## Contents

## Introduction

This paper describes a mini digital photo album project which loads bitmap images (BMP) from an SD card inserted into the SD card reader module. It reads the data in a contiguous manner and as such the card needs to be set up with the image files prior to reading them off. The microcontroller displays the images on the SD card on a 128x128 graphic lcd. It also employs a slideshow function that triggers the system to read the next image in a preset amount of time and a power button that puts the system in a low power state.

## Implementation

The project was developed using the LPC2148 microcontroller and PG-128x128-A lcd screen. Simulation was performed using proteus 8.1 software and code was written and compiled in embedded c using keil uvision 4.0 written below is a sort of documentation describing the components of the system and the source code.

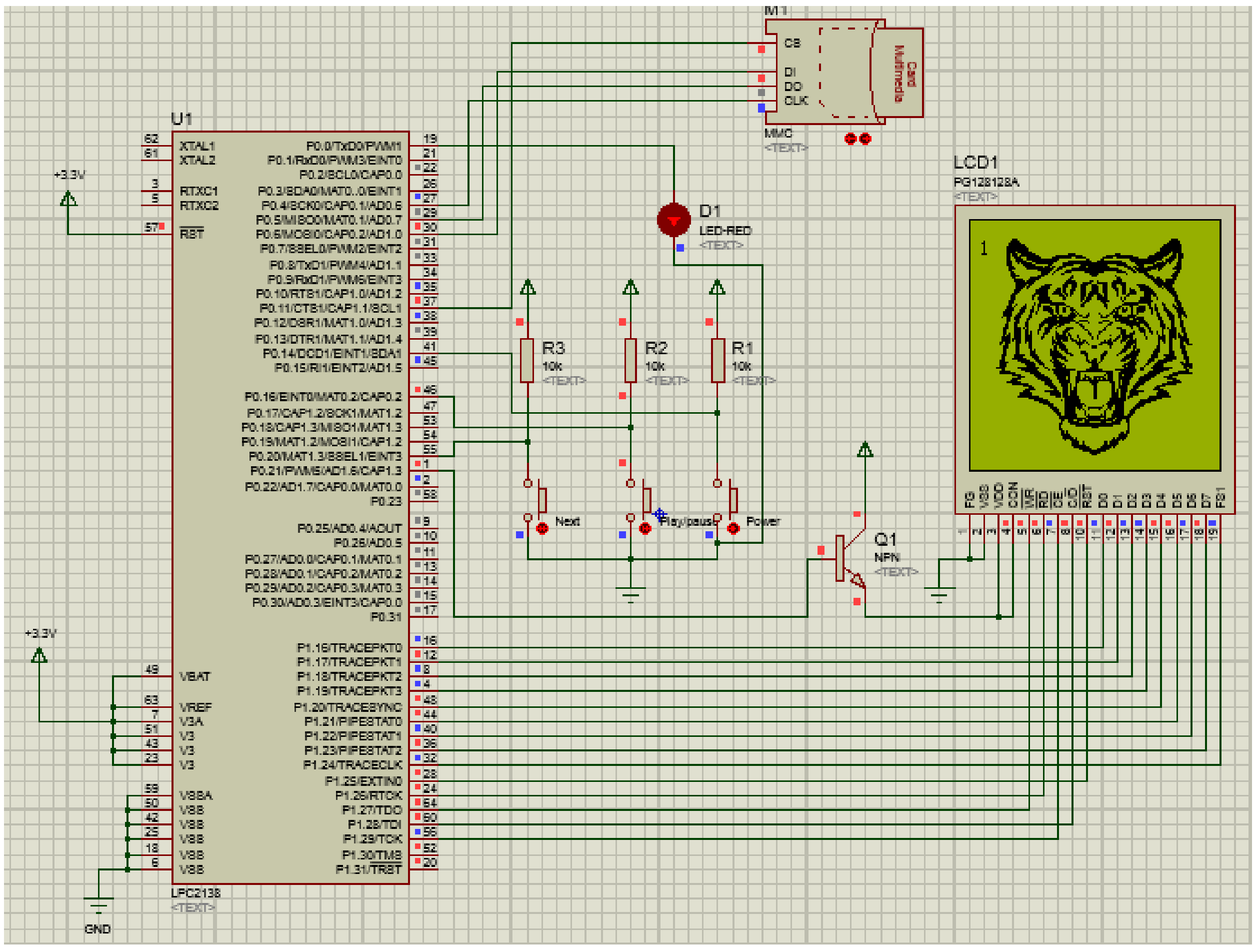


Figure 1 Proteus simulation

## Components

### Power LED

The LED lights up to signify that the system is active and functional if system is powered down the led turns off.

### Buttons

There are three buttons implemented each triggering their own interrupts the Play/Pause button as is obvious by its name controls whether or not the slide show is running. The Next button opens the next image on the memory card. The power button turns off all peripherals and powers down screen. The buttons were selected so that the power button is the button on p0.14 and the others are on the joystick so as to make it easier to test directly on the kit.

### SD reader

The SPI mode is an alternative operating mode that is defined to use the MMC/SDCs without a native host interface [More details in appendix]. The memory card is initialized with hex arrays consisting of 128x128 sized bitmaps meaning one line is 128/8 = 16 bytes long and total image is 128 lines or 2Kb the SD card operates by dividing its space to sectors sized 512 bytes, so every picture takes up 4 sectors. The SD reader functions by reading an image in to a char array from the memory card given starting sector address.

### Display unit

The display unit in this case is a graphic 128x128 LCD screen its power supply or Vdd is driven by the microcontroller an interface has been implemented to display images and text on the lcd.

### Slide show

The slide show is implemented with a timer interrupt that triggers the system to display next image every 5 seconds.

## Performance

The time the system takes to react to button press, retrieve and display the next image is less than 0.2 sec. The power down button turns off the lcd and puts the microcontroller in power down mode. According to the manual in power down mode the microchip typically consumes around 2uW meaning the system can stay in this state for more than two years on a standard 1.5V 'AA' size battery!

## Documentation

### System.c

PLL (Phase Locked Loop) - Is an electronic circuit consisting of Current controlled oscillator(CCO), phase detector and frequency multiplier. The task of the PLL is to generate the required clock for the CPU from the crystal oscillator base frequency which in this case is 12MHz.

void clock_init(void) – initializes the system clock to count at 48 MHz

void feedSeq(void) - The byte sequence (0xAA-0x55) is required to change the PLL config. to protect it from changing inadvertently by spurious system activity or external processes.

void setupPLL0(void) – sets up system to operate at frequency 4 times faster than the crystal oscillator.

void connectPLL0(void) - checks whether PLL has locked on to the  desired frequency.

### Timer.c

Timer0 is set up with a vectored irq on slot3. Prescale is set to 48000 , at system time of 48MHz the prescale means that the counter increments every milli second, the match register is set to 5000 meaning an interrupt is triggered every 5000 milli seconds.

void timer_init(void) – initializes timer and installs it as an interrupt

__irq void T0ISR(void) – the interrupt routine executed upon interrupt, it flips the next_pending flag

### Buttons.c

Pins P0.14, P0.16 and P0.20 are set up as external interrupts also the global flags ‘next_pending’, ‘power’ and ‘slide_show’ are initiated here to be used in all other files.

void button_init(void) – initializes pins as external interrupts and install them in vectored slots on the interrupt vector slot. Power button is set up to wake the micro controller from power down mode

__irq void next_button_ISR (void) – turns on next_pending flag and if slide_show is active it also resets the timer

__irq void play_button_ISR (void) – toggles slide_show and depending on the value of slide show it sets the timer to either reset or start

__irq void power_button_ISR (void) – toggles power, if powering off it disables all peripherals and puts the microcontroller in power down mode. If powering on it enables SPI and timer0 and resumes regular operations.

## Lcd.c

The PG-128x128-A lcd has the following available pin connections

| Pin # | Symbol | Description |
|---|---|---|
| 1 | FGND | Frame ground |
| 2 | Vss | Power supply(GND) |
| 3 | Vdd | Power supply(+) |
| 4 | Vo | Contrast Adjust |
| 5 | WR | Data write |
| 6 | RD | Data read |
| 7 | CE | Chip enable |
| 8 | C/D | Command / data select |
| 10 | RST | Reset |
| 11 - 18 | DB0 - DB7 | Data bus line |
| 19 | FS | Font select |

void lcd_init (void) – initializes p1.16 – p1.29 as outputs and initializes screens output mode and turns it on.

void wr_xd (data, command) – writes data and sends command.

void wr_auto (unsigned char data) – automatically writes data to lcds Data bus line.

void chk_busy (unsigned char autowr) – converts Data bus line pins to input and checks if buffer is clear.

void clrram (void) – clears the screen by writing 0 to every pixel byte by byte.

void disp_img (addr, xl, yl, img) – draws img of size xl , yl starting at addr.

void draw_string(x, y, str) – draws string at x,y (lcd_init sets display mode as img xor text so text is overlaid over the image in contrast).

## Spi.c

The Serial Peripheral Interface (SPI) is a synchronous serial communication interface. The communication is Carried out by using Four lines (serial data out (SDO), serial data in(SDI), serial clock (SCK) and Slave select (SSEL)).

SPI_init(void) – Initializes pins and starts the SPI module as a master on the bus.

SPI_write(char data) - Sends a data byte on the SPI bus as a master.

SPI_read(void) - Reads a data byte from the SPI bus.

## SD.c

SD_init(void) - initializes the SD card in SPI mode.

SD_sendCommand(cmd, arg) - sends a standard command to SD/MMC.

SD_readImage(startBlock, img) - reads a single image(2048 Bytes) from SD/MMC.

## Main.c

Almost all of the functionality is done using the interrupts but since it would be a very bad idea to run bulky tasks from with in interrupts they are mainly used to set flags letting the main function know what to do next

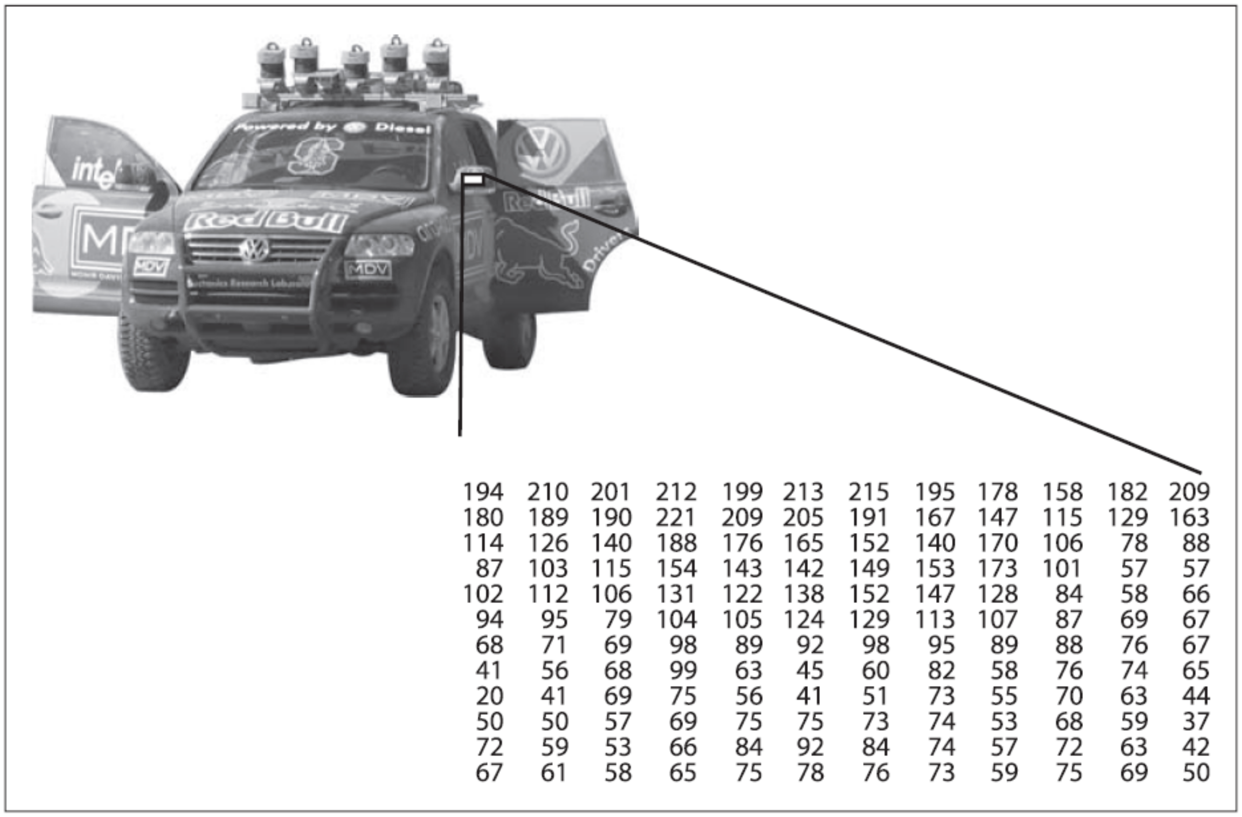
Void inititalize(void) – initializes system clock, buttons, lcd, timer, spi module and sd card.

Void next(void) – retrieves next img from sd card and displays it on the lcd along with its index.

## Appendix

### Image Data Basics

A digital image in its essence consists of a two-dimensional array of numbers. The color or gray shade displayed for a given picture element (pixel) depends on the number stored in the array for that pixel. A grayscale image is set up in such a way that each pixel takes on a value between zero and the number of gray scales or gray levels that the camera can record. Color images are similar to gray scale except that there are three bands, or channels, corresponding to the colors red, green, and blue. Thus, each pixel has three values associated with it.



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 194 | 210 | 201 | 212 | 199 | 213 | 215 | 195 | 178 | 158 | 182 | 209 |
| 180 | 189 | 190 | 221 | 209 | 205 | 191 | 167 | 147 | 115 | 129 | 163 |
| 114 | 126 | 140 | 188 | 176 | 165 | 152 | 140 | 170 | 106 | 78 | 88 |
| 87 | 103 | 115 | 154 | 143 | 142 | 149 | 153 | 173 | 101 | 57 | 57 |
| 102 | 112 | 106 | 131 | 122 | 138 | 152 | 147 | 128 | 84 | 58 | 66 |
| 94 | 95 | 79 | 104 | 105 | 124 | 129 | 113 | 107 | 87 | 69 | 67 |
| 68 | 71 | 69 | 98 | 89 | 92 | 98 | 95 | 89 | 88 | 76 | 67 |
| 41 | 56 | 68 | 99 | 63 | 45 | 60 | 82 | 58 | 76 | 74 | 65 |
| 20 | 41 | 69 | 75 | 56 | 41 | 51 | 73 | 55 | 70 | 63 | 44 |
| 50 | 50 | 57 | 69 | 75 | 75 | 73 | 74 | 53 | 68 | 59 | 37 |
| 72 | 59 | 53 | 66 | 84 | 92 | 84 | 74 | 57 | 72 | 63 | 42 |
| 67 | 61 | 58 | 65 | 75 | 78 | 76 | 73 | 59 | 75 | 69 | 50 |

Figure 2: digital grayscale image

The simplest type of image data is black and white. It is a binary image meaning each pixel is either 0 or 1. For this implementation we work with binary images as the screen used for display is a monochrome one.

## SD card basics

The MMC/SDC can be attached to the most microcontrollers via a generic SPI interface or some GPIO ports. Therefore the SPI mode is suitable for low cost embedded applications with no native host interface available.

The pins for SPI on the SD card are as follows

| Pin on SD card | Function | Connection to Pin on lpc |
|---|---|---|
| MOSI (Master Out Slave In) | The master refers to the device that generates the clock (the microcontroller), the SD card is the slave. Data on this pin travels from the microcontroller to the SD card. Also known as "DI". | MOSI |
| MISO (Master In Slave Out) | Data on this pin travels from the SD card to the microcontroller. Also known as "DO". | MISO |
| SCK | Serial clock pin, also known as "CLK" | SCK |
| CS | Chip select, the SD card pays attention to the data traveling on the SPI bus when this pin is low, and ignores the data on the bus when this pin is high | P0.11 |

**Power ON or card insersion** - After supply voltage reached 2.2 volts, wait for one millisecond at least. Set SPI clock rate between 100 kHz and 400 kHz. Set DI and CS high and apply 74 or more clock pulses to SCLK. The card will enter its native operating mode and go ready to accept native command.

**Software reset -** Send a *CMD0 with CS low* to reset the card. The card samples CS signal on a CMD0 is received successfully. If the CS signal is low, the card enters SPI mode and responds R1 with In Idle State bit (0x01). Since the CMD0 must be sent as a native command, the CRC field must have a valid value. When once the card enters SPI mode, the CRC feature is disabled and the CRC is not checked by the card, so that command transmission routine can be written with the hardcorded CRC value that valid for only CMD0 and CMD8 with the argument of zero. The CRC feature can also be switched with CMD59.

**Initialization -** In idle state, the card accepts only CMD0, CMD1, ACMD41,CMD58 and CMD59. Any other commands will be rejected. In this time, read OCR register and check working voltage range of the card. In case of the system supply voltage is out of working voltage range, the card must be rejected. Note that all cards work at supply voltage range of 2.7 to 3.6 volts at least, so that the host controller needs not check the OCR if supply voltage is in this range. The card initiates the initialization process when a *CMD1* is received. To detect end of the initialization process, the host controller must send CMD1 and check the response until end of the initialization.

## SPI Command Set

Each command is expressed in abbreviation like GO_IDLE_STATE or CMD<n>, <n> is the number of the command index and the value can be 0 to 63. Following table describes commands used for generic read/write and card initialization.

| Command Index | Argument | Response | Data | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD0 | None(0) | R1 | No | GO_IDLE_STATE | Software reset. |
| CMD1 | None(0) | R1 | No | SEND_OP_COND | Initiate initialization process. |
| ACMD41(*1 ) | *2 | R1 | No | APP_SEND_OP_COND | For only SDC. Initiate initialization process. |
| CMD8 | *3 | R7 | No | SEND_IF_COND | For only SDC V2. Check voltage range. |
| CMD9 | None(0) | R1 | Yes | SEND_CSD | Read CSD register. |
| CMD10 | None(0) | R1 | Yes | SEND_CID | Read CID register. |
| CMD12 | None(0) | R1b | No | STOP_TRANSMISSION | Stop to read data. |
| CMD16 | Block length[31:0] | R1 | No | SET_BLOCKLEN | Change R/W block size. |
| CMD17 | Address[31:0] | R1 | Yes | READ_SINGLE_BLOC K | Read a block. |
| CMD18 | Address[31:0] | R1 | Yes | READ_MULTIPLE_BLO CK | Read multiple blocks. |
| CMD23 | Number of blocks[15:0] | R1 | No | SET_BLOCK_COUNT | For only MMC. Define number of blocks to transfer with next multi-block read/write command. |
| ACMD23(*1 ) | Number of blocks[22:0] | R1 | No | SET_WR_BLOCK_ERAS E_COUNT | For only SDC. Define number of blocks to pre-erase with next multi-block write command. |
| CMD24 | Address[31:0] | R1 | Yes | WRITE_BLOCK | Write a block. |
| CMD25 | Address[31:0] | R1 | Yes | WRITE_MULTIPLE_BL OCK | Write multiple blocks. |
| CMD55(*1) | None(0) | R1 | No | APP_CMD | Leading command of ACMD<n> command. |
| CMD58 | None(0) | R3 | No | READ_OCR | Read OCR. |

## References

[1] Engineers garage – connecting sd card to avr

[2] Image processing in C – Dwayne Philips