## Intro:

This lab explores the interfacing of the c8051 microcontroller with different types of external RAM. External memory can be accessed using Ports 5 and 6 as an address bus, Port 7 as a data bus, and P4.6 and P4.7 as read and write enables respectively. The addresses 0x0000 to 0x1FFF are internal to the c8051, after which the microcontroller looks to external memory. The two types of RAM used for this lab were the Am9128, a 2k x 8-bit memory chip, and the MCM6147, a 4k x 1-bit memory chip. Since these chips are old, the clock must be slowed down to accommodate the read/write latency of the chips.

Materials/Procedure:

Part 1 of this lab makes sure everything works and is properly set up; one Am9128 chip is wired to the microcontroller, with the lower 8 address bits connected to P6, the higher 3 connected to P5, and the data lines connected to P7. The higher 5 bits of the address bus are used in the glue logic to enable the chip when writing to the range of 0x2000-0x27FF (see attached diagrams for more information). To make sure the chip is working correctly, data is written to the chip and read back and printed out. In proper operation, read/write should be possible while the chip is being powered, and 0xFF should be read back when power is off (as unknown bits tend to float high).

Part 2 of this lab checks that data is written correctly to external memory. With the same hardware setup as Part 1, 0x55 is written to addresses 0x2000 to 0x27FF and read back. If the read is different from the write, the address at which this occurred is written to a buffer array of length 256. If this buffer fills (ie. contains 256 elements), its contents are dumped to the screen and emptied; otherwise, the errors are printed once the program has finished writing to all addresses. If everything works as expected, for addresses 0x2000 to 0x27FF, no errors should occur. Beyond this, if no more memory is added to accept these writes/reads-- or if the chip is not powered, errors will occur from 0xFF being read back (for the reason stated above).

In part 3, additional memory is added, in the form of the MCM6147 chip(s) to hold addresses 0x3000 to 0x3FFF. The address bus, data line (8 chips x one data line each for a full byte), and read/write enables are connected to the same as for the Am9128. The top 4 bits of the address bus are used in glue logic to enable the chip in its designated range. If everything is done correctly, writing 2 bytes to the 0x2000-0x27FF range will read back the same data, writing to the 0x2800-0x2FFF range will read back 0xFF, as no memory is associated with those addresses, and writing to the 0x3000-0x3FFF range will read back as many bits as MCM6147 chips wired up, with the remaining bits of the full 2 bytes being HIGH.

## Results:

From the lab preface, we could see that the key we input mirrored the output character. This indicated that the external memory was connected correctly. This was quite laborious as the wires we had cut out the first time turned out to be a little too long to guarantee accurate writing and reading. In part 1, we saw that the output being read from the chip was the same as the input we provide the chip, “AA”. When the chip is powered off, the AA lingers for a little bit before shifting to ‘FF”. This indicated that we were indeed writing to and reading from external memory. This was slightly troublesome as the Am9128 seemed to magically draw power from the ether. It was later determined that that buffers, whilst plugged in, but in no way attached to the proto-board, managed to deliver approximately 3 volts to the proto-board. Part 2 required that we try and detect memory errors and push these errors into a buffer, to be flushed when full or the loop terminates. Our program indicates that no error is present when the chip is powered on. Upon powering off, the screen is quickly filled with the buffer flushes indicating and error in reading from the chip. This part was also tricky, as the chip then started to draw power from the underlying mana permeating the room. This problem could not be replicated and was dismissed as a mis-wiring mishap. Having repeated this for patterns “0xAA” and “0x55”, we could safely assume that our chip was working properly. After determining the proper glue logic, the MCM6147 was properly attached to the 8051. Due to the limited number of chips, we only wired one MCM6147. This meant we could only read the last bit when reading from the MCM6147. When our program scrolled from memory address 0x2000 to 0x27FF, we saw outputs of 1 or 0, depending on what we wrote to the chip. At 0x2800, the output switched to “FF” indicating that there was nothing read from the chip, as there was no memory present at that address. When the counter reached 0x3000, we saw the output change to “FE” or stay “FF” depending on whether we wrote a 0 or a 1 to the memory. The most time consuming part of this was the proper wiring of the glue logic, as we encountered a faulty NAND chip inhibiting progress.

## Conclusion:

In conclusion, our results matched everything specified by the lab. With more time, we would have definitely wired more chips up, another Am9128 and other MCM6147s. We could have also wired an LED to a memory slot to be controlled by writing to a particular memory address.

## Appendix:

Source Code:

Lab5-1

//------------------------------------------------------------------------------------

// lab5-1.c

// Shirley Du

// Victor Huang

//------------------------------------------------------------------------------------

// This software writes a character to a specific address in external memory

// NOTES:

// (1) /WR = P4.7 (CNTRL signal)

// (2) /RD = P4.6 (CNTRL signal)

// (3) D0-D7 = P7.0-P7.7 (DATA bus)

// (4) A0-A7 = P6.0-P6.7 (ADR bus lo byte)

// (5) A8-A15 = P5.0-P5.7 (ADR bus hi byte)

//------------------------------------------------------------------------------------

// Includes

//------------------------------------------------------------------------------------

#include <c8051f120.h>

#include <stdio.h>

#include "putget.h"

//------------------------------------------------------------------------------------

// Global Constants

//------------------------------------------------------------------------------------

#define EXTCLK 22118400 // External oscillator frequency in Hz

#define SYSCLK 22118400 // Output of crystal oscillator

#define BAUDRATE 28800 // UART baud rate in bps

//------------------------------------------------------------------------------------

// Function Prototypes

//------------------------------------------------------------------------------------

void main(void);

void SYSCLK\_INIT(void);

void PORT\_INIT(void);

void UART0\_INIT(void);

unsigned char \_sdcc\_external\_startup(void);

//------------------------------------------------------------------------------------

// \_sdcc\_external\_startup

//------------------------------------------------------------------------------------

//

//This is special function called by the system BEFORE main() is executed

//

// Disable watchdog timer before normal initialization - needed for memory

//

unsigned char \_sdcc\_external\_startup(void)

{

WDTCN = 0xDE; // Disable the watchdog timer

WDTCN = 0xAD;

return 0; // init everything else normally

}

//------------------------------------------------------------------------------------

// MAIN Routine

//------------------------------------------------------------------------------------

void main(void)

{

xdata unsigned char \*p;

int mem = 0x2000;

int i;

p = (\_\_xdata unsigned char\*)(0x2000);

SYSCLK\_INIT(); // Initialize the oscillator

PORT\_INIT(); // Initialize the Crossbar and GPIO

UART0\_INIT(); // Initialize UART0

SFRPAGE = UART0\_PAGE; // Direct output to UART0

printf("\033[2J"); // Erase ANSI terminal & move cursor to home position

printf("crystal fair\n\n\r");

\*p = 'a';

while(1)

{

for(i=0; i<0x900;i++)

{

p[i] = 0xAA; //write "AA" to the slot

printf("\r\nCharacter stored in memory %x: %x\r\n", mem, p[i]);

mem++; //update what memory address is being read

}

mem=0x2000;

}

}

//------------------------------------------------------------------------------------

// SYSCLK\_Init

//------------------------------------------------------------------------------------

//

// Initialize the system clock to use a 22.1184MHz crystal as its clock source

//

void SYSCLK\_INIT(void)

{

int i;

char SFRPAGE\_SAVE;

SFRPAGE\_SAVE = SFRPAGE; // Save Current SFR page SFRPAGE = CONFIG\_PAGE;

SFRPAGE = CONFIG\_PAGE;

OSCXCN = 0x67; // Start ext osc with 22.1184MHz crystal

for(i=0; i < 3000; i++); // Wait for the oscillator to start up

while(!(OSCXCN & 0x80));

CLKSEL = 0x01; // Switch to the external crystal oscillator

OSCICN = 0x00 ; // Disable the internal oscillator

SFRPAGE = SFRPAGE\_SAVE; // Restore SFR page

}

//------------------------------------------------------------------------------------

// PORT\_Init

//------------------------------------------------------------------------------------

//

// Configure the Crossbar and GPIO ports

//

void PORT\_INIT(void)

{

char SFRPAGE\_SAVE = SFRPAGE; // Save Current SFR page

SFRPAGE = CONFIG\_PAGE;

XBR0 = 0x04; // Enable UART0

XBR1 = 0x00;

XBR2 = 0x40; // Enable Crossbar and weak pull-up

P0MDOUT |= 0x01; // Set TX0 pin to push-pull

P4MDOUT = 0xFF; // Output configuration for P4 all pushpull

P5MDOUT = 0xFF; // Output configuration for P5 pushpull EM addr

P6MDOUT = 0xFF; // Output configuration for P6 pushpull EM addr

P7MDOUT = 0xFF; // Output configuration for P7 pushpull EM data

P5 = 0xFF;

P6 = 0xFF;

P7 = 0xFF;

// EMI\_Init, split mode with no banking

SFRPAGE = EMI0\_PAGE;

EMI0CF = 0x3b; //34

EMI0TC = 0xFF;

SFRPAGE = SFRPAGE\_SAVE; // Restore SFR page

}

//------------------------------------------------------------------------------------

// UART0\_Init

//------------------------------------------------------------------------------------

//

// Configure the UART0 using Timer1, for <baudrate> and 8-N-1

//

void UART0\_INIT(void)

{

char SFRPAGE\_SAVE;

SFRPAGE\_SAVE = SFRPAGE; // Save Current SFR page

SFRPAGE = TIMER01\_PAGE;

TCON = 0x40;

TMOD &= 0x0F;

TMOD |= 0x20; // Timer1, Mode 2, 8-bit reload

CKCON |= 0x10; // Timer1 uses SYSCLK as time base

// TH1 = 256 - SYSCLK/(BAUDRATE\*32) Set Timer1 reload baudrate value T1 Hi Byte

TH1 = 0xE8; // 0xE8 = 232

TR1 = 1; // Start Timer1

SFRPAGE = UART0\_PAGE;

SCON0 = 0x50; // Mode 1, 8-bit UART, enable RX

SSTA0 = 0x00; // SMOD0 = 0, in this mode

// TH1 = 256 - SYSCLK/(baud rate \* 32)

TI0 = 1; // Indicate TX0 ready

SFRPAGE = SFRPAGE\_SAVE; // Restore SFR page

}

Lab5-2

//------------------------------------------------------------------------------------

// lab5-2.c

// Shirley Du

// Victor Huang

//------------------------------------------------------------------------------------

// This software writes a character to a specific address in external memory

// NOTES:

// (1) /WR = P4.7 (CNTRL signal)

// (2) /RD = P4.6 (CNTRL signal)

// (3) D0-D7 = P7.0-P7.7 (DATA bus)

// (4) A0-A7 = P6.0-P6.7 (ADR bus lo byte)

// (5) A8-A15 = P5.0-P5.7 (ADR bus hi byte)

//------------------------------------------------------------------------------------

// Includes

//------------------------------------------------------------------------------------

#include <c8051f120.h>

#include <stdio.h>

#include "putget.h"

//------------------------------------------------------------------------------------

// Global Constants

//------------------------------------------------------------------------------------

#define EXTCLK 22118400 // External oscillator frequency in Hz

#define SYSCLK 22118400 // Output of crystal oscillator

#define BAUDRATE 28800 // UART baud rate in bps

//------------------------------------------------------------------------------------

// Function Prototypes

//------------------------------------------------------------------------------------

void main(void);

void SYSCLK\_INIT(void);

void PORT\_INIT(void);

void UART0\_INIT(void);

unsigned char \_sdcc\_external\_startup(void);

//------------------------------------------------------------------------------------

// \_sdcc\_external\_startup

//------------------------------------------------------------------------------------

//

//This is special function called by the system BEFORE main() is executed

//

// Disable watchdog timer before normal initialization - needed for memory

//

unsigned char \_sdcc\_external\_startup(void)

{

WDTCN = 0xDE; // Disable the watchdog timer

WDTCN = 0xAD;

return 0; // init everything else normally

}

//------------------------------------------------------------------------------------

// MAIN Routine

//------------------------------------------------------------------------------------

void main(void)

{

xdata unsigned char \*p;

int mem = 0x2000;

int i;

unsigned static char \_\_xdata count[256];

int buf = 0;

int j;

p = (\_\_xdata unsigned char\*)(0x2000);

SYSCLK\_INIT(); // Initialize the oscillator

PORT\_INIT(); // Initialize the Crossbar and GPIO

UART0\_INIT(); // Initialize UART0

SFRPAGE = UART0\_PAGE; // Direct output to UART0

printf("\033[2J"); // Erase ANSI terminal & move cursor to home position

printf("crystal fair\n\n\r");

\*p = 'a';

while(1)

{

for(i=0; i<0x800;i++)

{

//write the value to the address

p[i] = 0x55;

//printf("\r\nCharacter stored in memory %x: %x\r\n", mem, p[i]);

mem++;

if (p[i] != 0x55)

{

//printf("error\n\r");

count[buf] = i;

buf++;

//buffer is full, dump

if (buf == 255)

{

printf("dumping buffer:\n\r");

for(j=0; j<255; j++)

printf("%x\n\r", count[j]+0x2000);

buf = 0;

}

}

}

mem=0x2000;

//When the addresses are visited, print out errors

if(buf > 0)

{

for(j=0; j<buf; j++)

printf("%d\n\r", count[j]);

}

else

{

printf("no errors found\n\r");

}

}

}

//------------------------------------------------------------------------------------

// SYSCLK\_Init

//------------------------------------------------------------------------------------

//

// Initialize the system clock to use a 22.1184MHz crystal as its clock source

//

void SYSCLK\_INIT(void)

{

int i;

char SFRPAGE\_SAVE;

SFRPAGE\_SAVE = SFRPAGE; // Save Current SFR page SFRPAGE = CONFIG\_PAGE;

SFRPAGE = CONFIG\_PAGE;

OSCXCN = 0x67; // Start ext osc with 22.1184MHz crystal

for(i=0; i < 3000; i++); // Wait for the oscillator to start up

while(!(OSCXCN & 0x80));

CLKSEL = 0x01; // Switch to the external crystal oscillator

OSCICN = 0x00 ; // Disable the internal oscillator

SFRPAGE = SFRPAGE\_SAVE; // Restore SFR page

}

//------------------------------------------------------------------------------------

// PORT\_Init

//------------------------------------------------------------------------------------

//

// Configure the Crossbar and GPIO ports

//

void PORT\_INIT(void)

{

char SFRPAGE\_SAVE = SFRPAGE; // Save Current SFR page

SFRPAGE = CONFIG\_PAGE;

XBR0 = 0x04; // Enable UART0

XBR1 = 0x00;

XBR2 = 0x40; // Enable Crossbar and weak pull-up

P0MDOUT |= 0x01; // Set TX0 pin to push-pull

P4MDOUT = 0xFF; // Output configuration for P4 all pushpull

P5MDOUT = 0xFF; // Output configuration for P5 pushpull EM addr

P6MDOUT = 0xFF; // Output configuration for P6 pushpull EM addr

P7MDOUT = 0xFF; // Output configuration for P7 pushpull EM data

P5 = 0xFF;

P6 = 0xFF;

P7 = 0xFF;

// EMI\_Init, split mode with no banking

SFRPAGE = EMI0\_PAGE;

EMI0CF = 0x3b; //34

EMI0TC = 0xFF;

SFRPAGE = SFRPAGE\_SAVE; // Restore SFR page

}

//------------------------------------------------------------------------------------

// UART0\_Init

//------------------------------------------------------------------------------------

//

// Configure the UART0 using Timer1, for <baudrate> and 8-N-1

//

void UART0\_INIT(void)

{

char SFRPAGE\_SAVE;

SFRPAGE\_SAVE = SFRPAGE; // Save Current SFR page

SFRPAGE = TIMER01\_PAGE;

TCON = 0x40;

TMOD &= 0x0F;

TMOD |= 0x20; // Timer1, Mode 2, 8-bit reload

CKCON |= 0x10; // Timer1 uses SYSCLK as time base

// TH1 = 256 - SYSCLK/(BAUDRATE\*32) Set Timer1 reload baudrate value T1 Hi Byte

TH1 = 0xE8; // 0xE8 = 232

TR1 = 1; // Start Timer1

SFRPAGE = UART0\_PAGE;

SCON0 = 0x50; // Mode 1, 8-bit UART, enable RX

SSTA0 = 0x00; // SMOD0 = 0, in this mode

// TH1 = 256 - SYSCLK/(baud rate \* 32)

TI0 = 1; // Indicate TX0 ready

SFRPAGE = SFRPAGE\_SAVE; // Restore SFR page

}

Lab5-3

//------------------------------------------------------------------------------------

// lab5-3.c

// Shirley Du

// Victor Huang

//------------------------------------------------------------------------------------

// This software writes a character to a specific address in external memory

// NOTES:

// (1) /WR = P4.7 (CNTRL signal)

// (2) /RD = P4.6 (CNTRL signal)

// (3) D0-D7 = P7.0-P7.7 (DATA bus)

// (4) A0-A7 = P6.0-P6.7 (ADR bus lo byte)

// (5) A8-A15 = P5.0-P5.7 (ADR bus hi byte)

//------------------------------------------------------------------------------------

// Includes

//------------------------------------------------------------------------------------

#include <c8051f120.h>

#include <stdio.h>

#include "putget.h"

//------------------------------------------------------------------------------------

// Global Constants

//------------------------------------------------------------------------------------

#define EXTCLK 22118400 // External oscillator frequency in Hz

#define SYSCLK 22118400 // Output of crystal oscillator

#define BAUDRATE 28800 // UART baud rate in bps

//------------------------------------------------------------------------------------

// Function Prototypes

//------------------------------------------------------------------------------------

void main(void);

void SYSCLK\_INIT(void);

void PORT\_INIT(void);

void UART0\_INIT(void);

unsigned char \_sdcc\_external\_startup(void);

//------------------------------------------------------------------------------------

// \_sdcc\_external\_startup

//------------------------------------------------------------------------------------

//

//This is special function called by the system BEFORE main() is executed

//

// Disable watchdog timer before normal initialization - needed for memory

//

unsigned char \_sdcc\_external\_startup(void)

{

WDTCN = 0xDE; // Disable the watchdog timer

WDTCN = 0xAD;

return 0; // init everything else normally

}

//------------------------------------------------------------------------------------

// MAIN Routine

//------------------------------------------------------------------------------------

void main(void)

{

\_\_xdata unsigned char \*p;

int mem = 0x2000;

int i;

p = (\_\_xdata unsigned char\*)(0x2000);

SYSCLK\_INIT(); // Initialize the oscillator

PORT\_INIT(); // Initialize the Crossbar and GPIO

UART0\_INIT(); // Initialize UART0

SFRPAGE = UART0\_PAGE; // Direct output to UART0

printf("\033[2J"); // Erase ANSI terminal & move cursor to home position

printf("crystal fair\n\n\r");

\*p = 'a';

while(1)

{

//go from 0x2600 to 0x3100

for(i=0x0600; i<0x1100;i++)

{

p = (\_\_xdata unsigned char\*)(mem);

\*p = 0;

printf("\r\nCharacter stored in memory %x: %x\r\n", mem, \*p);

mem = 0x2000 + i;

}

}

}

//------------------------------------------------------------------------------------

// SYSCLK\_Init

//------------------------------------------------------------------------------------

//

// Initialize the system clock to use a 22.1184MHz crystal as its clock source

//

void SYSCLK\_INIT(void)

{

int i;

char SFRPAGE\_SAVE;

SFRPAGE\_SAVE = SFRPAGE; // Save Current SFR page SFRPAGE = CONFIG\_PAGE;

SFRPAGE = CONFIG\_PAGE;

OSCXCN = 0x67; // Start ext osc with 22.1184MHz crystal

for(i=0; i < 3000; i++); // Wait for the oscillator to start up

while(!(OSCXCN & 0x80));

CLKSEL = 0x01; // Switch to the external crystal oscillator

OSCICN = 0x00 ; // Disable the internal oscillator

SFRPAGE = SFRPAGE\_SAVE; // Restore SFR page

}

//------------------------------------------------------------------------------------

// PORT\_Init

//------------------------------------------------------------------------------------

//

// Configure the Crossbar and GPIO ports

//

void PORT\_INIT(void)

{

char SFRPAGE\_SAVE = SFRPAGE; // Save Current SFR page

SFRPAGE = CONFIG\_PAGE;

XBR0 = 0x04; // Enable UART0

XBR1 = 0x00;

XBR2 = 0x40; // Enable Crossbar and weak pull-up

P0MDOUT |= 0x01; // Set TX0 pin to push-pull

P4MDOUT = 0xFF; // Output configuration for P4 all pushpull

P5MDOUT = 0xFF; // Output configuration for P5 pushpull EM addr

P6MDOUT = 0xFF; // Output configuration for P6 pushpull EM addr

P7MDOUT = 0xFF; // Output configuration for P7 pushpull EM data

P5 = 0xFF;

P6 = 0xFF;

P7 = 0xFF;

// EMI\_Init, split mode with no banking

SFRPAGE = EMI0\_PAGE;

EMI0CF = 0x3b; //34

EMI0TC = 0xFF;

SFRPAGE = SFRPAGE\_SAVE; // Restore SFR page

}

//------------------------------------------------------------------------------------

// UART0\_Init

//------------------------------------------------------------------------------------

//

// Configure the UART0 using Timer1, for <baudrate> and 8-N-1

//

void UART0\_INIT(void)

{

char SFRPAGE\_SAVE;

SFRPAGE\_SAVE = SFRPAGE; // Save Current SFR page

SFRPAGE = TIMER01\_PAGE;

TCON = 0x40;

TMOD &= 0x0F;

TMOD |= 0x20; // Timer1, Mode 2, 8-bit reload

CKCON |= 0x10; // Timer1 uses SYSCLK as time base

// TH1 = 256 - SYSCLK/(BAUDRATE\*32) Set Timer1 reload baudrate value T1 Hi Byte

TH1 = 0xE8; // 0xE8 = 232

TR1 = 1; // Start Timer1

SFRPAGE = UART0\_PAGE;

SCON0 = 0x50; // Mode 1, 8-bit UART, enable RX

SSTA0 = 0x00; // SMOD0 = 0, in this mode

// TH1 = 256 - SYSCLK/(baud rate \* 32)

TI0 = 1; // Indicate TX0 ready

SFRPAGE = SFRPAGE\_SAVE; // Restore SFR page

}