```
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include "altera_avalon_timer.h"
#include "altera_avalon_timer_regs.h"
#include <altera_up_sd_card_avalon_interface.h>

// Necessary variables //
int songMins[] = {1, 0, 2, 3}; // Change this and the one below for testing
int songSecs[] = {18, 36, 23, 2};
int tracknumber = 0;
int timertrack = 0;
int totalSecs;
int partitions;
int scrub = 0;
int progress = 0;

// Pushbutton variables //
volatile int edge_capture;
int paused = 0;

int Delay() //delay used for next and prev song 1 second delay
{
for (int delay; delay<200000; delay++);
return 0;
}

void NewSong(void)
{
   alt_up_sd_card_dev *device = NULL;
   int connected = 0;

   device = alt_up_sd_card_open_dev(SDCARD_NAME);
   if (device != NULL)
   {
   printf("Initialized. Waiting for SD card...\n");
   while(1)
   {
           if ((connected == 0) && (alt_up_sd_card_is_Present()))
           {
                   printf("Card connected.\n");
                   if (alt_up_sd_card_is_FAT16())
                   {
                   printf("FAT16 file system detected.\n");
```

```
            printf("Looking for first file.\n");
            char * firstFile = "filenameunchanged";
            alt_up_sd_card_find_first(".", firstFile);
            printf("Volume Name: '%s'\n\n", firstFile);

            short file;
            while((file = alt_up_sd_card_find_next(firstFile)) != -1)
            {
                int contentCount = 0;
                printf("===========================\n");
                printf("Found file: '%s'\n", firstFile);

                short fileHandle = alt_up_sd_card_fopen(firstFile,false);
                printf("File handle: %i\n", fileHandle);

                printf("Contents:\n");
                short int readCharacter;

                while ((readCharacter = alt_up_sd_card_read(fileHandle)) != -1)
                {
                    printf("%c", readCharacter);
                    ++contentCount;
                }

                printf("\nContent size: %i", contentCount);
                printf("\n===========================\n\n");
            }
            }
            else
            {
            printf("Unknown file system.\n");
            }

            connected = 1;
        }
        else if ((connected == 1) && (alt_up_sd_card_is_Present() == false))
        {
            printf("Card disconnected.\n");
            connected = 0;
        }
    }
    }
    else
    {
    printf("Initialization failed.\n");
    }
return;
}

int main()
{
        volatile int *edge_capture_ptr = (volatile int*) edge_capture;

        // Song Run Time Variables //
        int
min[]={2139062080,2139062137,2139062052,2139062064,2139062041,2139062034};
//minutes 0-5

        int
sec[]={1077968767,1081704319,1076133759,1076920191,1075412863,1074954111,10739055
35,

1081638783,1073774463,1074823039,2034270079,2038005631,2032435071,2033221503,

2031714175,2031255423,2030206847,2037940095,2030075775,2031124351,608206719,
        611942271,606371711,607158143,605650815,
605192063,604143487,611876735,604012415,

605060991,809533311,813268863,807698303,808484735,806977407,806518655,805470079,

813203327,805339007,806387583,423657343,427392895,421822335,422608767,421101439,

420642687,419594111,427327359,419463039,420511615,306216831,309952383,304381823,

305168255,303660927,303202175,302153599,309886847,302022527,303071103,75530111};
//seconds 00-60
        int a = 0;
        int b = 0;

        // Shut off 7 segment displays to start //
        IOWR_ALTERA_AVALON_PIO_DATA(HEX1_BASE, 2139062143);
        IOWR_ALTERA_AVALON_PIO_DATA(HEX2_BASE, 2139062143);
```

```
// Timer configuration //
        IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_BASE, 0b1000); // Initial stop
        IOWR_ALTERA_AVALON_TIMER_PERIODH(TIMER_BASE, 0x02FA); // Top half of
50,000,000
        IOWR_ALTERA_AVALON_TIMER_PERIODL(TIMER_BASE, 0xF080); // Bottom half of
50,000,000
        IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_BASE, 0b0110); // Start,
Continuous


        // May want to export the following code to a function //



        // Loop to continuously check if a partition is passed //
        int TOBit;
        while(1) // Change this condition later to something while unpaused
        {
        totalSecs = (songMins[tracknumber] * 60) + songSecs[tracknumber];
        partitions = totalSecs / 18;
        *edge_capture_ptr = IORD_ALTERA_AVALON_PIO_EDGE_CAP(KEYS_BASE) &
0b1111;
        if (*edge_capture_ptr == 0b0000) // Default; assuming a song is playing
        {
        TOBit = IORD_ALTERA_AVALON_TIMER_STATUS(TIMER_BASE) & 0b0001;
        if (TOBit == 0b0001) // Meaning a whole second has passed
        {

                scrub++;
                if (scrub < totalSecs) // As long as the timer is still within the song playing
                {
                        if (scrub >= partitions) // If a threshold has passed and an LED needs to
light up.
                        {
                                        scrub = 0; // Resets scrub so as to keep partitions static
                                        progress = (progress * 2) + 1; // Shifts left, then keeps the
previous LEDs lit
                                        IOWR_ALTERA_AVALON_PIO_DATA(LEDR_BASE,
progress); // Updates LEDs
                        }
                        //IOWR_ALTERA_AVALON_TIMER_STATUS(TIMER_BASE, 0b00); //
Resets TO bit to continue operations; IMPORTANT
                }
                else // When a song has completely elapsed
                {
                        break; // Exit loop, change later
                        // Will probably want to reset all the variables used after the song is
finished
                }


                // Song runtime code //
                if (sec[a] == 75530111) // If sec=60 increase minutes and reset seconds
                {
                        a=0; //reset second mask counter
                        b++; //increase minute mask counter
                        timertrack--;
                        IOWR_ALTERA_AVALON_PIO_DATA(HEX1_BASE, sec[a]);
                        IOWR_ALTERA_AVALON_PIO_DATA(HEX2_BASE, min[b]);
                }
                else if(timertrack > totalSecs)
                {
                IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_BASE, 0x8);
                }
                else //seconds is not yet equal to 60, continue counting
                {
                        IOWR_ALTERA_AVALON_PIO_DATA(HEX1_BASE, sec[a]);
                        IOWR_ALTERA_AVALON_PIO_DATA(HEX2_BASE, min[b]);
                        a++; //increase second mask counter
                }
                // end of Song runtime code //


                IOWR_ALTERA_AVALON_TIMER_STATUS(TIMER_BASE, 0b00); // Resets TO
bit to continue operations; IMPORTANT
                timertrack++;
        }
        }
        else if (*edge_capture_ptr == 0b0100) // Assuming the Play/Pause button is KEY2
        // Add Play functionality later
        {
        if (paused == 0) // If not paused, pauses
        {
                paused = 1;

                // Stops Timer, thereby stopping normal operations but not the PC from checking
Pushbuttons again
                IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_BASE, 0b1000);
```

```
			IOWR_ALTERA_AVALON_TIMER_STATUS(TIMER_BASE, 0b00);

			// !!! Insert whatever other code needs to execute and pause here !!! //

			IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEYS_BASE, 0x00); // Resets the
Pushbutton context; IMPORTANT
		}
		else // If paused, plays
		{
			paused = 0;

			IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_BASE, 0b0110);

			// !!! Insert whatever other code needs to execute and play here !!! //

			IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEYS_BASE, 0x00); // Resets the
Pushbutton context; IMPORTANT
		}
		}
		else if (*edge_capture_ptr == 0b1000) // Previous song KEY3
			// Add Play functionality later
		{
		int WriteLED = 0;
		progress = 0;
		scrub = 0;
		progress = 0;
		a = 0;
		b = 0;
		timertrack = 0;
		if(tracknumber == 0)
		{tracknumber = 3;}
		else
			{tracknumber--;}

		// Shut off 7 segment displays to start //
		IOWR_ALTERA_AVALON_PIO_DATA(HEX1_BASE, 2139062143);
		IOWR_ALTERA_AVALON_PIO_DATA(HEX2_BASE, 2139062143);

		//Play previous song code
		NewSong();

		IOWR_ALTERA_AVALON_PIO_DATA(LEDR_BASE,WriteLED); // Shut off LEDR
		IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_BASE, 0b0110); // Start,
Continuous
```

```
		IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEYS_BASE, 0x00); // Resets the
Pushbutton context; IMPORTANT
		}

		else if (*edge_capture_ptr == 0b0010) // Next song KEY1
			// Add Play functionality later
		{
		int WriteLED = 0;
		progress = 0;
		scrub = 0;
		progress = 0;
		a = 0;
		b = 0;
		timertrack = 0;
		if(tracknumber == 3)
		{tracknumber = 0;}
		else
		{tracknumber++;}

		// Shut off 7 segment displays to start //
		IOWR_ALTERA_AVALON_PIO_DATA(HEX1_BASE, 2139062143);
		IOWR_ALTERA_AVALON_PIO_DATA(HEX2_BASE, 2139062143);
		IOWR_ALTERA_AVALON_PIO_DATA(LEDR_BASE,WriteLED); // Shut off LEDR
		IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_BASE, 0b0110); // Start,
Continuous
		//Play NEXT song code
		NewSong();

		IOWR_ALTERA_AVALON_PIO_EDGE_CAP(KEYS_BASE, 0x00); // Resets the
Pushbutton context; IMPORTANT
		}
  }
		return 0;
}
```