LOUGHBOROUGH UNIVERSITY

DEPARTMENT OF ELECTRONIC AND ELECTRICAL ENGINEERING

Advanced Laboratory: “MicroMouse Development”

OBJECTIVES

1. To build upon the basic understanding of microcontroller function and programming gained in the Part B lab experiment.

2. To gain practical experience in the design and development of sensor interfaces and their associated real-time software control programs.

COMPLETION DEADLINE: End of Week 11, Semester 2

EQUIPMENT REQUIRED

1 Mouse Chassis kit with pre-assembled motor drive board and microcontroller board
1 ‘Mouse Maze’ in W2.53.

TASKS

1. Design and build sensor/interface PCB to fit on top of mouse chassis.
2. Link motor drive and sensor signals to this interface.
3. Develop program in 8051 Assembler language, C or Modula2, to negotiate the maze provided in W2.53 as quickly and as efficiently as possible. This will involve the development of a control ‘model’ using information supplied with this data pack

RESOURCES

Circuit schematics and other information are attached. Much more information plus useful Web links are to be found on the Department’s Micromouse Web site. Manuals for the microcontroller are installed on the PCs in W2.53 and W2.50.

FURTHER INFORMATION

The microcontroller device is an 87C151SA with 8 kbytes of ROM pre-programmed with a simple loader/monitor. Some useful subroutines which can be called from your program have been included. The ‘mouse’ program is downloaded from a PC Serial COM port, via a plug-in RS-232 link, into the 32 kbyte RAM chip on the board. The operation of this serial link is controlled by the loader, and a terminal emulation program running on the PC. The RAM chip provides both program and data space. How much is used for each is determined by the user’s program. Note that there are an additional 256 bytes of data memory plus Special Function Registers (SFRs) on the microcontroller chip itself. The processor clock runs at 11.0592 MHz.

The 87C151SA is virtually identical to the 87C51FA device, but with updated internal architecture so that it runs about six times faster. We use it because it contains a Programmable Counter Array (PCA) which supports Pulse Width Modulation (PWM) drive of DC motors, and can simplify the processing of feedback signals controlling motor speed. Two useful references to get hold of are the Intel Application notes: **AP-425 Small DC Motor Control** and **AP-440 8XC51FA/FB/FC PCA Cookbook**. Both are down-loadable from an Intel Internet Web site (see the Department’s MicroMouse Web site for this and other useful links.) This Intel MCS51 site provides access to manuals, technical notes and a free development tool called **ApBuilder**. The latter is installed on some machines in W2.50 and W2.53, but you can obtain your own copy via the Internet.

Memory Map for Mk.2 Controller Board

0000-1FFF	Internal ROM programmed with MOUSEMON
2000-2FFF	Decoded chip-select signal available to user
3000-3FFF	Decoded chip-select signal available to user
4000-4FFF	Decoded chip-select signal available to user
5000-5FFF	Decoded chip-select signal available to user
6000-6FFF	Decoded chip-select signal available to user
8000-FFFF	External User Program/Data RAM

The MicroMouse Resident Monitor Program: MOUSEMON

The program ROM area of the microcontroller is pre-programmed with a monitor that allows you to download your assembled programs from a PC via the plug-in serial lead. Program development proceeds as follows:

1. Create a SOURCE code file using Windows Notepad or MS-DOS Edit. Include the statement **$MOD151** right at the start of your source code to tell the assembler that you are using the ‘151 device. This source code consists of program statements in 8051 Assembler language. The filename will be given the extension .ASM. Assemble your program using the Metalink ASM51 Cross-Assembler resident on machines in the micro labs. You now have a down-loadable Intel-Hex format file (.HEX) and a listing file (.LST).

   ASM51 filename <CR>

   Note: ASM51 is run from DOS, either directly or from a DOS Window.
2. Run the Windows terminal emulator on one of the PCs in the mouse lab by clicking on the icon **Terminal**. Set up the baud rate, etc. manually, or **Open** the file mm2. The Com2 port is used.
3. Plug the ‘phone jack plug into the mouse, and switch on a +5 volt supply to the board. Press the mouse reset button and the following should appear in the emulator window:

```
LUT Mobile Robot Monitor Vsn 3.4 ready
(R)eg Dump  e(X)amine  (S)tore Reg  (M)em Dump
(W)ipe Mem  (N)ext  (J)ump  (G)o User  (T)est  (Q)uit
>
```

4. Download your program by selecting **Send Text File** from the **Transfer** menu item. Make sure you click on the .HEX version of your code!
5. When program loading is complete you can start it running by typing **G**. This command assumes that the program starts at location 8012H. You will want to disconnect the serial lead now and place the mouse in the maze. For this reason you should include in your program a delay of say 10 seconds before the wheels start turning!
6. To stop the mouse press the reset button, plug in the serial lead and press reset again. The opening message should appear on the PC.

Drive Specification

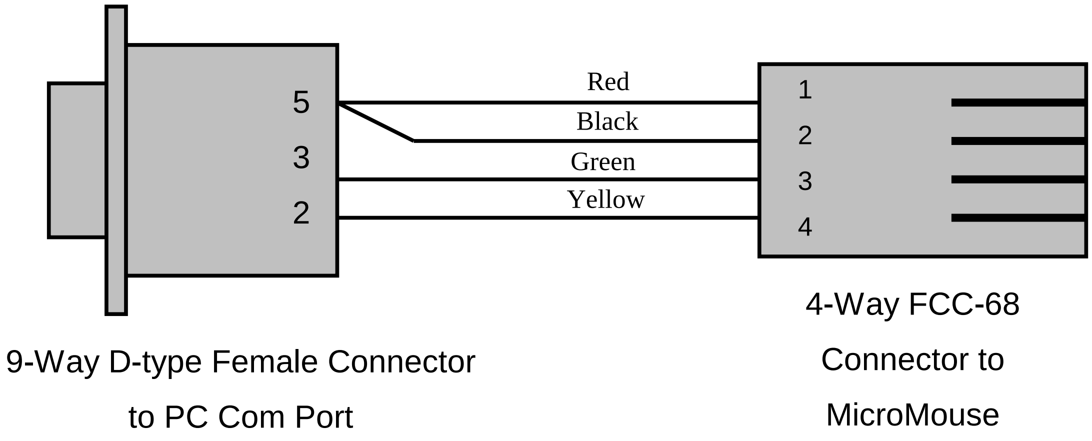
Two Mabuchi RC-280SA-20120 DC motors each giving 29 g.cm torque at 6400 rpm. Each takes 0.57 A of current from a 6 V supply at max efficiency. Stall torque from a 6 V supply is 175 g.cm at a stall current of 2.85 A.

An 8-slot slotted-disc tachometer is mounted on each motor shaft with LED/phototransistors appropriately mounted to provide a TTL level (open-collector) signal consisting of eight pulses for each revolution. This corresponds to 0.8 mm of linear movement at the wheels per pulse. The gearbox provides a 16 : 1 reduction.

The motor drive board contains two independent H-Bridge circuits, one for each motor. All the control signals have TTL levels. Connect these to appropriate ports on the microcontroller board. Use a PWM signal on the Enable inputs to drive each motor at varying speeds. The A and B inputs provide four permutations: Drive Forwards/Backwards, Free-wheel and Fast Stop (Brake). However, this chassis contains a worm and gear drive, *which cannot work backwards*. Hence if the motor power is turned off, the (considerable) momentum of the chassis simply causes the wheels to lock-up and skid because a worm gear can drive a pinion, but not the other way around. To avoid skidding you must ‘ramp’ the speed down when you wish the mouse to stop: it is not advisable to use the Free-wheel or Fast Stop options.

Power Supply

You will be provided with four AA size Ni-Cd rechargeable batteries. They need to be removed from the holder for charging. The charger in W2.53 performs a fast discharge/charge cycle followed by trickle charge. The discharge action prevents formation of Ni-Cd ‘memory’. We have found that this battery pack will drive both the motors and the electronics for up to an hour between charges (depending on how many sensors you use), although falling voltage may cause the motors to slow down. A control algorithm can compensate for the latter effect by altering the PWM signal in response to tacho feedback. The MCU will continue running with less than 4 volts! It is recommended that you link the battery pack to your sensor board with an in-line snap connector of the type used by model car racers. Include an on/off switch on the board to reduce wear and tear on the connector. Note that when you exit a user program with Q(uit), the microcontroller drops into low-power mode until the next interrupt.


Red
Black
Green
Yellow
5
3
2
1
2
3
4
9-Way D-type Female Connector to PC Com Port
4-Way FCC-68 Connector to MicroMouse

LUT MicroMouse Controller to PC Connector Cable

Using the Thurlby LA4800 Logic Analyser

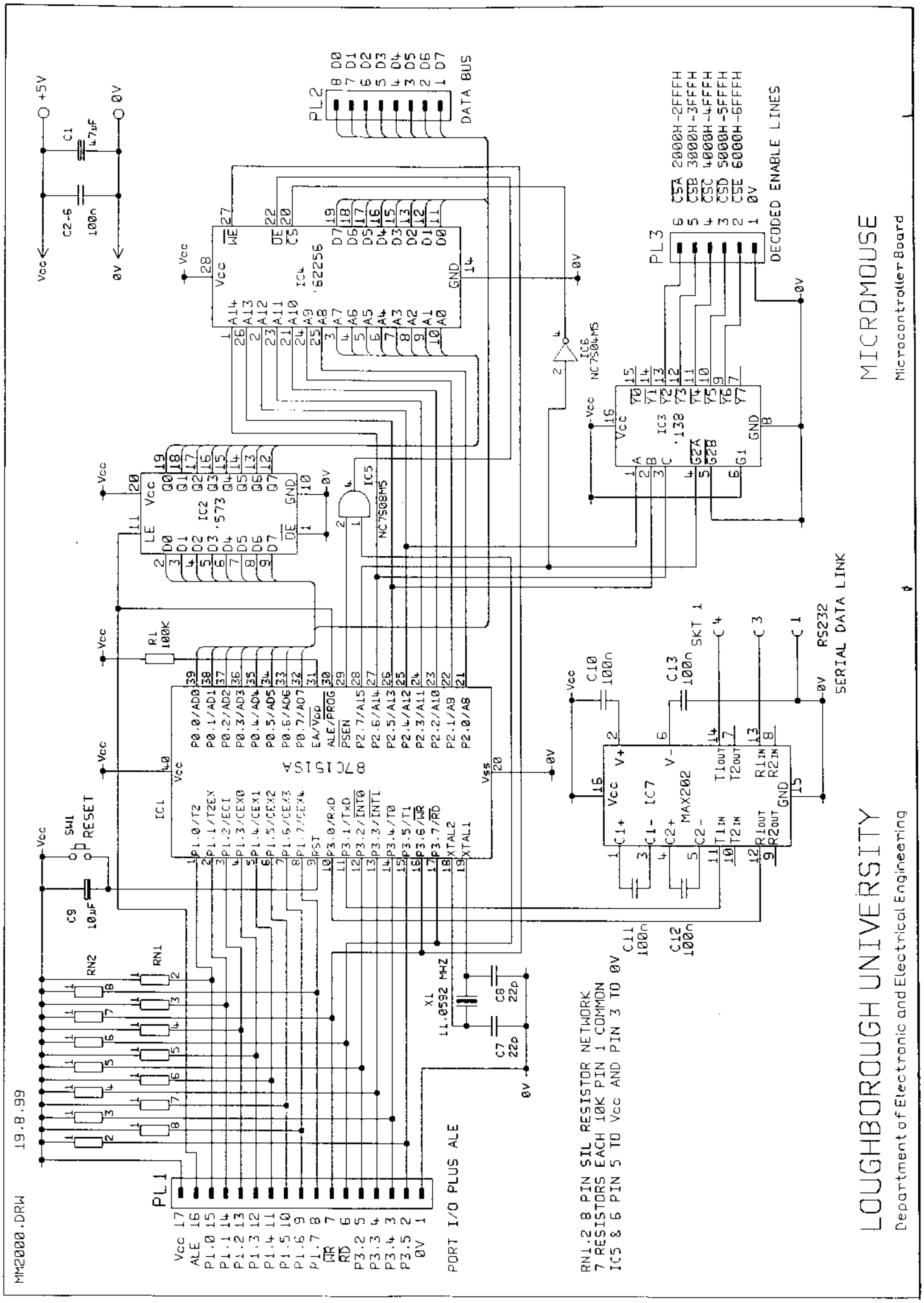
The LA4800 Logic Analyser has a Disassembler Pod for the 8051-series of microcontrollers which allows you to see what is happening as your program runs. The analyser displays in Assembler language form each line of code as it executes, so you should be able to spot unexpected jumps for instance. It does not show internal status of registers: you will need the Single-Step capability of MOUSEMON to do that.

Connect the analyser pod to the microcontroller chip using the 40-pin ‘crocodile clip’. The red stripe on the ribbon cable signifies pin 1, so make sure it’s at the top end of the device. Take care clipping on the connector; don’t lever the chip out of its socket. It is important that all 40 pins connect, so some ‘wiggling’ may be necessary. Switch on the Analyser and the mouse board. Download your program from the PC into mouse memory.

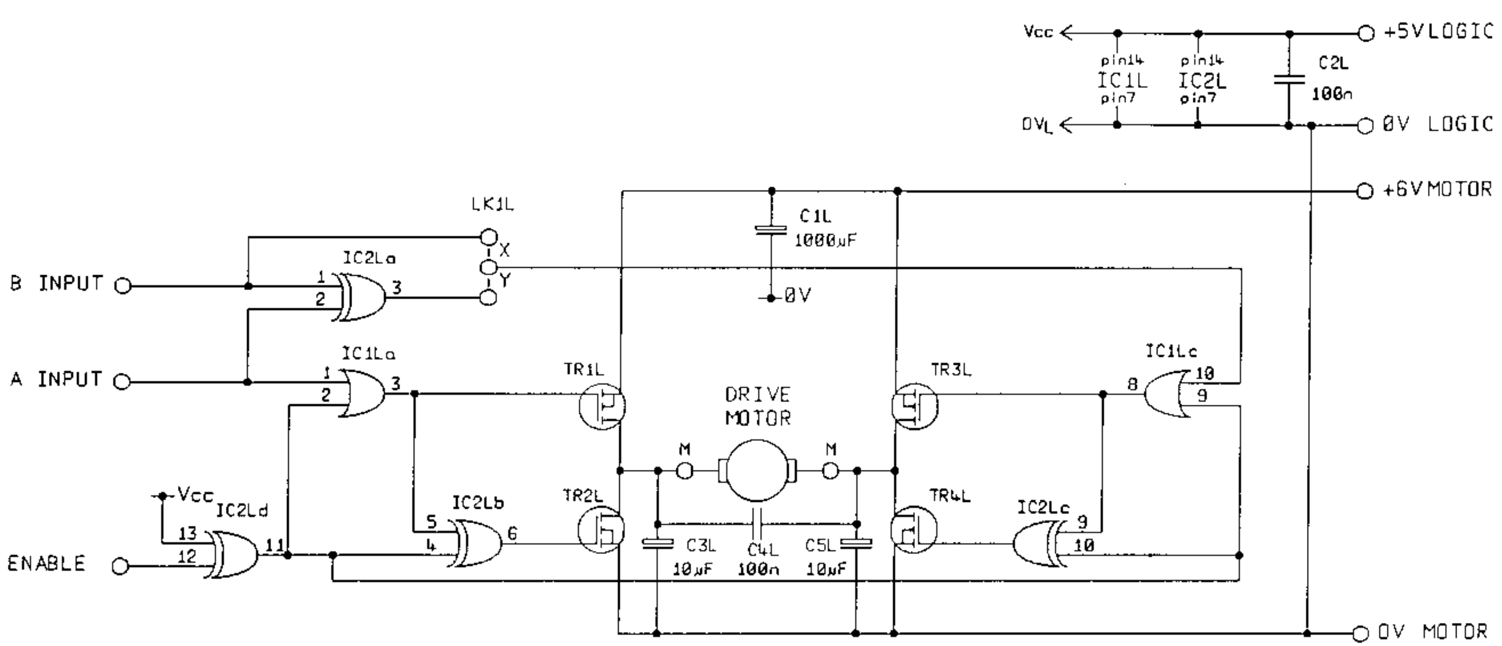
Select CONFIRM Disassembler, and from the Main Menu, Select (4) Trigger. User program memory begins at 8000H, so type 80xx into the Trig Word Address. You want the analyser to start ‘capturing’ program execution as soon as you jump into it; the start address is likely to be in that range, usually at 8012H. TAB to Trigger Position and Select START. Press MENU.

Select (7) Disassembler. Press RUN, then begin capture by selecting SINGLE. The analyser will now wait for a Trigger Address to appear on the Address bus. Run your mouse program by typing on the PC keyboard ‘G’ or ‘xxxxJ’. The program steps should appear on the analyser screen as they execute until the capture memory is full. Select GOTO TRG to display from the trigger word and use INCREMENT and DECREMENT to scroll through the memory. You will probably be seeing all the intermediate steps such as op-code and operand fetches. TAB to EXPANDED MODE and Select OFF, so that you see only the assembler-style ‘listing’.

WGM Rev 3.4 5/2000 MouseLab.doc


MICROMOUSE
Microcontroller Board
LOUGHBOROUGH UNIVERSITY
Department of Electronic and Electrical Engineering
MM2000.DRW
19.8.99
+5V
0V
C1 4.7uF
C2-6 100n
Vcc
PL2
DATA BUS
D0 D1 D2 D3 D4 D5 D6 D7
IC4 '62256
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14
WE OE CS GND
IC6 NC7S04M5
IC5 NC7S08M5
IC2 '573
D0 D1 D2 D3 D4 D5 D6 D7 Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7 LE OE
IC3 '138
A B C G2A G2B G1 Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
PL3
DECODED ENABLE LINES
CSA 2000H-2FFFH
CSB 3000H-3FFFH
CSC 4000H-4FFFH
CSD 5000H-5FFFH
CSE 6000H-6FFFH
0V
R1 100K
IC1 87C51SA
P0.0/AD0 P0.1/AD1 P0.2/AD2 P0.3/AD3 P0.4/AD4 P0.5/AD5 P0.6/AD6 P0.7/AD7
EA/Vpp ALE/PROG PSEN
P2.7/A15 P2.6/A14 P2.5/A13 P2.4/A12 P2.3/A11 P2.2/A10 P2.1/A9 P2.0/A8
Vss
P1.0/T2 P1.1/T2EX P1.2/ECI P1.3/CEX0 P1.4/CEX1 P1.5/CEX2 P1.6/CEX3 P1.7/CEX4 RST
P3.0/RxD P3.1/TxD P3.2/INT0 P3.3/INT1 P3.4/T0 P3.5/T1 P3.6/WR P3.7/RD XTAL2 XTAL1
SW1 RESET
C9 10uF
RN1 RN2
X1 11.0592 MHZ
C7 22p
C8 22p
IC7 MAX202
C1+ C1- C2+ C2- V+ V- T1IN T2IN T1OUT T2OUT R1IN R2IN R1OUT R2OUT GND Vcc
C10 100n
C11 100n
C12 100n
C13 100n
SKT 1
RS232
SERIAL DATA LINK
PL1
PORT I/O PLUS ALE
Vcc 17
ALE 16
P1.0 15
P1.1 14
P1.2 13
P1.3 12
P1.4 11
P1.5 10
P1.6 9
P1.7 8
WR 7
RD 6
P3.2 5
P3.3 4
P3.4 3
P3.5 2
0V 1
RN1,2 8 PIN SIL RESISTOR NETWORK
7 RESISTORS EACH 10K PIN 1 COMMON
IC5 & 6 PIN 5 TO Vcc AND PIN 3 TO 0V

MOUSECON.DRW 26.8.98



CIRCUIT SHOWN FOR ONE MOTOR ONLY PRINTED CIRCUIT BOARD CONTAINS TWO IDENTICAL CIRCUITS FOR LEFT HAND MOTOR COMPONENT IDENTIFICATION IS 'L', 'R' FOR RIGHT HAND MOTOR

TRUTH TABLE LK1 in Pos. X

A	B	En	
0	1	1	Forwards
1	0	1	Backwards
1	1	1	Fast Stop
0	0	1	Fast Stop
X	X	0	Free Run

TRUTH TABLE LK1 in Pos. Y

A	B	En	
0	1	1	Forwards
1	1	1	Backwards
X	0	1	Fast Stop
X	X	0	Free Run

Enable can be pulsed for variable speed
X = EITHER 1 or 0
Forwards and backwards directions are relative to the motors being at the rear of the unit

LOUGHBOROUGH UNIVERSITY
Department of Electronic and Electrical Engineering

MICROMOUSE
Motor Control Board