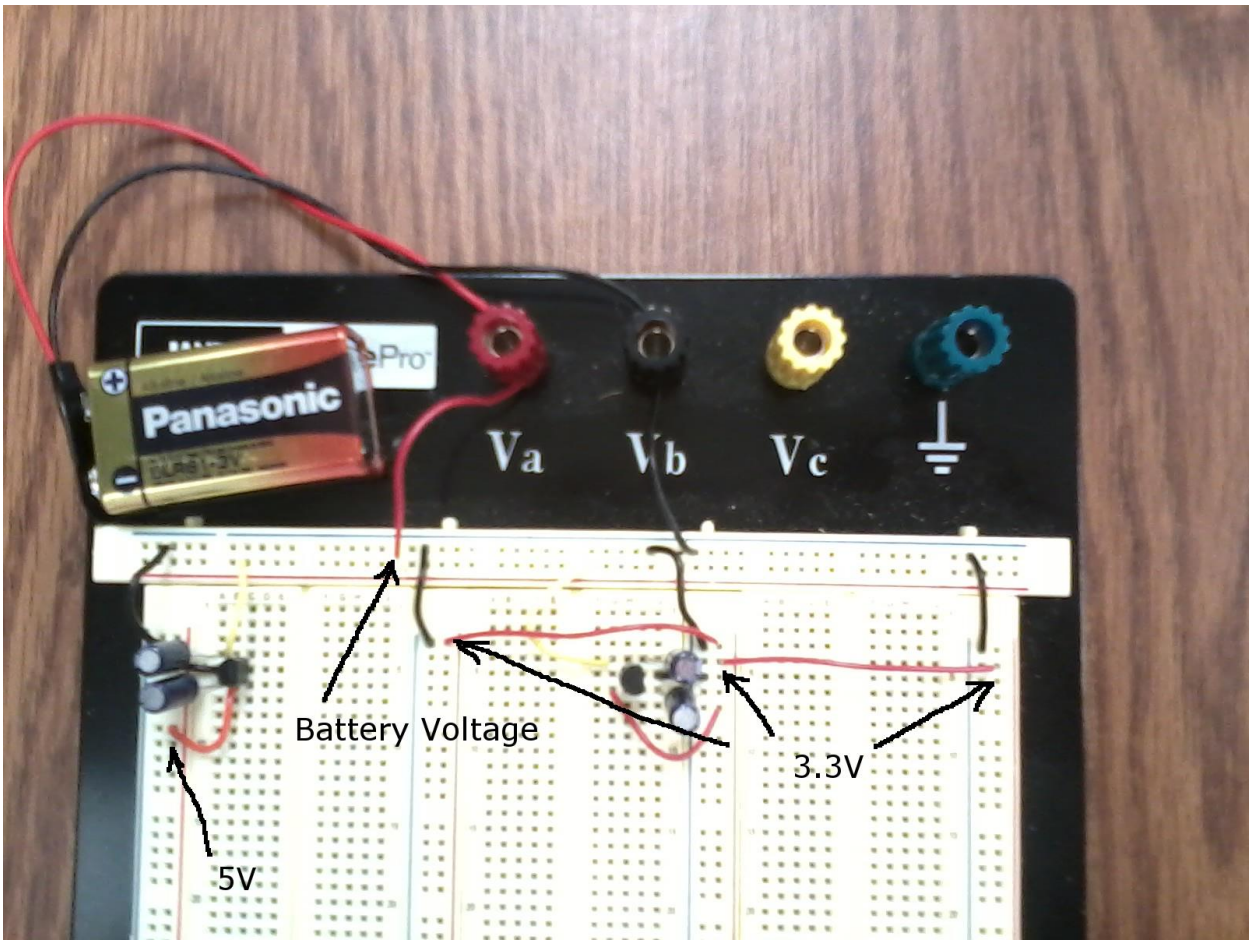
CpE5160 Hardware Construction and Debugging Guide

The purpose of this document is to give a few guidelines for the construction of the 8051 microcontroller circuit. There will also be a few pointers when debugging hardware and software.

The Power Supply

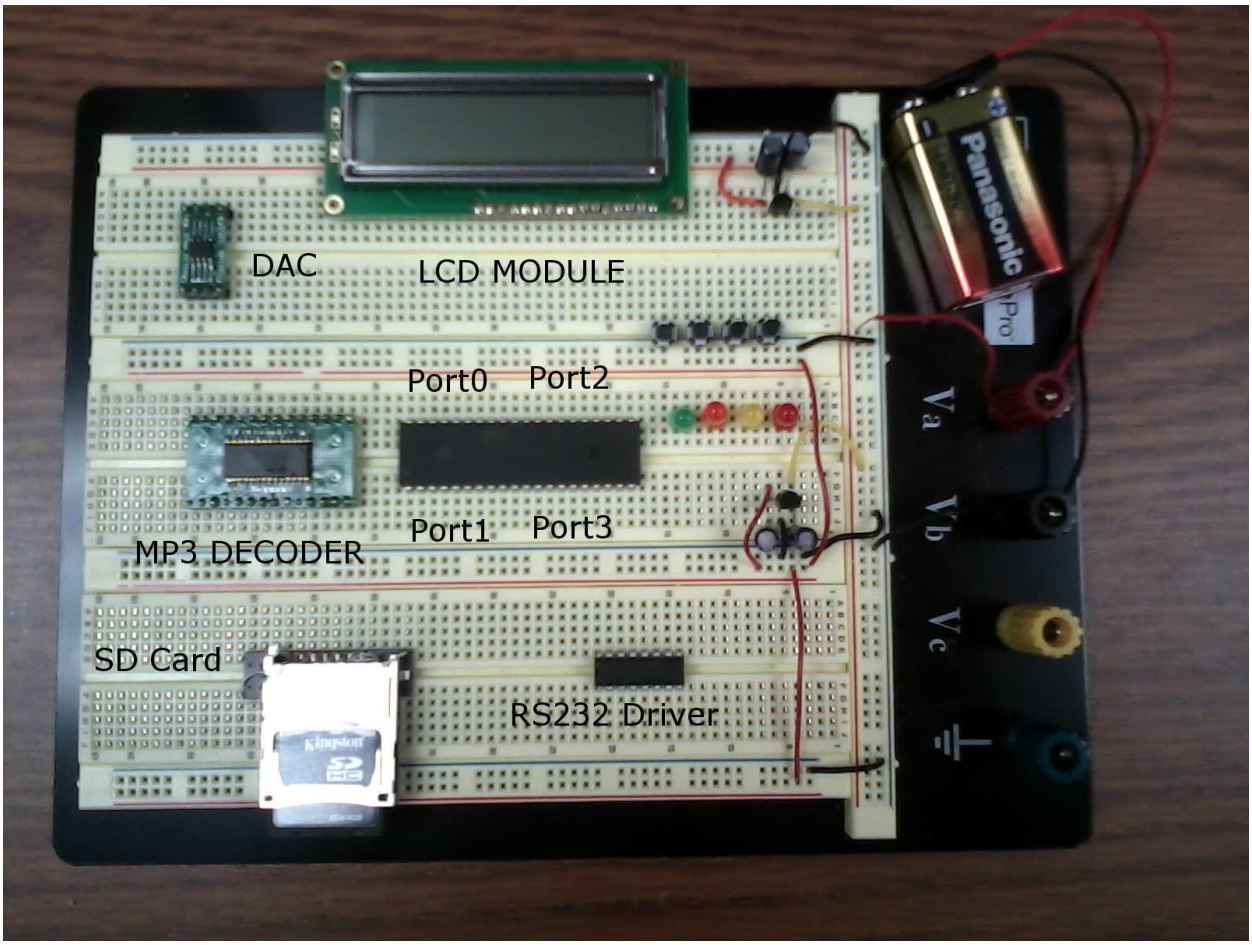
The recommendation is to connect the 9V battery clip wires to a couple of binding posts (preferably the red and black posts). Then connect solid wires from the binding posts to the horizontal bus connections. One of these wires can then be used as a power switch. The horizontal bus connections can then be connected to the 3.3V and 5V voltage regulators. The regulated voltage can then be connected to the vertical bus connections. See the figure below.



Voltage Distribution Diagram

Parts Placement

Where you choose to place the parts is not very critical. My recommendation is to place the microcontroller in the center section of the breadboard with pin 1 near the MP3 decoder. The SD card and MP3 decoder will connect to port 1. The UART connections are on port 3, so the RS232 driver should be close to this port. The DAC and LCD module must be connected to 5V, so they are near the 5V rail or regulator. The other connections are up to the student. I connected the LCD to port 0 and the switches and LEDs to port 2, but you may choose your own connections. A diagram is given below to show a possible arrangement of parts.



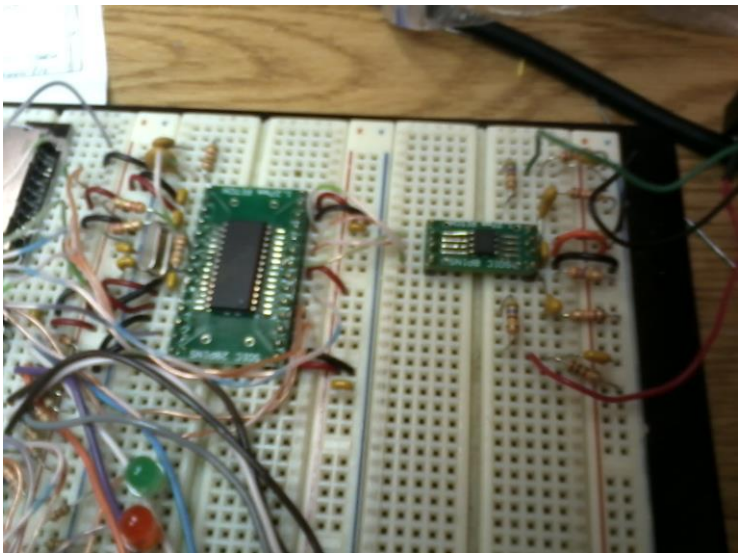
Parts Placement Diagram

This arrangement keeps the SD card and MP3 decoder close to Port1 so the high speed communication connections are kept short. There is also a high speed communication connection between the MP3 decoder and the DAC, so these parts are located close together. The LCD module is located close to Port0 and the RS232 driver is located close to Port3 just to cut down on the wire mess. The switches and LEDs are low speed connections and can be located wherever there is room.

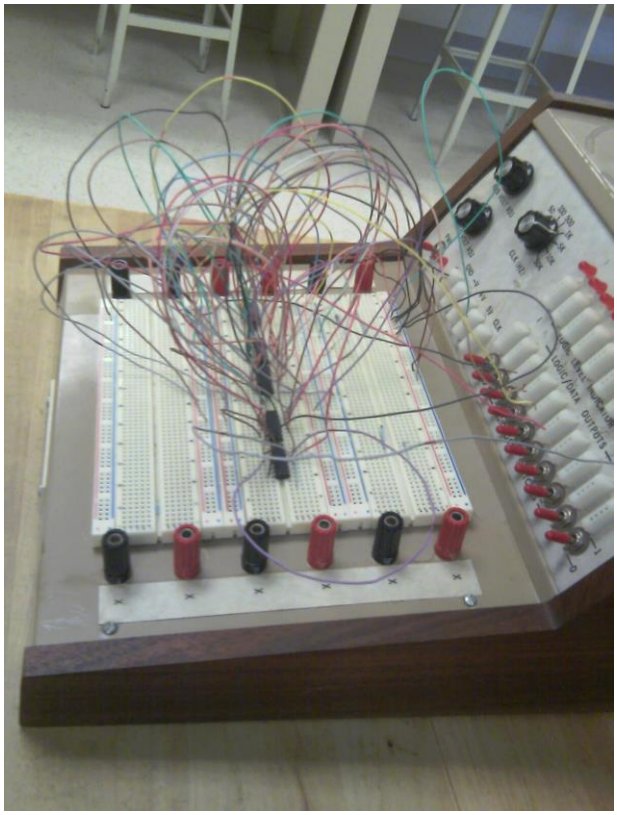
Wiring the Circuit

It is good practice to keep wire lengths as short as possible. This is especially important around the MP3 decoder and DAC. Poor wiring in this area can result in clock jitter which results in poor music quality

and it is difficult to determine if this is a hardware or software problem. The MP3 decoder will need a 0.1uF bypass capacitor on each power supply connection and very short connections for the PLL filter. The figures given below show an example of good short wiring and an example bad long wiring.



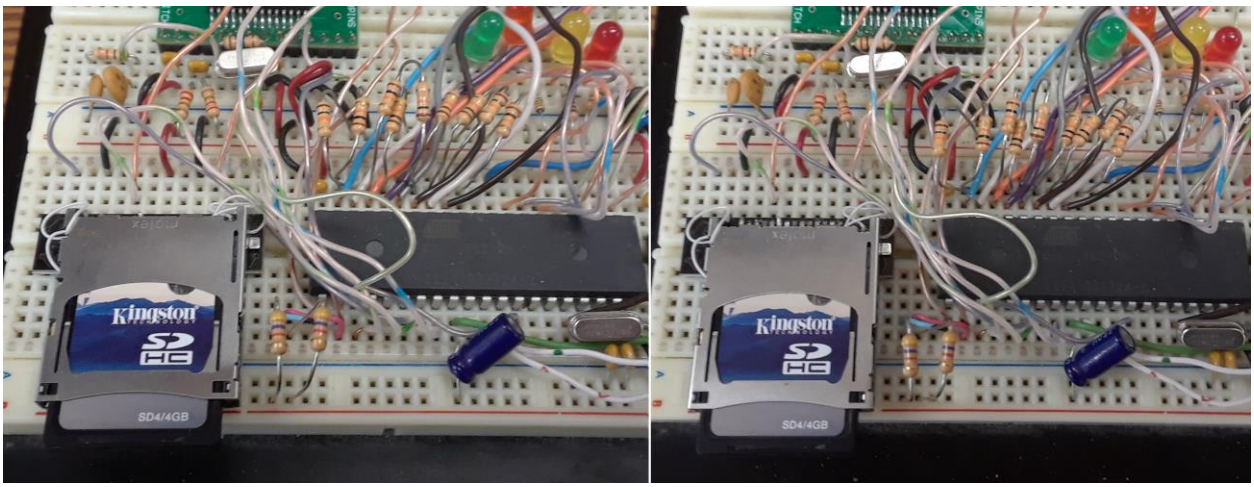
Example of short wires used in a circuit



Example of long wires used in a circuit

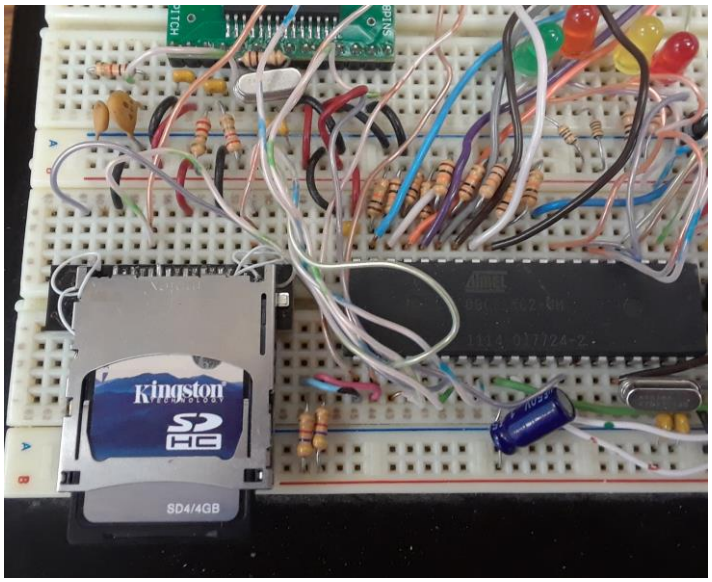
From a debugging point of view it is good practice not to run wires over parts. It makes it hard to measure signals on the pins and it makes the parts hard to remove.

Using short connections also applies to the component leads. The component leads are bare wires that can easily short to another component if they have long connections. I have had many students tell me that their hardware worked yesterday, but does not work today and the reason is that a component moved and caused a short circuit. Look at the following two pictures and see if you can determine which one works and which one does not and why.



Bad Examples of Component Lead Length

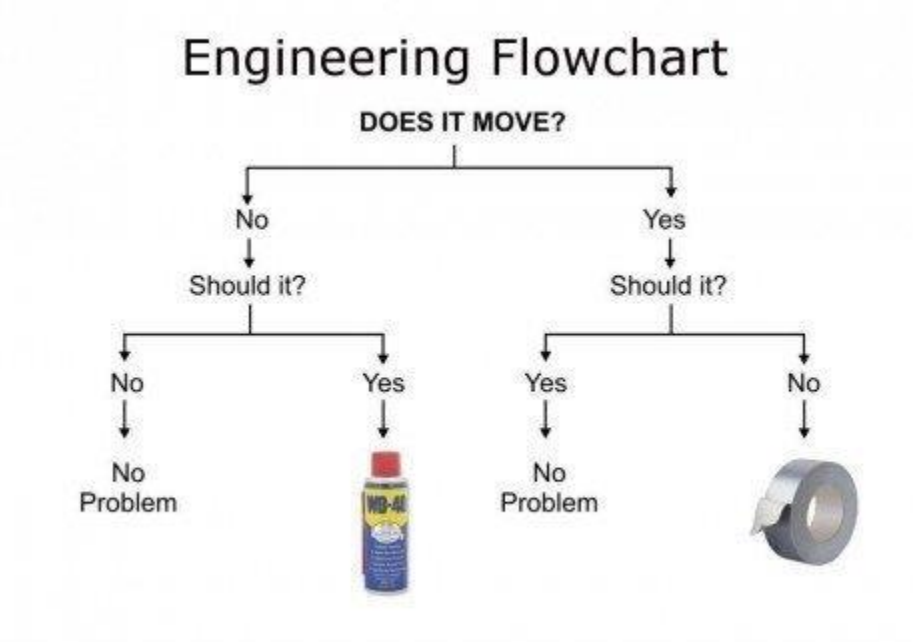
The circuit on the left is the one that does not work. There is short circuit that is relatively easy to spot between the 10K resistors in the center of the picture. This would be between resistors 5 and 6, if you numbered them from right to left. There is also a second short circuit between resistors 1 and 2 that is more difficult to spot. This shows how difficult it is to see the difference between the working and non-working circuits and why you should trim the component leads. The figure below show a good example of component lead length. Notice that the components do not have to be trimmed so that they are snug to the board. Just short enough that they don't bend easily and short to other components.



Example of Trimmed Component Leads

Debugging Hardware and Software

The following diagram about how mechanical engineers debug their systems will help us determine our approach to debugging our hardware and software.



This diagram illustrates two important parts of debugging a system. The first part is illustrated by the line "does it move?" This corresponds to taking measurements. This may be voltage measurements taken with a multi-meter or an oscilloscope or viewing waveforms with an oscilloscope or a logic analyzer. This could also correspond with observations made with a simulator or emulator when executing a program. The second part of debugging a system is understanding the expected operation of the system. Datasheets, application notes and classroom notes will be sources of information to help you determine what the expected operation is.

The Basic 8051 Microcontroller Circuit

The schematic for the basic microcontroller circuit is given with this handout. Also included with this handout is a complete parts list for the entire project. This parts list has a column that specifies which part of the project a particular part is used in. If you find that you have parts missing, please let me know. Parts with long leads such as resistors, capacitors and LEDs may have their leads trimmed to help keep the wiring short and neat.

A good first step is to build the power supply and check that it has the correct voltage before connecting other components. Remember to switch the power back off before wiring the rest of the circuit. After adding the additional circuitry, recheck the power supply to verify that the supply is not shorted. Also check the voltage on the RS232 Driver V+ and V- pins. These should be at roughly +6V and -6V. An oscilloscope can be used to check to see if the oscillator on the processor is working, however sometimes an oscilloscope probe may have too much capacitance and can stop an oscillator circuit. (Note that using long wires in the oscillator circuit can also prevent it from oscillating.) An alternative is

to check the ALE pin, it should output a signal at 1/6 (or 1/3 if the X2 bit is enabled) of the oscillator frequency (unless it has been set to quiet mode which disables the ALE pin).

After verifying that the microcontroller is operating, then you should attempt to download a program using F.L.I.P. (F.L.I.P. is a free program available from Atmel). The "Hello, Embedded World" program available on Canvas (originally given in chapter 3 of the book and modified to match the LED pin definition to your circuit) can be used to test your circuit. You should make sure the RTS and DTR pins are connected to the transistors for downloading and disconnected for running a program. A double pole, double throw, (DPDT) DIP switch is used to accomplish this in the circuit given in this handout. Also make sure the serial cable is connected between the PC COM port and your circuit. The first step is to connect to the microcontroller by selecting a microcontroller (under the Device menu or by clicking on the leftmost icon) and then connecting for communications (under the Settings menu or by clicking on the next icon and selecting RS232). If you cannot connect, then use an oscilloscope to verify that the Reset and /PSEN pins are switching properly (/PSEN must be low when Reset transitions from high to low). If these signals are not okay, check for the RTS and DTR signals. If they are not present, check the settings on F.L.I.P. (Settings>Preferences… ISP hardware conditions controlled by F.L.I.P. should be enabled). If RTS and DTR signals are present, recheck your wiring of the transistors in the ISP circuit. If these signals are okay, then check for an incoming "U" character on the RxD pin of the microcontroller and a corresponding "U" being transmitted back out on the TxD pin. If a character is received, but not transmitted back out, possibly the microcontroller has been damaged or is not connected correctly.

Tips for Debugging Software

Debugging a large project can be a daunting task where you are at a loss as to even where to start looking. The approach that can be taken is similar to the top down design approach, where a large problem is broken down into smaller more manageable problems. That is one reason why we have broken down this project into smaller parts. Once each part has been fully debugged, then errors should most likely come from any new code written for the new part of the project. Using this narrower focus we can then use whatever technique is most appropriate to narrow the search further until the error is found.

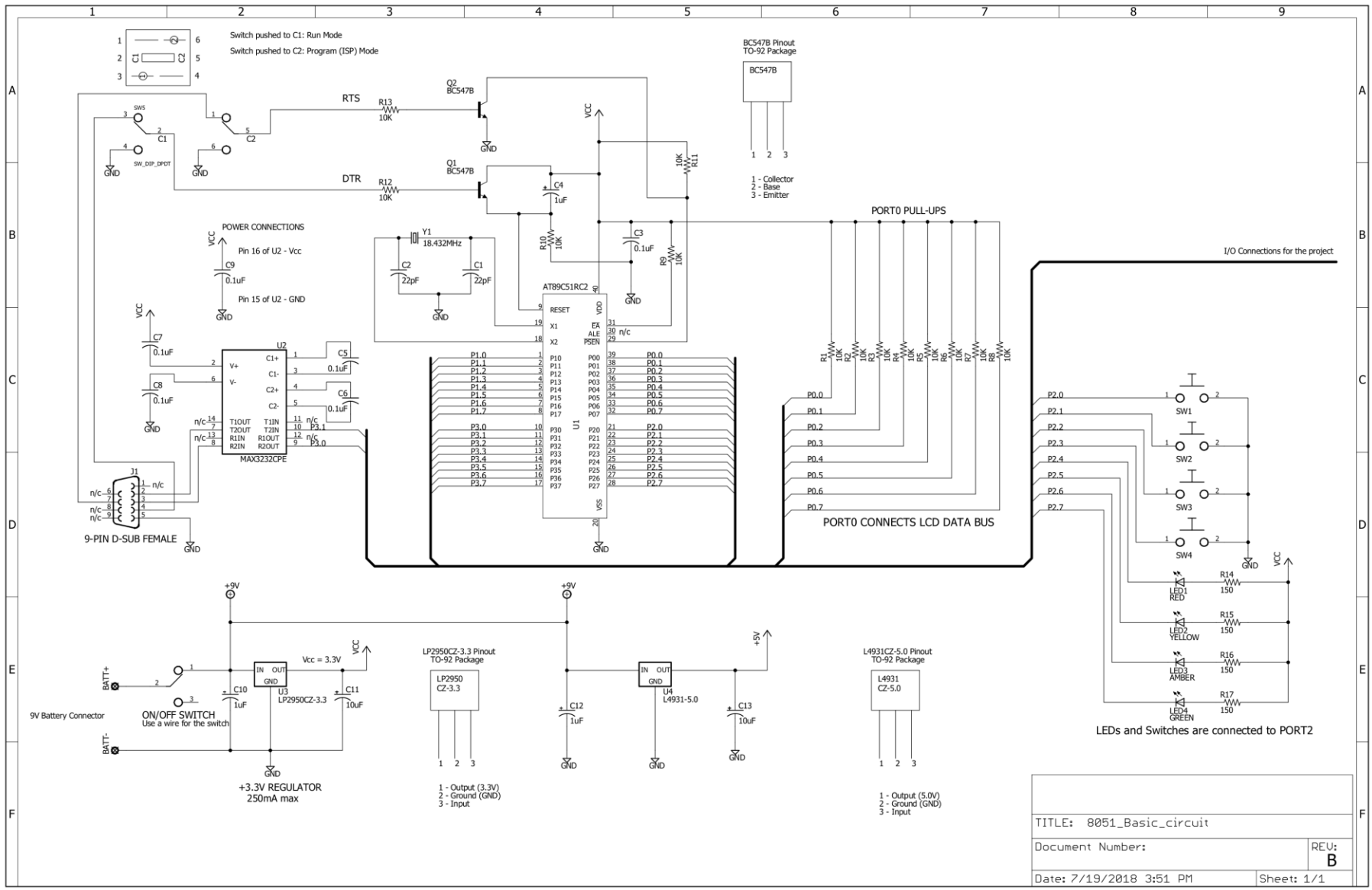
A simulator can be used to check that data is being manipulated correctly. An example is the calculations for the baud rate generator reload value. The compiler calculated value can be compared against a value calculated by the programmer. A complex calculation can be broken up into steps, so that intermediate values can be seen to help find what may be causing an error in the calculation. The results of bitwise logic operations and number conversions are other good places to use a simulator to check values. A simulator can also be used to determine if a loop is operating correctly, looping the correct number of times or why it does not exit.

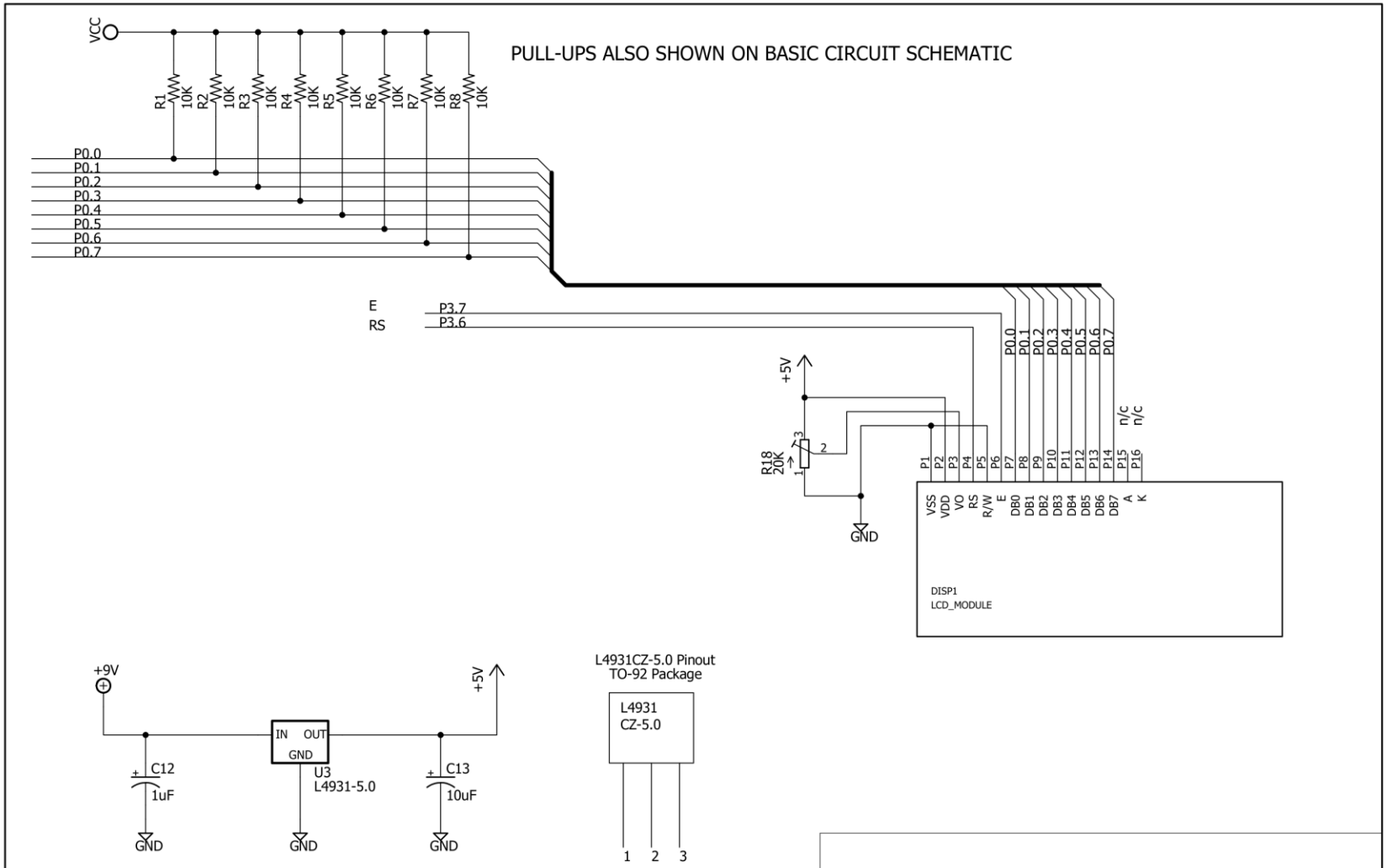
If a simulator can accurately represent the peripheral devices, then it can be used to check if the peripheral device has been initialized properly. The older version of Keil uVision that we have may not completely represent peripherals correctly because of the x2 mode that our microcontroller has and the availability of a baud rate generator. You may be able to compensate for this by adjusting the oscillator frequency in the simulator or by downloading a newer demo version of Keil uVision just for checking peripheral device initialization.

If you have access to an emulator, this gives you the ability to have breakpoints and single step mode like a simulator and allows you to see the code executed in your system. This gives you a lot more capability to see how your code interacts with external devices. An example of this would be a loop where the code is waiting for a value to be received from an external device. A breakpoint would be placed just after the loop so that code execution would stop when the value is received and then you can view it to see if it was the value you expected.

Since we do not have an emulator for the microcontroller, we will have to use other techniques to check code execution. Sometimes all that is needed is a few LEDs. If you think that the execution of your code is getting stuck somewhere, then switching on LEDs at entry points and switching them off at exit points can isolate the errant section of code. Using the example given above where a loop is waiting for a value to be received. An LED could be switched on just before the loop starts to indicate that the loop is executing and then switched off after the loop exits. This would tell the user that something was received. The value could then be written to a port, printed using the UART or sent to another output device to allow you to see the value. In the first experiment we are going to set up a couple of output devices that can be used to display portions of memory. The original purpose of these output devices is to allow us to see what is received from the SD card and what is stored on the SD card. They may also be useful in other areas of debugging the code.

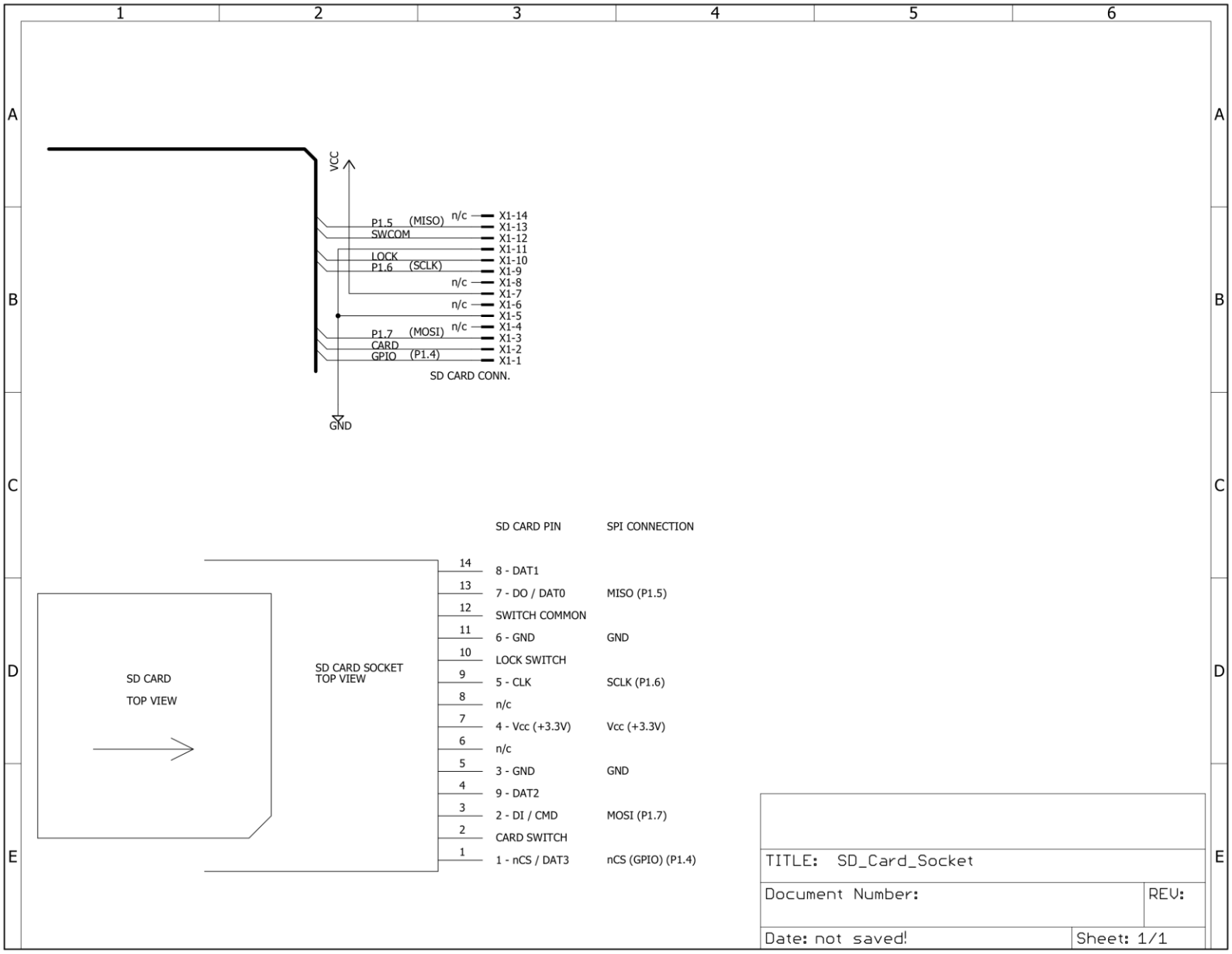
We will also learn how to use the logic analyzer to capture important signals. A logic analyzer can be used to view the input pin where a value was received and capture what signals were on that pin as the value was received. This can be compared to what was received to determine where an error might have occurred. For example, did the external device send an unexpected value, did the receive routine incorrectly receive the value or did another function act unexpectedly even though a correct value was received. The logic analyzer can also be used to view output signals that we generate to verify that those signals are what we expect them to be. These techniques may not show you what the error is, but they can narrow down the area of code that has the error. After the area is narrowed down, you may need to copy the errant code into its own small project with test input values and then step through the code to find the problem.

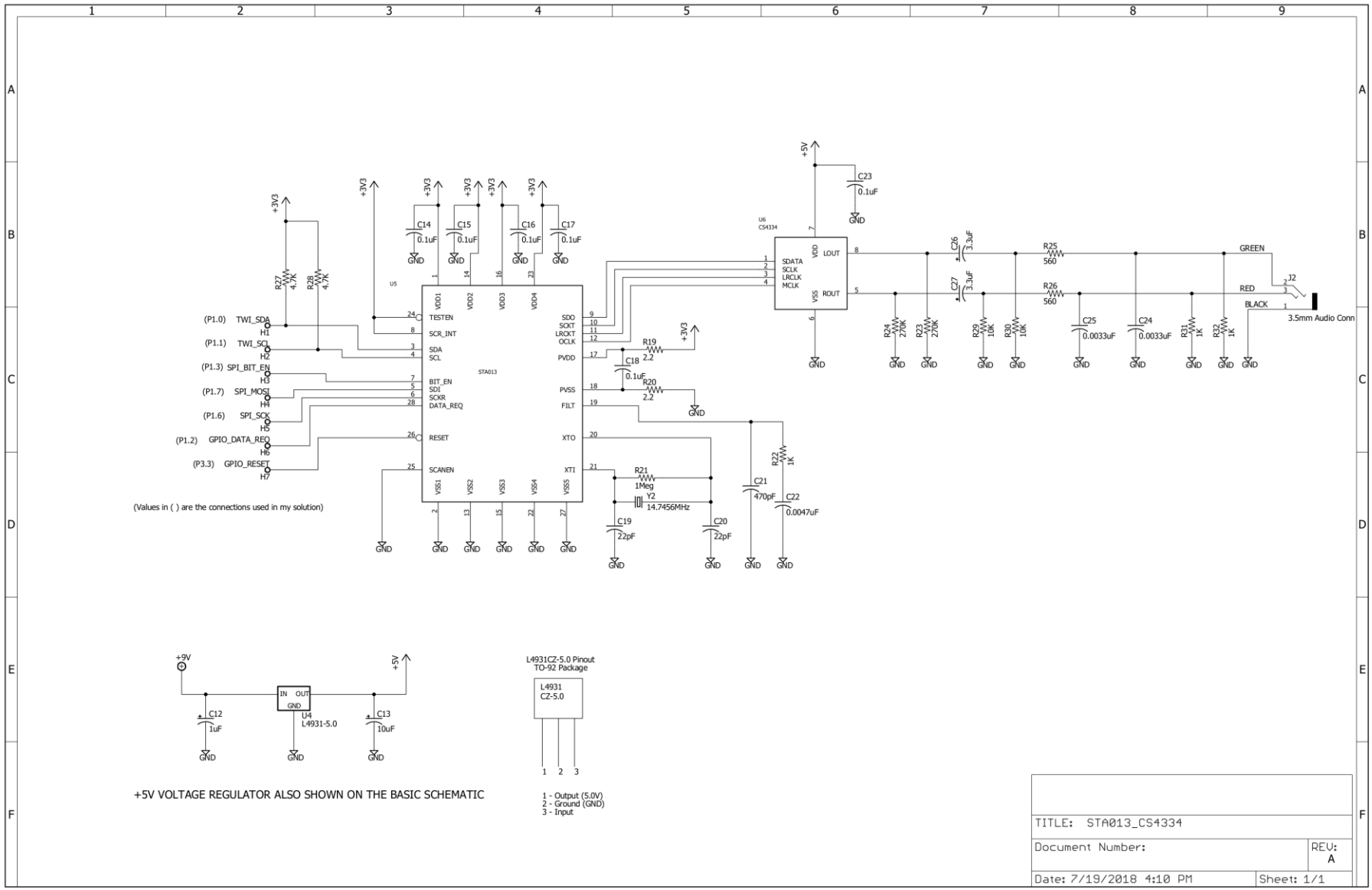

Switch pushed to C1: Run Mode
Switch pushed to C2: Program (ISP) Mode
SW5
SW_DIP_DPDT
RTS
DTR
R13 10K
R12 10K
Q2 BC547B
Q1 BC547B
BC547B Pinout TO-92 Package
BC547B
1 - Collector
2 - Base
3 - Emitter
R11 10K
C4 1uF
C3 0.1uF
R10 10K
R9 10K
Y1 18.432MHz
C2 22pF
C1 22pF
AT89C51RC2
RESET
X1
X2
EA
ALE
PSEN
VDD
VSS
n/c
U1
PORT0 PULL-UPS
R1 10K
R2 10K
R3 10K
R4 10K
R5 10K
R6 10K
R7 10K
R8 10K
PORT0 CONNECTS LCD DATA BUS
I/O Connections for the project
POWER CONNECTIONS
Pin 16 of U2 - Vcc
Pin 15 of U2 - GND
C9 0.1uF
C7 0.1uF
C8 0.1uF
C5 0.1uF
C6 0.1uF
U2
MAX3232CPE
T1OUT
T2OUT
R1IN
R2IN
T1IN
T2IN
R1OUT
R2OUT
C1+
C1-
C2+
C2-
V+
V-
P3.1
P3.0
J1
9-PIN D-SUB FEMALE
SW1
SW2
SW3
SW4
LED1 RED
LED2 YELLOW
LED3 AMBER
LED4 GREEN
R14 150
R15 150
R16 150
R17 150
LEDs and Switches are connected to PORT2
9V Battery Connector
ON/OFF SWITCH
Use a wire for the switch
BATT+
BATT-
C10 1uF
C11 10uF
U3 LP2950CZ-3.3
Vcc = 3.3V
+3.3V REGULATOR
250mA max
LP2950CZ-3.3 Pinout TO-92 Package
LP2950 CZ-3.3
1 - Output (3.3V)
2 - Ground (GND)
3 - Input
C12 1uF
C13 10uF
U4 L4931-5.0
L4931CZ-5.0 Pinout TO-92 Package
L4931 CZ-5.0
1 - Output (5.0V)
2 - Ground (GND)
3 - Input
TITLE: 8051_Basic_circuit
Document Number:
REV: B
Date: 7/19/2018 3:51 PM
Sheet: 1/1



TITLE: LCD_Module_w_pullups	
Document Number:	REV: A
Date: 7/19/2018 3:49 PM	Sheet: 1/1

Description	Qty	Sub-section
9V Battery Clip	1	Power Supply
Battery, 9V	1	Power Supply
Capacitor, Electrolytic, 1uF	2	Power Supply
Capacitor, Electrolytic, 10uF	2	Power Supply
IC, Voltage Regulator, 3.3V	1	Power Supply
IC, Voltage Regulator, 5.0V	1	Power Supply
Resistor, 10K, 1/4W	4	RS232/Download
Transistor, BC547	2	RS232/ Download
IC, MAX3232	1	RS232/ Download
Capacitor, Ceramic, 0.1uF	5	RS232/ Download
9-pin DSUB Connector Assembly	1	RS232/ Download
Serial Cable, Male to Female	1	RS232/ Download
Switch, DIP, DPDT	1	RS232/ Download
IC, Microcontroller, AT89C51RC2	1	Microcontroller
Capacitor, Ceramic, 0.1uF	1	Microcontroller
Capacitor, Ceramic, 22pF	2	Microcontroller
Crystal, 11.0592MHz	1	Microcontroller
Crystal, 18.432MHz	1	Microcontroller
Capacitor, Electrolytic, 1uF	1	Microcontroller
Resistor, 10K, 1/4W	9	Microcontroller
Resistor, 150 ohm, 1/4W	4	LED Outputs
LED, Red	1	LED Outputs
LED, Green	1	LED Outputs
LED, Yellow	1	LED Outputs
LED, Amber	1	LED Outputs
Pushbutton Switch	4	Switch Inputs
LCD Module	1	LCD Module
10K or 20K, Variable Resistor	1	LCD Module
SD Card	1	SD Card
SD Card socket assembly	1	SD Card
IC, MP3 decoder, STA013	1	MP3 Decoder
Crystal, 14.7456MHz	1	MP3 Decoder
Capacitor, Ceramic, 22pF	2	MP3 Decoder
Capacitor, Ceramic, 0.1uF	5	MP3 Decoder
Capacitor, Ceramic, 470pF	1	MP3 Decoder
Capacitor, Ceramic, 0.0047uF	1	MP3 Decoder
Resistor, 4.7K, 1/4W	2	MP3 Decoder
Resistor, 2.2 ohm, 1/4W	2	MP3 Decoder
Resistor, 1Meg, 1/4W	1	MP3 Decoder
Resistor, 1K, 1/4W	1	MP3 Decoder
IC, DAC, CS4334	1	DAC
Capacitor, Ceramic, 0.1uF	1	DAC
Resistor, 1K, 1/4W	2	DAC
Resistor, 10K, 1/4W	2	DAC
Resistor, 270K, 1/4W	2	DAC
Resistor, 560 ohm, 1/4W	2	DAC
Capacitor, Tantalum, 3.3uF	2	DAC
Capacitor, Ceramic, 0.0033uF	2	DAC
Jack, 3.5mm, stereo	1	DAC


1
2
3
4
5
6
A
B
C
D
E
VCC
P1.5 (MISO)
SWCOM
LOCK
P1.6 (SCLK)
P1.7 (MOSI)
CARD GPIO (P1.4)
n/c
X1-14
X1-13
X1-12
X1-11
X1-10
X1-9
X1-8
X1-7
X1-6
X1-5
X1-4
X1-3
X1-2
X1-1
SD CARD CONN.
GND
SD CARD
TOP VIEW
SD CARD SOCKET
TOP VIEW
SD CARD PIN
SPI CONNECTION
14 8 - DAT1
13 7 - DO / DAT0 MISO (P1.5)
12 SWITCH COMMON
11 6 - GND GND
10 LOCK SWITCH
9 5 - CLK SCLK (P1.6)
8 n/c
7 4 - Vcc (+3.3V) Vcc (+3.3V)
6 n/c
5 3 - GND GND
4 9 - DAT2
3 2 - DI / CMD MOSI (P1.7)
2 CARD SWITCH
1 1 - nCS / DAT3 nCS (GPIO) (P1.4)
TITLE: SD_Card_Socket
Document Number:
REV:
Date: not saved!
Sheet: 1/1


U5
STA013
VDD1
VDD2
VDD3
VDD4
TESTEN
SCR_INT
SDA
SCL
BIT_EN
SDI
SCKR
DATA_REQ
RESET
SCANEN
SDO
SCKT
LRCKT
OCLK
PVDD
PVSS
FILT
XTO
XTI
VSS1
VSS2
VSS3
VSS4
VSS5
(P1.0) TWI_SDA
(P1.1) TWI_SCL
(P1.3) SPI_BIT_EN
(P1.7) SPI_MOSI
(P1.6) SPI_SCK
(P1.2) GPIO_DATA_REQ
(P3.3) GPIO_RESET
(Values in ( ) are the connections used in my solution)
U6
CS4334
SDATA
SCLK
LRCLK
MCLK
VDD
VSS
LOUT
ROUT
GREEN
RED
BLACK
J2
3.5mm Audio Conn
+3V3
+5V
GND
C14 0.1uF
C15 0.1uF
C16 0.1uF
C17 0.1uF
C18 0.1uF
C19 22pF
C20 22pF
C21 470pF
C22 0.0047uF
C23 0.1uF
C24 0.0033uF
C25 0.0033uF
C26 3.3uF
C27 3.3uF
R19 2.2
R20 2.2
R21 1Meg
R22 1K
R23 270K
R24 270K
R25 560
R26 560
R27 4.7K
R28 4.7K
R29 10K
R30 10K
R31 1K
R32 1K
Y2 14.7456MHz
+9V
IN OUT
GND
U4
L4931-5.0
C12 1uF
C13 10uF
+5V VOLTAGE REGULATOR ALSO SHOWN ON THE BASIC SCHEMATIC
L4931CZ-5.0 Pinout
TO-92 Package
L4931
CZ-5.0
1 2 3
1 - Output (5.0V)
2 - Ground (GND)
3 - Input
TITLE: STA013_CS4334
Document Number:
REV: A
Date: 7/19/2018 4:10 PM
Sheet: 1/1