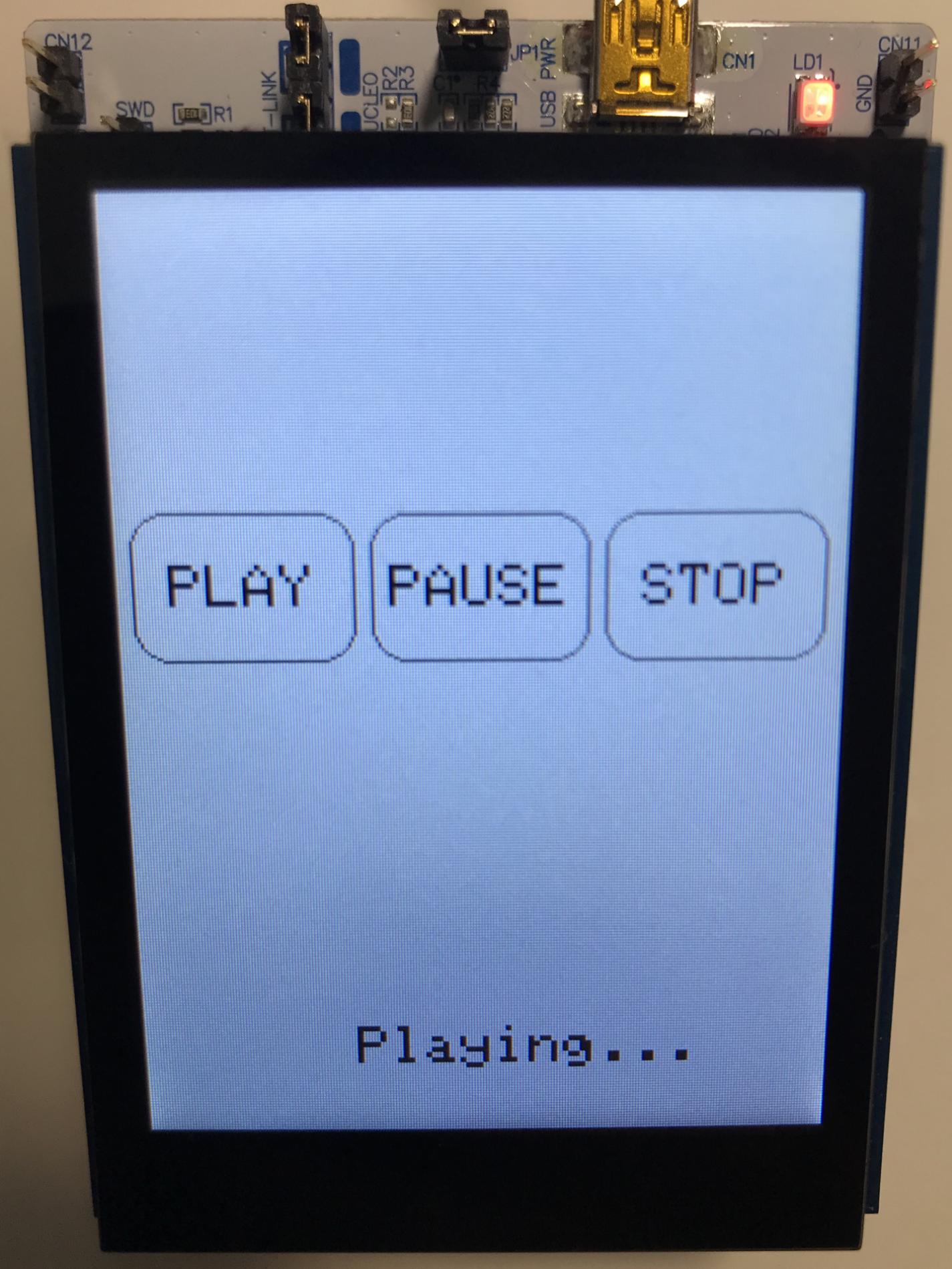
Igor Grinchenko

I. INTRODUCTION

This report describes my implementation of MP3 player using STM32F401 microcontroller board, Adafruit VS1053 Music Maker, and Adafruit ILI9341 touchscreen. This letter is organized as follows. Section II describes player features and user instructions. Section III describers player design and implementation.

II. FEATURES AND USER INSTRUCTIONS

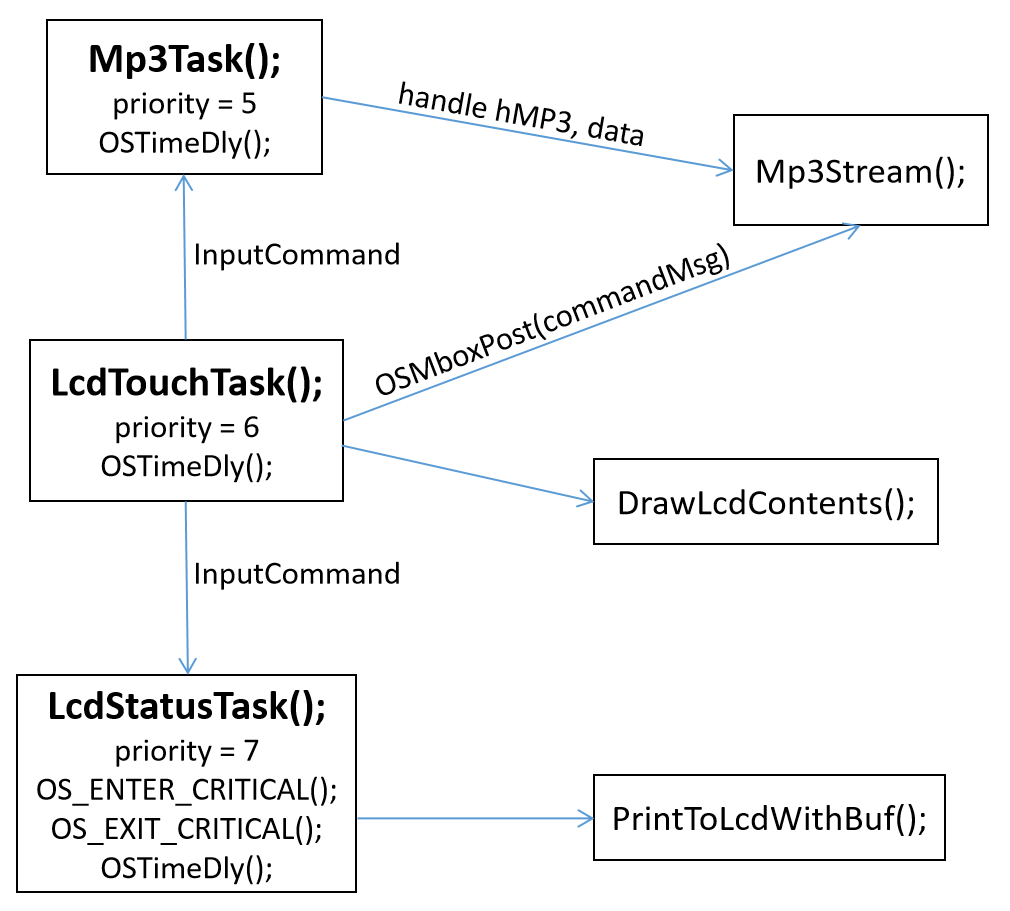
My MP3 player has 3 buttons - Play, Pause, and Stop (See Figure 1). If the user selects Play button, the player will indefinitely play “Popcorn” music track until Pause or Stop button is selected. Selecting Pause button would pause music until user selects Play button. Selecting Stop button will stop music until user selects Play button to play music from the beginning. There is also player status shown on the bottom of the screen.



**Figure 1.** User Interface Design

III. DESIGN AND IMPLEMENTATION

This MP3 player is implemented using MicroC OS-II Real Time Operating System (RTOS). The player design consists of 3 tasks - *MP3Task*, *LcdTouchTask*, and *LcdStatusTask* (See Figure 2). The *MP3Task* has priority set to 5, *LcdTouchTask* priority set to 6, and *LcdStatusTask* is set to lowest priority of 7.



**Figure 2.** Function Block Diagram

The *LcdTouchTask* initially initializes the LCD display driver over Serial Peripheral Interface (SPI) and LCD touch driver over I2C using PJ Driver Framework. This task also makes a call to *DrawLcdContents* function which creates buttons on the screen using functions provided inside *Adafruit\_GFX.cpp* file. The *LcdTouchTask* does polling when it has CPU time to check if any button has been selected. If any button is selected, this task updates global enumeration *InputCommand* and does a post to the RTOS mailbox, *commandMsg,* for *Mp3Stream* function.

The *LcdStatusTask* checks if the global enumeration *InputCommand* has changed, and if so, this task updates player status on the screen using *PrintToLcdWithBuf* function. Since *LcdStatusTask* has the lowest priority, I utilize OS\_ENTER\_CRITICAL and OS\_EXIT\_CRITICAL RTOS macros to disable processor interrupts when the player status is being updated.

The *MP3Task* has the highest priority and initially initializes MP3 driver over SPI using PJ Driver Framework. This task uses global enumeration *InputCommand* to check user input command and if the command is set to INPUTCOMMAND\_PLAY this task makes a call to *Mp3Stream* function which is provided inside *mp3Util.c* file. If the enumeration *InputCommand* is set to anything but INPUTCOMMAND\_PLAY, this task keeps calling *OSTimeDly* function (provided by RTOS) to give CPU time to other tasks.

I have modified the *Mp3Stream* function to use RTOS mailbox, *commandMsg,* to receive mail from *LCDTouchTask*. As I mentioned above, whenever user selects a button, *LcdTouchTask* posts user selected command to the mailbox. Whenever *Mp3Stream* function is executing, it constantly checks for user command change and if the command is not INPUTCOMMAND\_PLAY or INPUTCOMMAND\_PAUSE this function returns to the caller. If the user selected command is INPUTCOMMAND\_PAUSE, the *Mp3Stream* function continuously calls the *OSTimeDly* function inside of it.

My MP3 player implementation has the following memory footprint:

* ROM code - 18.5KB
* ROM data - 202.8KB
* RAM data - 9.5KB

REFERENCES

[1] - Labrosse, Jean J. *MicroC/OS-II: the Real-Time Kernel*. Second ed., CMP Books, 2002.