IEEE Micro Mouse using the 87C751 microcontroller — AN443

Author: Tracy Ching

DESCRIPTION

Micro Mouse is an IEEE contest first proposed by the author of IEEE Spectrum in 1977. It consists of an autonomous robot known as a "mouse" which navigates through a maze of 256 two-inch-high, seven-inch squares in a 16×16 arrangement. The robot is self powered and has no knowledge of the maze configuration prior to releasing it in the maze. The first time it is released into the maze, its prime objective is to find a path from the starting square which is located in a corner of the maze to the destination square which can be located in the center or a different corner depending on competition level. The destination square for the advanced level can be found only by using a smart algorithm which will make the mouse gravitate towards the center without becoming lost. The destination square for a novice contest can be found by using a wall hugging algorithm. The analogy for this algorithm is to imagine a blind person holding their right hand out against a wall and following the walls until they reach the destination. A maximum of ten runs or 15 minutes is given to each mouse. Leaving the starting square constitutes one run. The mouse with the fastest run time from the starting square to the destination square wins.

The contest is held with different competition levels for novices and advanced. The novice level is typically held for college students trying to exercise newly acquired hardware and software skills, whereas the advanced level encompasses international talent and may require the implementation of a proportional integral derivative (PID) controller in software to utilize commutated or brushless DC motors and complex navigation algorithms.

DESIGN OBJECTIVES

Several design objectives were as follows: minimize weight, minimize part size and count, minimize power consumption which allows the use of smaller and lighter batteries, minimize cost and maximize speed.

The main objective was to keep the total weight at a minimum to obtain the fastest acceleration from the stepper motors and to reduce wheel slippage during deceleration and turning. The objectives are inter-related such that changing one will affect the other. Hence, having a low part count means lighter weight, faster acceleration and lower cost.

A typical mouse may consist of a microcontroller with supporting memory and GLU logic to interface the motor controllers and sensors. The 87C751 is suitable for this application with its small size. The need for external RAM is eliminated by using the internal RAM to store minimum maze information suitable for the novice level. Nickel cadmium batteries are used because of the cost, size and power density obtainable versus other battery types. Nickel metal hydride was not available during development but would be a good choice over nickel cadmium batteries.

The 87C751 Microcontroller

The 87C751 is an 8-bit microcontroller based on the 8051 microcontroller family. It is code compatible with the exception of the MOVX, LJMP, and LCALL instructions. The MOVX instruction and external memory accesses are not supported. LJMP and LCALL instructions are not needed since AJMP and ACALL can reach the entire program memory range (2k bytes) of the 751. The 87C751 contains a $2k \times 8$ EPROM, a 64×8 RAM, 19 I/O lines, a 16 bit auto-reload counter/timer, a fixed rate timer, a five source fixed priority interrupt structure, a bidirectional Inter-Integrated circuit (I^2C) bus interface, and an internal oscillator. The 87C751 comes in an erasable quartz package (87C751), one time programmable (87C751), and mask ROM (83C751).

HARDWARE DESCRIPTION

Figure 1 is a schematic diagram of the mouse. The stepper motors are 4 volts 0.95 amperes per coil giving about 14 oz-inches of torque. Each motor is driven by an Allegro UCN-5804B unipolar stepper motor translator/driver which contains the sequencing logic and high current darlington outputs. The sequencing logic only requires clock, direction of rotation, and output enable signals from the 87C751 which relieves the chore of having to cycle through a sequencing table for both motors therefore reducing code size. Fast recovery diodes are used to protect the darlington outputs from negative voltage due to motor winding flyback.

The infrared (IR) emitters are Optek OP-240A. The sensors are Optek OPL-560-OC which have a built-in light amplifier and TTL open collector output. Each sensor bank is enabled via a high side P-channel MOSFET. A 74LS04 (U4) is used to drive the P-channel MOSFET. Data from each bank of eight IR sensors is fed into port 3. By using open collector output sensors, the need for latching the data using a 3-State buffer is eliminated. Port pins P0.0 and P0.1 are used for enabling either the left or right sensor bank via the 74LS04 (U4) and the MOSFETs. Each sensor bank hangs over the two inch high walls whereby the light emitted from the OP-240 is reflected into the OPL-560-OC if a wall is present. Two sensor pairs in the middle of the array are typically sensing the presence of a wall and the remaining sensors are used for guidance to keep the mouse running parallel to the walls.

A 74LS573 connected to port 3 which latched data into eight LEDs was used in the debugging process. P1.7 was used on the latch signal for the 74LS573. Since the eight LEDs and latch were not needed for the final product, they were removed thus reducing weight and battery drain. LEDs D1 and D2 were also used in the debugging process and are currently used for visual feedback when selecting options during operation.

Power for the digital circuitry is fed by an 8.4 volt NiCAD battery pack, packaged in a 9 volt battery case, via a 7805 regulator. The circuit can run constantly with the sensors taking "snapshots" of the walls intermittently for at least 30 minutes satisfying the 15 minute maximum requirement. Power for the motors is fed by four "A" size NiCAD cells which have enough energy to run the motors for 15 minutes before becoming useless at about 1 volt per cell.

TASK PRIORITIES

Several tasks take place during program execution which are as follows in order of importance: pulsing the stepper motors according to a velocity profile table, gathering sensor data, deciding whether to accelerate, decelerate, turn left or right.

In-line coding is used to avoid calling subroutines that cause the program counter to be pushed onto the stack and use valuable RAM for the turn decisions. Interrupt subroutines are an exception.

Interrupt timer 0 (T0) vectors to the routine which supplies a pulse to the stepper motor drivers. This is the most important task because the stepper motors require a smooth train of pulses in order to prevent jerky or sporadic motions. Thus, T0 must have the highest priority and must not be interrupted. Although timer 0 is a 16 bit timer, only eight bits are used with the higher byte set to FFH. T0 takes care of pulsing the left and right motor drivers at different times by using two external registers which are used as prescalers. Each motor can be assigned a different prescale value via the assigned register in order to step each motor at a different step rate.

IEEE Micro Mouse using the 87C751 microcontroller AN443

The velocity profile table for T0 was designed using a spreadsheet program with visual graphing. This aided the ability to derive an exponential table for the fastest stepper motor acceleration using qualitative observations.

The first part of the in-line code contains the routine to accelerate the mouse from a stopped position. A pointer for each motor is incremented until the end of the velocity profile table is reached. During acceleration, the left and right sensors are strobed and stored after each step caused by T0.

The routine that strobes the sensors for wall data returns several results through the use of flags. The routine first stores the sensor data in registers. This information is used to determine if the sensor array or mouse is too far right, left, or aligned in a square and to set the corresponding flags. It also returns the presence of a front wall using the innermost sensor pairs. The deceleration routine is similar to the acceleration routine except the pointers decrement through the velocity profile table skipping a few values each time. Deceleration uses less steps than acceleration.

The routine which decides whether to continue acceleration, deceleration, or turn left or right keeps track of step count or position of the mouse within a square in order to store wall information at the proper time. After the previous routines have stored the data for the front, left, and right walls, a decision is obtained. If the mouse is not in a back-up mode, the wall information, status of the back-up flag and left/right algorithm flag forms an offset byte. If the mouse is in a back-up mode, the decision from above plus the previous decision on the stack is used to form the offset byte. This offset is stored in the accumulator. The decision which contains the direction to turn is obtained using the MOVC instruction which points to the table using the data pointer. The logic table is shown in Tables 1 and 2 below.

External interrupt 0 (INT0) vectors to the routine which brings the mouse to a halt. INT0 subroutine disables T0, sets output enable high on the stepper motor drivers, and sets the return address for the program counter to 0000H. This causes the mouse to restart program execution without losing the internal RAM.

Table 1. Logic Table: Non-Back-Up Mode

(1)	(2)	(3)	(4)
R	NONE	R	push R onto stack
R	F	R	push R onto stack
R	R	S	push S onto stack
R	R, F	L	push L onto stack
R	L	R	push R onto stack
R	L, F	R	push R onto stack
R	L, R	S	ignore
R	L, R, F	180	turn on B-U flag
L	NONE	L	push L onto stack
L	F	L	push L onto stack
L	R	L	push L onto stack
L	R, F	L	push L onto stack
L	L	S	push S onto stack
L	L, F	R	push R onto stack
L	L, R	S	ignore
L	L, R, F	180	turn on B-U flag

NOTES:
Abbreviations used:
R = right
L = left
F = front
S = straight
B-U = back-up flag
stack = RAM used for storing decisions (not for the program counter)

Non back-up mode:
(1) wall hugging algorithm (user selected)
(2) walls surrounding present square
(3) direction to turn
(4) operations to perform

Table 2. Logic Table: Back-Up Mode

(1)	(2)	(3)	(4)
R	R	R	push S onto stack, clear B-U
R	L	L	pop stack only
R	S	S	push L onto stack, clear B-U
L	R	R	pop stack only
L	L	L	push S onto stack, clear B-U
L	S	S	push R onto stack, clear B-U
S	R	R	push L onto stack, clear B-U
S	L	L	push R onto stack, clear B-U
S	S	S	pop stack only

NOTES:
Abbreviations used:
R = right
L = left
F = front
S = straight
B-U = back-up flag
stack = RAM used for storing decisions (not for the program counter)

Back-up mode:
(1) previous decision from top of stack
(2) decision from the above table for this current square
(3) direction to turn (should be equal to (2)
(4) operations to perform (all operations require lowering the stack by one before pushing data)

IEEE Micro Mouse using the 87C751 microcontroller AN443

OPERATION

Reset

True reset is accomplished only during power on. After completing the maze, the operator brings the mouse to a stop using the start/stop switch which effectively disables the motors and resets the program counter to 0000H, restarting the mouse's program with the exception that the RAM contains vital maze information.

Starting/Stopping

After power-up, the mouse remains idle with the motors and sensors disabled, awaiting an interrupt from switch SW1. The first time the switch is pressed, the mouse will start. The second time the switch is pressed, the mouse will stop and the cycle repeats.

Selecting right or left wall hugging

After power on reset, the program defaults to right wall hugging. Pressing switch SW2 will cause the mouse to follow the left walls. If it is pressed again, it will follow the right walls—toggling between the left and right wall hugging algorithm each time it is pressed.

Increasing motor speed

After completing the first run, pressing switch SW2 causes the timer value to increase by an index of one, increasing the speed of the motors. This allows the mouse to be run at increasing speeds each run in order to attain the fastest possible time before crashing into a wall or incurring a condition in the stepper motors called “pull-out”. Pull-out is a condition wherein the fields are changing in the coils faster than the rotor can maintain synchronism with the coils, causing the rotor to stall.

SOFTWARE LIMITATIONS

The 87C751 has 64 bytes of RAM. The program requires 31 bytes of RAM leaving 33 bytes for storing decisions. One hundred thirty two decisions can be stored in 33 bytes of RAM since four decisions are stored per byte. Although there are 16 times 16 squares with possibly 256 decisions to be made, this condition is not very probable since every square typically is not made into a turning point. Long straight ways are used as well as turning points. The probability that the RAM will fill to the maximum capacity is very slim in the novice level where there are typically 50 decisions to be made. Hence, RAM which can hold 132 decisions is adequate but not infallible.

FUTURE ENHANCEMENTS

Although the 87C751 has minimal RAM, I^2C RAM could be easily added by switching the MOSFET drivers using the LED port pins and placing the I^2C RAM on port pins P0.0 and P0.1. This would allow the implementation of a full mapping algorithm thus allowing entry to the advanced class. The code size for the I^2C routines and recursive algorithm to find the center of the maze would add about 800 bytes more of code. Since some of the code for the novice class would be eliminated, it would bring the total code size to less than 2k bytes. A few maze solving algorithms have been implemented successfully on other mice. These are the flooding or Bellman's algorithm, backtracking algorithm and others. The first two algorithms are recursive, thus the code sizes are quite small.

Another added feature would be the ability to execute a rounded turn rather than pivot. This requires a more complex navigation scheme and velocity profiler to make the motors turn at differing speeds in order to make the rounded turn.

In addition to the enhancements aforementioned, methods to track a wall through the use of an A/D converter which measures the reflectivity strength from infrared sensors pointing at the sides of the walls can also be implemented on the I^2C bus. This would eliminate the array of sensors hanging in front of the mouse on top of the walls thus decreasing the weight.

CONCLUSION

The 87C751 microcontroller provides the required computing resources and I/O ports necessary to control a robotics device known as a “Micro Mouse”. The I^2C interface allows for an abundance of variations on this robotics device.

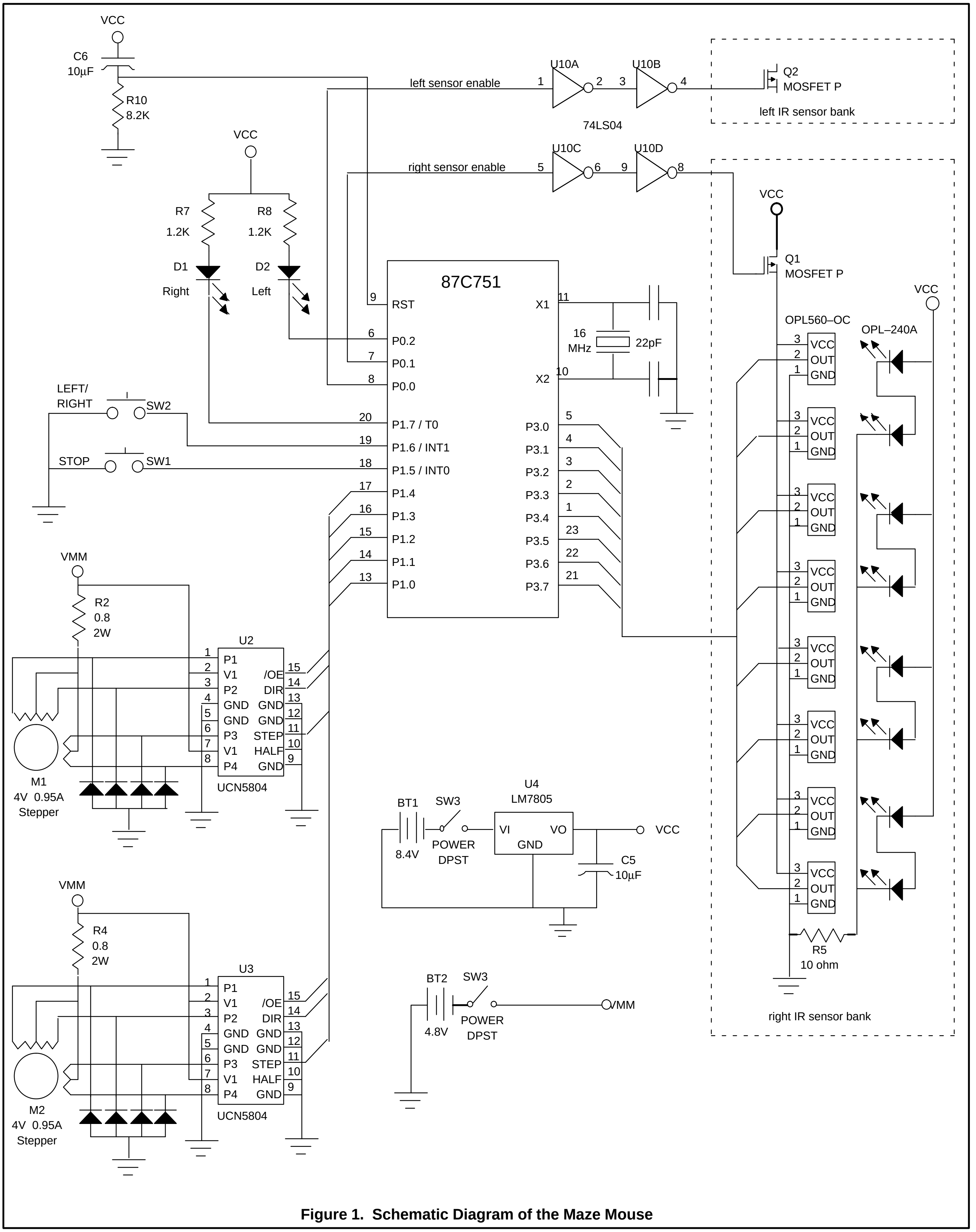

Figure 1. Schematic Diagram of the Maze Mouse

IEEE Micro Mouse using the 87C751 microcontroller AN443

```
MCS-51 MACRO ASSEMBLER    751MAIN                                          04/16/92   PAGE    1

LOC  OBJ            LINE     SOURCE
                       1     ;*******************************************************
                       2     ; 87C751 Micro Mouse version 4.0 started 9-27-91        *
                       3     ;                   version 4.3 finished 4-16-92        *
                       4     ; Algorithms and programming by Tracy Ching             *
                       5     ; Some symbols commented out for use as in-line coding  *
                       6     ; rather than being used as subroutines (S.R.)         *
                       7     ;*******************************************************
                       8
                       9 +1  $include        (751.equ)
                =1    10     ;These are all RAM addresses
                =1    11     ;equate table of constants
                =1    12     ;ind_reg        equ     00      ;used for ind addr of regs
                =1    13     ;map_org        equ     01      ;pointer for storing map of maze
                =1    14     ;               equ     02      ;decide storage-local,
                =1    15     ;               equ     03      ;do180, turn90 use it
                =1    16     ;               equ     04      ;snapshot R wall storage
                =1    17     ;               equ     05      ;snapshot L wall storage
                =1    18     ;               equ     06      ;store_val, decel, int 1, gen purp
                =1    19     ;               equ     07      ;pause setting, gen purp
                =1    20
  0008          =1    21     step_count      equ     08h     ;step count used by decide
  0009          =1    22     temp_reg1       equ     09h     ;
  000A          =1    23     ls373           equ     0ah     ;storage for 74LS373 debug data leds
  000B          =1    24     count_fw        equ     0bh     ;count for wall, step count
  000C          =1    25     l_timr          equ     0ch     ;value to be used in the isr
  000D          =1    26     r_timr          equ     0dh     ;
  000E          =1    27     l_ptr           equ     0eh     ;points at accel table
  000F          =1    28     r_ptr           equ     0fh     ;
  0010          =1    29     map_offset      equ     10h     ;points to the specific two bits in map ptr
                =1    30     ;       equ     11h
                =1    31     ;       equ     12h
                =1    32     ;       equ     13h
                =1    33     ;       equ     14h
                =1    34     ;       equ     15h
                =1    35     ;       equ     16h
                =1    36     ;       equ     17h
                =1    37
                =1    38     ;PCON   EQU     87H
                =1    39     ;TA     EQU     0C7H
  008B          =1    40     rtl     equ     8bh
  008D          =1    41     rth     equ     8dh
                =1    42
                =1    43
                =1    44     ;flag declarations, they are erased after every restart
  0020          =1    45     ss_bits         equ     20h     ;sens flag reg
  0000          =1    46     too_r           bit     20h.0   ;too close to right wall
  0001          =1    47     too_l           bit     20h.1   ;              left wall
  0002          =1    48     r_wall          bit     20h.2   ;right wall present, snapshot storage
  0003          =1    49     l_wall          bit     20h.3   ;left wall present, snapshot storage
  0004          =1    50     f_wall          bit     20h.4   ;front wall present, snapshot storage
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                             04/16/92   PAGE    2

LOC  OBJ            LINE     SOURCE
  0005          =1    51     aligned         bit     20h.5   ;sensors detect OK straightness
  0006          =1    52     far_r           bit     20h.6   ;used to sync step count
  0007          =1    53     far_l           bit     20h.7   ;ditto - left
                =1    54
  0008          =1    55     r_turn          bit     21h.0   ;decide sets this for turn S.R.
  0009          =1    56     l_turn          bit     21h.1   ;ditto
  000A          =1    57     s_s_int         bit     21h.2   ;start stop bit
  000B          =1    58     time_int        bit     21h.3   ;timer 0 int has occurred
  000C          =1    59     r_int           bit     21h.4   ;r motor was stepped
  000D          =1    60     l_int           bit     21h.5   ;l mot
  000E          =1    61     slow_r          bit     21h.6   ;make int_0 incr timer flag
  000F          =1    62     slow_l          bit     21h.7   ;ditto left
                =1    63
  0010          =1    64     get_out         bit     22h.0   ;used to flag accel to end so decel can do
  0011          =1    65     r_decide        bit     22h.1   ;r wall present, decide storage
  0012          =1    66     l_decide        bit     22h.2   ;l wall present, decide storage
  0013          =1    67     f_decide        bit     22h.3   ;f wall present, decide storage
  0014          =1    68     make_180        bit     22h.4   ;decide says do 180
  0015          =1    69     prev_r          bit     22h.5   ;decide uses for detect wall transition
  0016          =1    70     prev_l          bit     22h.6   ;ditto left
  0017          =1    71     cw180           bit     22h.7   ;toggle dir of 180
                =1    72
  0018          =1    73     det_L2H         bit     23h.0   ;used to detect the low to high wall
  0019          =1    74     look4f          bit     23h.1   ;skips to look for front wall
  001A          =1    75     back_up         bit     23h.2   ;tells decide that it is backing up
  001B          =1    76     look            bit     23h.3   ;start looking at wall
  001C          =1    77     count_en        bit     23h.4   ;enables step counting
  001D          =1    78     skip_decide     bit     23h.5   ;TEMPORARY (testing only)
  001E          =1    79     genp2           bit     23h.6   ;chk R or L sens flag
  001F          =1    80     genp1           bit     23h.7   ;gen purpose, watch for conflicts betwx S.R.s
                =1    81
                =1    82
                =1    83
                =1    84     ;these are the flags that don't get erased after restarting
  0020          =1    85     done            bit     24h.0   ;tells the prog that it has gone thru
  0021          =1    86     l_r_bit         bit     24h.1   ;hug left or right bit, set=left, clr=right
  0022          =1    87     temp_bit1       bit     24h.2   ;local bit var
                =1    88
                =1    89
                =1    90     ;hardware definitions
  0082          =1    91     l_led           bit     p0.2
  0097          =1    92     r_led           bit     p1.7
  0090          =1    93     mot_en          bit     p1.0
  0081          =1    94     r_sens          bit     p0.1
  0080          =1    95     l_sens          bit     p0.0
  0092          =1    96     r_step          bit     p1.2
  0091          =1    97     r_dir           bit     p1.1
  0094          =1    98     l_step          bit     p1.4
  0093          =1    99     l_dir           bit     p1.3
  0095          =1   100     s_s_sw          bit     p1.5
  0096          =1   101     l_r_sw          bit     p1.6
  00B0          =1   102     sensors         equ     p3
                     103 +1  $include        (extrn751.tab)
                =1   104     ;These are all of the mouse constants.
                =1   105
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                          04/16/92   PAGE    3

LOC  OBJ            LINE     SOURCE
                =1           extrn number    (timer_reload)
                =1           extrn number    (pivot_speed)
                =1           extrn number    (acc_steps)
                =1           extrn number    (half_way)
                =1           extrn number    (hold_val)
                =1           extrn number    (dec_acc)
                =1           extrn number    (decel_steps)
                =1           extrn number    (decel_decr)
                =1           extrn number    (steps90)
                =1           extrn number    (half_90)
                =1           extrn number    (look90)
                =1           extrn number    (steps180)
                =1           extrn number    (half_180)
                =1           extrn number    (look180)
                =1           extrn number    (map_org_addr)
                =1           extrn number    (sens_pat)
                =1           extrn number    (pause_val)
                =1           extrn number    (w2nw_cnt)
                =1           extrn number    (chk4fr)

0000                                         org     00
0000 0115                                    ajmp    main
0003                                         org     03
0003 618D                                    ajmp    int_0
000B                                         org     0bh
000B 6123                                    ajmp    tim_0
0013                                         org     13h
0013 6164                                    ajmp    int_1


                             ;*********************************************************************
                             ; This section initializes the registers.  Note that the maze info   *
                             ; is not cleared.                                                    *
                             ;*********************************************************************

0015 C220                    main:           clr     done                     ;start intial run
0017 758B00   F                              mov     rtl,#timer_reload        ;reload value for 751
001A 758DFF                                  mov     rth,#0ffh                ;

001D D281                    main_1:         setb    r_sens
001F D280                                    setb    l_sens
0021 43907F                                  orl     p1,#7fh         ;lines high
0024 752000                                  mov     20h,#0          ;clear all flags except for
0027 752100                                  mov     21h,#0          ;   location 24h bits
002A 752200                                  mov     22h,#0
002D 752300                                  mov     23h,#0
0030 7810                                    mov     r0,#10h         ;clear out 10h to 0h ram
0032 7600                    main_0:         mov     @r0,#0          ;    clears the regs
0034 D8FC                                    djnz    r0,main_0
                                     ;initialize values
0036 758125                                  mov     sp,#25h         ;start of STACK
0039 7900     F                              mov     r1,#map_org_addr;start of mapping memory
003B 75A887                                  mov     ie,#10000111b   ;enable ints
003E 300AFD                                  jnb     s_s_int,$       ;stay here until start pressed

```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                04/16/92   PAGE    4

LOC  OBJ           SOURCE

                   ;**********************************************************************
                   ; This section accelerates the mouse thru the velocity table.       *
                   ; The table value is lowered if the mouse is too close to a wall.   *
                   ;**********************************************************************

                   accel:          ;loads acc_table and goes from there
0041 71B7                          acall   snapshot        ;
0043 A202                          mov     c,r_wall        ;  check the walls
0045 9215                          mov     prev_r,c
0047 A203                          mov     c,l_wall        ;  for both sides
0049 9216                          mov     prev_l,c
004B 750841                        mov     step_count,#65  ;  start at 75stps, sens mid way
004E 750E00        acc_aa:         mov     l_ptr,#0
0051 750F00                        mov     r_ptr,#0        ;clear offsets
0054 750DFE                        mov     r_timr,#0feh    ;even up the timers to step
0057 750CFE                        mov     l_timr,#0feh    ;   evenly
005A D28C                          setb    tr0             ;enable timer int

005C 300BFD        acc_0:          jnb     time_int,$      ;stay here until int happens
005F C20B                          clr     time_int
0061 71B7                          acall   snapshot        ;get status of too_l too_r

                                   ;right routine
0063 300C21                        jnb     r_int,acc_1     ;skip if right hasn't been stepped yet
0066 C20C                          clr     r_int           ;ackn r int
0068 E50F                          mov     a,r_ptr         ;check to see if at end of acc
006A B40002    F                   cjne    a,#acc_steps,acc_1c
006D 0171                          ajmp    acc_1b
006F 050F          acc_1c:         inc     r_ptr           ;    point to next accel value
0071 300113        acc_1b:         jnb     too_l,acc_1     ;if not too left then fall thru
0074 C3                            clr     c               ;subtr halfway from ptr
0075 E50F                          mov     a,r_ptr
0077 9400      F                   subb    a,#half_way     ;C set if ptr < half_way
0079 E50D                          mov     a,r_timr
007B 4006                          jc      acc_1a
007D 2400      F                   add     a,#hold_val     ;   slow down even more
007F F50D                          mov     r_timr,a        ;load it back into timer decel value
0081 0187                          ajmp    acc_1           ;get out
0083 2400      F   acc_1a:         add     a,#dec_acc      ;dec_acc used when going fast
0085 F50D                          mov     r_timr,a        ;load it back into timer decel value

                                   ;left routine
0087 300D21        acc_1:          jnb     l_int,acc_2     ;if L hasn't been stepped, skip
008A C20D                          clr     l_int           ;ackn l int
008C E50E                          mov     a,l_ptr         ;check to see if at end of acc
008E B40002    F                   cjne    a,#acc_steps,acc_2c
0091 0195                          ajmp    acc_2b
0093 050E          acc_2c:         inc     l_ptr           ;    point to next accel value
0095 300013        acc_2b:         jnb     too_r,acc_2     ;if not too right then fall thru
0098 C3                            clr     c               ;subtr halfway from ptr
0099 E50E                          mov     a,l_ptr
009B 9400      F                   subb    a,#half_way     ;C set if ptr < half_way
009D E50C                          mov     a,l_timr
009F 4006                          jc      acc_2a
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                             04/16/92   PAGE    5

LOC  OBJ            LINE     SOURCE
00A1 2400     F      216                     add     a,#hold_val     ;   slow down even more
00A3 F50C            217                     mov     l_timr,a        ;load it back into timer decel value
00A5 01AB            218                     ajmp    acc_2           ;get out
00A7 2400     F      219     acc_2a:         add     a,#dec_acc      ;dec_acc used when going fast
00A9 F50C            220                     mov     l_timr,a        ;load it back into timer decel value
                     221
                     222     acc_2:
                     223
                     224
                     225     ;*********************************************************************
                     226     ; This section is the "decision-maker".  It makes the decision to   *
                     227     ; quit acceleration, turn left or right, make a 180 pivot, store    *
                     228     ; a decision, or get a decision from the stack.                      *
                     229     ;*********************************************************************
                     230
                     231     ;decide:
                     232                     ;r_turn, l_turn, get_out, make_180 as public bits
                     233                     ;r_decide, l_decide, f_decide as local bits
                     234             ;This S.R. decides when to start the decel using a step count
                     235             ;when it sees a front wall or when it is time to turn. It also
                     236             ;decides which way to turn on a mapping run or a final run.
                     237                     ;1) look for a wall to no wall transition
00AB 201853          238                     jb      det_L2H,di_3_1  ;skip the stuff below if set
00AE A215            239                     mov     c,prev_r
00B0 B002            240                     anl     c,/r_wall       ;check for r wall transition
00B2 5003            241                     jnc     di_1            ;if no transition goto di 1
00B4 750800          242                     mov     step_count,#0   ;start counting
00B7 A216            243     di_1:           mov     c,prev_l
00B9 B003            244                     anl     c,/l_wall       ;check for l wall transition
00BB 5003            245                     jnc     di_2            ;if no transition goto di 2
00BD 750800          246                     mov     step_count,#0   ;start counting
                     247     di_2:
                     248                     ;2) now detect step count
00C0 E508            249                     mov     a,step_count    ;check to see if
00C2 C3              250                     clr     c
00C3 9454            251                     subb    a,#84           ;  80 <= steps <= 83
00C5 5013            252                     jnc     di_3
00C7 E508            253                     mov     a,step_count
00C9 C3              254                     clr     c
00CA 9450            255                     subb    a,#80
00CC 400C            256                     jc      di_3
00CE A202            257                     mov     c,r_wall        ;store R wall into decide storage
00D0 9211            258                     mov     r_decide,c      ;  for use later in CASE
00D2 A203            259                     mov     c,l_wall        ;
00D4 9212            260                     mov     l_decide,c      ;  when CASE statment is built below
00D6 C21D            261                     clr     skip_decide
00D8 2197            262     di_2_0:         ajmp    di_9            ;get out since did above stuff
                     263
                     264     di_3:           ;3)detect no wall to wall (lo to hi) and check for front wall
                     265                     ;count must be greater than 120 to allow for spaces in posts
00DA E508            266                     mov     a,step_count
00DC C3              267                     clr     c
00DD 9478            268                     subb    a,#120
00DF 4020            269                     jc      di_3_1
00E1 A215            270                     mov     c,prev_r        ;the prev should be low
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                   04/16/92   PAGE    6

LOC  OBJ           SOURCE
00E3 A002                          orl     c,/r_wall       ;check for r wall transition
00E5 400A                          jc      di_3_0          ;IF no lo to hi then di_3_0
                                                   ;found lo to hi, set step counter
00E7 750800   F                    mov     step_count,#w2nw_cnt
00EA C213                          clr     f_decide
00EC 750B00                        mov     count_fw,#0     ;clr count for use in decel S.R.
00EF D218                          setb    det_L2H
00F1 A216          di_3_0:         mov     c,prev_l        ;the prev should be low
00F3 A003                          orl     c,/l_wall       ;check for l wall transition
00F5 400A                          jc      di_3_1          ;IF no lo to hi then di_3_1
                                   ;found lo to hi, set step counter
00F7 750800   F                    mov     step_count,#w2nw_cnt
00FA C213                          clr     f_decide
00FC 750B00                        mov     count_fw,#0
00FF D218                          setb    det_L2H
                   di_3_1:                         ;condition that will get you to di_3_9
                                                   ;det_L2H & count > 162
0101 A204                          mov     c,f_wall        ;  trigger from extrn lite during
0103 9213                          mov     f_decide,c      ;  other points in square
0105 400E                          jc      di_3_9          ;get out if meet condition ELSE contnu
0107 3018CE                        jnb     det_L2H,di_2_0  ;this section checks second cond

010A E50B                          mov     a,count_fw
010C B40002   F                    cjne    a,#chk4fr,di_3_2
010F 2115                          ajmp    di_3_9
0111 050B          DI_3_2:         inc     count_fw        ;
0113 2197                          ajmp    di_9            ;get out

                   di_3_9: ;at this point we have detected it's time to stop, store in RAM, and
                           ;execute plan.... Therefore set bit to skip all of this
0115 201DC0                        jb      skip_decide,di_2_0
0118 D21D                          setb    skip_decide

011A C218                          clr     det_L2H
011C 750800                        mov     step_count,#0
011F 202054                        jb      done,dif_0      ;IF done THEN dif_0 ELSE fall thru

                   di_4:           ;4) at this point F L R are loaded, set up CASE table
                           ; L=01  R=10  S=11  stop-mouse=00    all represented as two bits.
0122 7400                          mov     a,#0
0124 A221                          mov     c,l_r_bit       ;if no back up then we have this...
0126 33                            rlc     a               ;  |
0127 A212                          mov     c,l_decide      ;  |
0129 33                            rlc     a               ;  |  L/R_bit  1=L
012A A211                          mov     c,r_decide      ;  |           0=R
012C 33                            rlc     a               ;  |
012D A213                          mov     c,f_decide      ;  |
012F 33                            rlc     a               ;| 0 | 0 | 0 | 0 |L/R| L | R | F |
0130 9001A1                        mov     dptr,#d_table   ;point to decision table
0133 93                            movc    a,@a+dptr       ;get decision
0134 FA                            mov     r2,a            ;save it for later

0135 201A14                        jb      back_up,di_4_0  ;IF backing up GOTO di
                           ;5) execute the table from here.  | A | part is taken care of here
0138 A2E7                          mov     c,acc.7         ;setting make_180 flag
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                       04/16/92   PAGE    7

LOC  OBJ            LINE     SOURCE
013A 9214                                   mov     make_180,c
013C 9210                                   mov     get_out,c
013E 921A                                   mov     back_up,c
0140 4055                                   jc      di_9            ;if make_180 then return
                                            ;this determines | D | direction to turn
0142 5403                                   anl     a,#00000011b    ;mask out junk to get | D |
0144 F5F0                                   mov     b,a
0146 31E3                                   acall   store_val       ;store the decision and then store
0148 31C1                                   acall   inc_map_ptr     ;  a halt afterwards and decr

014A 802E                                   sjmp    dx_58           ;go execute it below

014C 31D2                    di_4_0:        acall   dec_map_ptr
014E 5117                                   acall   get_val         ;get top of stack, put in B
0150 EA                                     mov     a,r2
0151 5403                                   anl     a,#00000011b    ;keep | D | (dir to turn)
0153 C5F0                                   xch     a,b             ;stack top in A, | D | in B
0155 23                                     rl      a               ;stack top now looks like this
0156 23                                     rl      a               ;   0000XX00
0157 45F0                                   orl     a,b             ;looks like 0000|prev| D |
0159 D2E4                                   setb    acc.4           ;looks like 000|B/U|prev| D |

015B 93                                     movc    a,@a+dptr       ;get decision
015C FA                                     mov     r2,a            ;save it just in case
015D C214                                   clr     make_180
015F A2E6                                   mov     c,acc.6         ;setting or clearing back-up
0161 921A                                   mov     back_up,c
0163 201A14                                 jb      back_up,dx_58   ;if still backing, dont store
0166 C4                                     swap    a
0167 5403                                   anl     a,#00000011b    ;| C | now in B
0169 F5F0                                   mov     b,a
016B 31E3                                   acall   store_val       ;  and now stored in RAM
016D 31C1                                   acall   inc_map_ptr
016F EA                                     mov     a,r2            ;get decision and mask to get
0170 5403                                   anl     a,#0000011b     ;   | D | dir to turn
0172 F5F0                                   mov     b,a
0174 8004                                   sjmp    dx_58           ;go execute it below

                                     ;6) skip all of the junk above and do this if running finals
                             dif_0:         ;do this if running final solution
0176 5117                                   acall   get_val         ;get the turn decision
0178 31C1                                   acall   inc_map_ptr     ;incr map pointer


                                     ;7) interpret A to set the turn flags accordingly
                             dx_58:                                 ;r_turn, l_turn flags
017A E5F0                                   mov     a,b             ;cjne's only work on acc not b
017C B40108                                 cjne    a,#01,dx_52     ;look for
017F D208                                   setb    r_turn
0181 C209                                   clr     l_turn
0183 D210                                   setb    get_out
0185 8010                                   sjmp    di_9
0187 B40208                  dx_52:         cjne    a,#02,dx_53
018A C208                                   clr     r_turn
018C D209                                   setb    l_turn
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                          04/16/92   PAGE    8

LOC  OBJ            LINE     SOURCE
018E D210                                   setb    get_out
0190 8005                                   sjmp    di_9
                             dx_53:         ;must be b=11 or b=00, straight or stop
                                    ;if 00 then stop or HALT
0192 B40002                                 cjne    a,#0,di_9
0195 8153                                   ajmp    halt

                             di_9:  ;end of S.R. here at di-9
0197 A202                                   mov     c,r_wall
0199 9215                                   mov     prev_r,c        ;store the wall condition for next
019B A203                                   mov     c,l_wall        ;  toggle look
019D 9216                                   mov     prev_l,c

019F 412C                                   ajmp    decide_x

                             ;*****************************************************************************
                             d_table:       ;this holds all of the answers for the decision table
                                            ;byte form looks like this.... | A | B | C | D | (two bits ea)
                                            ; A - XY, X=do 180, Y=B-U flag status
                                            ; B - add to stack if in B-U,  L=10  R=01  S=11  stop-mouse=00
                                            ; C - what to add to stack not in B-U  (B-U means back-up mode)
                                            ; D - direction to turn
01A1 05                                     db      00000101b,00000101b,00001111b,00001010b
01A2 05
01A3 0F
01A4 0A
01A5 05                                     db      00000101b,00000101b,00001111b,11001111b
01A6 05
01A7 0F
01A8 CF
01A9 0A                                     db      00001010b,00001010b,00001010b,00001010b
01AA 0A
01AB 0A
01AC 0A
01AD 0F                                     db      00001111b,00000101b,00001111b,11001111b
01AE 05
01AF 0F
01B0 CF
                                    ;these lines are for the B-U mode decisions, some non-valid
01B1 00                                     db      00h,00h,00h,00h
01B2 00
01B3 00
01B4 00
01B5 00                                     db      00h,00110001b,01000010b,00100011b
01B6 31
01B7 42
01B8 23
01B9 00                                     db      00h,01000001b,00110010b,00010011b
01BA 41
01BB 32
01BC 13
01BD 00                                     db      00h,00100001b,00010010b,01000011b
01BE 21
01BF 12
01C0 43
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                       04/16/92   PAGE    9

LOC  OBJ            SOURCE

                                    ;| 0 | 0 | 0 | 0 |L/R| L | R | F |       for normal address
                                    ;| 0 | 0 | 0 |B/U| PREV |WAY_2_GO|       for back-up address
                                    ;                |2 bits| 2 bits |
                    ;**********************
                    ;map representation in RAM will look like this...
                    ;  |most sig 2 bits| XX | XX |least sig 2 bits|
                    ;  location = | D | C | B | A | where A is the first decision, B is second
                    ;  etc. A, B, C, D are all two bits. The next byte of RAM is rep the same.
                    ;  It requires R1 as pointer and MAP_OFFSET 10h
                    ; L=10  R=01  S=11  stop-mouse=00    all represented as two bits.
                    ;If pointer = 3Fh and map offset=4 then end of RAM
                    ;**********************
                    inc_map_ptr:    ;this S.R. incs the map pointer from 0 to 3 and map_org
                                    ;MAP uses mem-addr 11h-1fh and 2ch-3fh. R1 holds addr.

01C1 AF10                           mov     r7,map_offset   ;cjne only works on local regs
01C3 BF0309                         cjne    r7,#3,imp_0     ;IF 3 THEN do below ELSE imp0
01C6 7510FF                         mov     map_offset,#0ffh ;start at MSB two bits (offset=0)
01C9 B91F02                         cjne    r1,#1fh,imp_1
01CC 792B                           mov     r1,#2bh
01CE 09             imp_1:          inc     r1              ;  and also move pointer high
01CF 0510           imp_0:          inc     map_offset
01D1 22                             ret

                    ;**********************
                    dec_map_ptr:    ;this S.R. decr the map pointer and returns what's on the
                                    ; top of the stack in B
01D2 AF10                           mov     r7,map_offset   ;cjne only works on local regs
01D4 BF0009                         cjne    r7,#0,dmp_0     ;IF 0 THEN do below ELSE dmp_0
01D7 751004                         mov     map_offset,#4   ;point |XX|XX|XX|00| at 00
01DA B92C02                         cjne    r1,#2ch,dmp_1
01DD A920                           mov     r1,20h
01DF 19             dmp_1:          dec     r1              ;map pointer R1
01E0 1510           dmp_0:          dec     map_offset
01E2 22                             ret

                    ;**********************
                    store_val:      ;requires the decision to be in B as 000000XX and pointed to
01E3 C7                             xch     a,@R1           ;get byte for below but save a
01E4 AEF0                           mov     r6,b            ;decision is in B
01E6 AF10                           mov     r7,map_offset   ;get it
01E8 BF0002                         cjne    r7,#0,sv_a      ;are we at 0 yet?
01EB 8004                           sjmp    sv_b            ;  I guess we are
01ED 03             sv_a:           rr      a               ;roll bits to shift |00|XX|00|00|
01EE 03                             rr      a               ;to get             |XX|00|00|00|
01EF DFFC                           djnz    r7,sv_a         ;keep shifting til  |00|00|00|XX|
01F1 C2E0           sv_b:           clr     acc.0           ;clear it out to load it below
01F3 C2E1                           clr     acc.1
01F5 BE0104                         cjne    r6,#01,sv_0     ;load R if 01
01F8 D2E0                           setb    acc.0           ;looks like |XX|XX|XX|01|
01FA 800E                           sjmp    sv_2
01FC BE0204         sv_0:           cjne    r6,#02,sv_1     ;load L if 02
01FF D2E1                           setb    acc.1           ;looks like |XX|XX|XX|10|
0201 8007                           sjmp    sv_2
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                          04/16/92   PAGE   10

LOC  OBJ             LINE     SOURCE
0203 BE0304           467     sv_1:          cjne    r6,#03,sv_2
0206 D2E0             468                    setb    acc.0          ;load Straight AKA S
0208 D2E1             469                    setb    acc.1          ;looks like |XX|XX|XX|11|
020A AF10             470     sv_2:          mov     r7,map_offset  ;roll until bits in original position
020C BF0002           471                    cjne    r7,#0,sv_c     ;are we at 0 yet?
020F 8004             472                    sjmp    sv_d           ;  I guess we are
0211 23               473     sv_c:          rl      a              ;roll bits to shift |00|00|00|XX|
0212 23               474                    rl      a              ;to get             |XX|00|00|00|
0213 DFFC             475                    djnz    r7,sv_c        ;keep shifting til  |00|XX|00|00|
0215 C7               476     sv_d:          xch     a,@R1          ;put it all back
0216 22               477                    ret
                      478
                      479     ;**********************
                      480     get_val:       ;this S.R. returns turn value in B
0217 F509             481                    mov     temp_reg1,a    ;save acc same as push acc
0219 E7               482                    mov     a,@r1
021A AF10             483                    mov     r7,map_offset  ;get it
021C BF0002           484                    cjne    r7,#0,gv_0     ;are we at 0 yet?
021F 8004             485                    sjmp    gv_1           ;  I guess we are
0221 03               486     gv_0:          rr      a              ;roll bits to shift |00|00|00|XX|
0222 03               487                    rr      a              ;to get             |00|00|XX|00|
0223 DFFC             488                    djnz    r7,gv_0        ;keep shifting til  |XX|00|00|00|
0225 5403             489     gv_1:          anl     a,#00000011b   ;now have |00|00|00|XX|
0227 F5F0             490                    mov     b,a            ;stuff it back
0229 E509             491                    mov     a,temp_reg1
022B 22               492                    ret
                      493
                      494
                      495     ;*********
                      496
022C 101002           497     decide_x:      jbc     get_out,decel  ;get out, time to stop
022F 015C             498                    ajmp    acc_0          ;do this again
                      499
                      500
                      501     ;*********************************************************************
                      502     ; This section decelerates the motors.  The table value is lowered   *
                      503     ; if the mouse is too close to a wall.                               *
                      504     ;*********************************************************************
                      505
                      506     decel:
0231 AE0B             507                    mov     r6,count_fw    ;get no. steps past post from count-fw
0233 7400      F      508                    mov     a,#decel_steps
0235 C3               509                    clr     c
0236 9E               510                    subb    a,r6
0237 FE               511                    mov     r6,a           ;R6 now has decel_steps - count_fw
                      512
0238 7B00      F      513                    mov     r3,#decel_decr ;decel decr is now in r3
023A E50F             514                    mov     a,r_ptr        ;check to see how far into accel tab
023C C3               515                    clr     c
023D 9400      F      516                    subb    a,#acc_steps   ;C set if r_ptr < top_speed
023F 5002             517                    jnc     dec_0a
0241 7B01             518                    mov     r3,#1          ;this is the decel_decr value
                      519
0243 EE               520     dec_0a:        mov     a,r6
0244 8BF0             521                    mov     b,r3
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                 04/16/92   PAGE   11

LOC  OBJ           LINE     SOURCE
0246 A4                                    mul      ab
0247 04                                    inc      a
0248 F50F                                  mov      r_ptr,a
024A F50E                                  mov      l_ptr,a

024C 300BFD                 dec_0:         jnb      time_int,$       ;stay here until int happens
024F C20B                                  clr      time_int
0251 71B7                                  acall    snapshot         ;take a look at the walls
                            ;right routine
0253 300C08                                jnb      r_int,dec_1      ;skip if right hasn't been stepped yet
0256 C20C                                  clr      r_int
0258 E50F                                  mov      a,r_ptr          ;get right pointer
025A C3                                    clr      c
025B 9B                                    subb     a,r3             ;slow down by getting lesser value
025C F50F                                  mov      r_ptr,a          ;ptr just got decremented

                            ;left routine
025E 300DEB                 dec_1:         jnb      l_int,dec_0      ;if need to do left then goto dec-1
0261 C20D                                  clr      l_int
0263 E50E                                  mov      a,l_ptr          ;get left pointer
0265 C3                                    clr      c
0266 9B                                    subb     a,r3             ;slow down by getting lesser value
0267 F50E                                  mov      l_ptr,a          ;ptr just got decremented
0269 DEE1                   dec_2:         djnz     r6,dec_0         ;decr number of steps to decel
026B C28C                                  clr      tr0              ;stop the timer int
026D 7F00       F                          mov      r7,#pause_val    ;pause value
026F 9145                                  acall    pause            ;stay stationary for a while


                            ;*********************************************************************
                            ; This section makes the mouse turn 90 or 180 degrees using the     *
                            ; decision from the "decision-maker".                               *
                            ;*********************************************************************

                            ;pivot:         ;routine to pivot the mouse 90 or 180 degrees
0271 E58B                                  mov      a,rtl
0273 C3                                    clr      c
0274 9400       F                          subb     a,#pivot_speed ;slower speed
0276 30140D                                jnb      make_180,turn90 ;do 180 else turn
0279 101706                                jbc      cw180,piv_0      ;180 cw
027C D217                                  setb     cw180            ;next time 180 ccw
027E C293                                  clr      l_dir            ;make L go backwards
0280 4190                                  ajmp     piv_1
0282 C291                   piv_0:         clr      r_dir            ;make R go bw
0284 4190                                  ajmp     piv_1

0286 A208                   turn90:        mov      c,r_turn         ;this section sets the motor dir
0288 B3                                    cpl      c                ;   bits according to which
0289 9291                                  mov      r_dir,c          ;   dir it has to turn.
028B A209                                  mov      c,l_turn
028D B3                                    cpl      c
028E 9293                                  mov      l_dir,c

                            piv_1:  ;THIS IS NEW.  Mod adds or subtracts steps depending on sensor info
0290 201404                                jb       make_180,piv_2y ;if mouse is skewed, it turns more
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                   04/16/92   PAGE   12

LOC  OBJ                SOURCE
0293 7F00     F                             mov     r7,#half_90     ;  or less degrees
0295 8002                                   sjmp    piv_2x
0297 7F00     F         piv_2y:             mov     r7,#half_180    ;
                        piv_2x: ;this section determines add or subt count depending on dir to pivot
                                ;(r_dir & too_l) | (l_dir & too_r) = - steps
                                ;(r_dir & too_r) | (l_dir & too_l) = + steps
0299 A291                                   mov     c,r_dir         ;check for first condition
029B 8201                                   anl     c,too_l
029D 9222                                   mov     temp_bit1,c
029F A293                                   mov     c,l_dir
02A1 8200                                   anl     c,too_r
02A3 7222                                   orl     c,temp_bit1
02A5 5003                                   jnc     piv_2z
02A7 1F                                     dec     r7              ;decr step count to do pivot
02A8 800F                                   sjmp    piv_2v
02AA A291               piv_2z:             mov     c,r_dir         ;check for second condition
02AC 8200                                   anl     c,too_r
02AE 9222                                   mov     temp_bit1,c
02B0 A293                                   mov     c,l_dir
02B2 8201                                   anl     c,too_l
02B4 7222                                   orl     c,temp_bit1
02B6 5001                                   jnc     piv_2v
02B8 0F                                     inc     r7              ;incr step count to do pivot

02B9 750F00             piv_2v:             mov     r_ptr,#0        ;point at first accel value
02BC 750E00                                 mov     l_ptr,#0
02BF 750DFE                                 mov     r_timr,#0feh    ;even up the timers to step
02C2 750CFE                                 mov     l_timr,#0feh    ;   evenly
02C5 D28C                                   setb    tr0             ;enable timer

02C7 300CFD             piv_2:              jnb     r_int,$         ;stay here until done
02CA C20C                                   clr     r_int           ;ack r interrupt
02CC 050F                                   inc     r_ptr           ;incr step count
02CE 050E                                   inc     l_ptr           ;incr step count
02D0 E50F                                   mov     a,r_ptr         ;see if at half way mark
02D2 201405                                 jb      make_180,piv_2a
02D5 B507EF                                 cjne    a,07,piv_2      ;IF half way fall thru to piv_3
02D8 8003                                   sjmp    piv_3
02DA B507EA             piv_2a:             cjne    a,07,piv_2      ;if half way, fall thru

02DD B40002   F         piv_3:              cjne    a,#look90,piv_4
02E0 D21B               piv_3a:             setb    look            ;if so, set look bit
02E2 300CFD             piv_4:              jnb     r_int,$         ;stay here until int done
02E5 C20C                                   clr     r_int           ;ack int
02E7 150F                                   dec     r_ptr           ;make it slow down
02E9 150E                                   dec     l_ptr
02EB 301B11                                 jnb     look,piv_5      ;if look is set, do snapshot

02EE 71B7                                   acall   snapshot        ;if aligned & wall exist truth table:
02F0 20051C                                 jb      aligned,piv_end
02F3 A201                                   mov     c,too_l
02F5 B093                                   anl     c,/l_dir
02F7 4016                                   jc      piv_end
02F9 A200                                   mov     c,too_r
02FB B091                                   anl     c,/r_dir
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                   04/16/92   PAGE   13

LOC  OBJ           SOURCE
02FD 4010                          jc      piv_end

02FF E50F          piv_5:          mov     a,r_ptr         ;
0301 70DA                          jnz     piv_3           ;IF end of count fall thru

0303 A202                          mov     c,r_wall
0305 7203                          orl     c,l_wall
0307 5006                          jnc     piv_end         ;IF no walls get out

0309 050F                          inc     r_ptr           ;keep steppin while you see
030B 050E                          inc     l_ptr           ;   a wall
030D 41E2                          ajmp    piv_4

                   piv_end:
030F C28C                          clr     tr0             ;off timer int
0311 850A8B                        mov     rtl,ls373       ;full speed again
0314 C21B                          clr     look            ;
0316 C20C                          clr     r_int
0318 C20D                          clr     l_int
031A 43900A                        orl     p1,#00001010b   ;set r dir l dir
031D 7F00      F                   mov     r7,#pause_val   ;pause for a bit
031F 9145                          acall   pause

0321 0141                          ajmp    accel           ;goto accel


                   ;*********************************************************************
                   ; This is the Timer 0 interrupt subroutine.                          *
                   ; It will step a motor if the "prescale" for the corresponding       *
                   ; motor overflows (or decrements to zero).                           *
                   ;*********************************************************************

                   tim_0:          ;used to step the motors
0323 C28C                          clr     tr0             ;quit counting
0325 C0D0                          push    psw
0327 C0E0                          push    acc
0329 C082                          push    dpl
032B C083                          push    dph
032D 900000    F                   mov     dptr,#acc_tab
0330 D50D16                        djnz    r_timr,tim_00
0333 C292                          clr     r_step          ;step the R motor
0335 E50F                          mov     a,r_ptr         ;load timer value from pointer
0337 93                            movc    a,@a+dptr       ;get higher byte accel value
0338 F50D                          mov     r_timr,a        ;load the timer value from table
033A E508                          mov     a,step_count
033C B48202                        cjne    a,#130,tim_x
033F 8002                          sjmp    tim_y
0341 0508          tim_x:          inc     step_count      ;decide uses this stuff
0343 D20C          tim_y:          setb    r_int           ;flag that the R mot was stepped
0345 D20B                          setb    time_int        ;int has occurred flag
0347 D292                          setb    r_step
0349 D50C0D        tim_00:         djnz    l_timr,tim_ret  ;get out if not zero
034C C294                          clr     l_step          ;step the L motor
034E E50E                          mov     a,l_ptr         ;load timer value from pointer
0350 93                            movc    a,@a+dptr       ;get higher byte accel value
```

IEEE Micro Mouse using the 87C751 microcontroller AN443

```
MCS-51 MACRO ASSEMBLER    751MAIN                                   04/16/92   PAGE   14

LOC  OBJ           SOURCE
0351 F50C                               mov     l_timr,a        ;load the timer value from table
0353 D20D                               setb    l_int           ;flag that the L mot was stepped
0355 D20B                               setb    time_int        ;int has occurred flag
0357 D294                               setb    l_step
                   tim_ret:
0359 D083                               pop     dph
035B D082                               pop     dpl
035D D0E0                               pop     acc
035F D0D0                               pop     psw
0361 D28C                               setb    tr0             ;start counting
0363 32                                 reti
                   ;*********************************************************************
                   ; This is the External interrupt 1 subroutine.                       *
                   ;*********************************************************************
                   int_1:               ;used for setting left or right hug
                                        ; L/R=0 right algo, L/R=1 left algo
                                        ;AND also setting speed of run
0364 202010                             jb      done,int_1_2    ;incr speed and get out
0367 10210B                             jbc     l_r_bit, int_1_0
036A D221                               setb    l_r_bit         ;hug left
036C C282                               clr     l_led
036E 7F0A                               mov     r7,#10          ;value for 2 seconds
0370 9145                               acall   pause
0372 D282                               setb    l_led
0374 32                                 reti
0375 C221          int_1_0:             clr     l_r_bit         ;hug right
0377 7E05          int_1_2:             mov     r6,#5
0379 302002        int_1_1:             jnb     done,int_1_3
037C 058B                               inc     rtl             ;incr the speed
037E C282          int_1_3:             clr     l_led
0380 7F02                               mov     r7,#2
0382 9145                               acall   pause
0384 D282                               setb    l_led
0386 7F02                               mov     r7,#2
0388 9145                               acall   pause
038A DEED                               djnz    r6,int_1_1
038C 32                                 reti
                   ;*********************************************************************
                   ; This is the External interrupt 0 subroutine.                       *
                   ;*********************************************************************
                   int_0:               ;used for starting and stopping the thing
038D C28C                               clr     tr0
038F 200A06                             jb      s_s_int, int_0_0        ;we are here to start up
0392 D20A                               setb    s_s_int
0394 C290                               clr     mot_en
0396 61AE                               ajmp    int_0_ret
0398 D290          int_0_0:             setb    mot_en          ;we are here cause at finish box
039A D220                               setb    done            ;tell the prog that we are done
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                     04/16/92   PAGE   15

LOC  OBJ            SOURCE
039C C20A                          clr     s_s_int
039E 90001D                        mov     dptr,#main_1    ;get address
03A1 A881                          mov     r0,sp           ;set up to start all over
03A3 A683                          mov     @r0,dph         ;load reti vector to 0004h
03A5 18                            dec     r0
03A6 A682                          mov     @r0,dpl
03A8 75A800                        mov     ie,#00
03AB 758800                        mov     tcon,#00        ;clears any ints

03AE C297           int_0_ret:     clr     r_led
03B0 7F14                          mov     r7,#20          ;value for 2 seconds
03B2 9145                          acall   pause
03B4 D297                          setb    r_led
03B6 32                            reti


                    ;*********************************************************************
                    ; This section strobes the sensors for data.                         *
                    ;*********************************************************************

                    snapshot:      ;takes a look at walls and sets bits accordingly
                    ;R4, R5 for sensor info. ACC, C, r l sens, bit addressables
03B7 75B0FF                        mov     p3,#0ffh
03BA 752000                        mov     ss_bits,#0      ;clear all flags
03BD C281                          clr     r_sens          ;enable right sensor bank
03BF A4                            mul     ab              ;causes a 6uS wait state
03C0 A4                            mul     ab
03C1 A4                            mul     ab
03C2 A4                            mul     ab
03C3 E5B0                          mov     a,p3            ;store right wall
03C5 D281                          setb    r_sens
03C7 FC                            mov     r4,a            ;wall is now repr as a high
03C8 33                            rlc     a               ;store R sens 0 for f_wall
03C9 9204                          mov     f_wall,c
03CB 6002                          jz      ss_0            ;if no wall goto ss_0
03CD D202                          setb    r_wall          ;right wall present

03CF C280           ss_0:          clr     l_sens          ;enable left sensor bank
03D1 A4                            mul     ab              ;causes a 6uS wait state
03D2 A4                            mul     ab
03D3 A4                            mul     ab
03D4 A4                            mul     ab
03D5 E5B0                          mov     a,p3            ;store left wall
03D7 D280                          setb    l_sens
03D9 FD                            mov     r5,a            ;wall is now repr as a high
03DA 33                            rlc     a               ;grab inner sens and or it
03DB 7204                          orl     c,f_wall
03DD 9204                          mov     f_wall,c        ;store front wall
03DF 6002                          jz      ss_2            ;if no wall goto ss_2
03E1 D203                          setb    l_wall          ;left wall present

03E3 20045E         ss_2:          jb      f_wall,ss_ret   ;if no front wall then cont
03E6 7400                          mov     a,#0            ;build case statement
03E8 A221                          mov     c,l_r_bit       ;     |     |    |    |
03EA 33                            rlc     a               ;00000| L/R | L  | R  |
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                    04/16/92   PAGE   16

LOC  OBJ            LINE     SOURCE
03EB A2O3                                        mov     c,l_wall        ;     |algo |wall|wall|
03ED 33                                          rlc     a               ;     |     |bit |bit |
03EE A202                                        mov     c,r_wall        ;     |     |    |    |
03F0 33                                          rlc     a               ;done shifting in bits for case stmnt
03F1 B40102                                      cjne    a,#01,ss_4      ;chk R
03F4 810F                                        ajmp    ss_9
03F6 B40302                  ss_4:               cjne    a,#03,ss_5      ;chk R
03F9 810F                                        ajmp    ss_9
03FB B40502                  ss_5:               cjne    a,#05,ss_6      ;chk R
03FE 810F                                        ajmp    ss_9
0400 B40202                  ss_6:               cjne    a,#02,ss_7      ;chk L
0403 8115                                        ajmp    ss_10
0405 B40602                  ss_7:               cjne    a,#06,ss_8      ;chk L
0408 8115                                        ajmp    ss_10
040A B40737                  ss_8:               cjne    a,#07,ss_ret    ;if eq then chk L else do nothing
040D 8115                                        ajmp    ss_10
                             ;**********
                             ss_9:               ;check the right side offset
040F 8CF0                                        mov     b,r4            ;put wall info into acc
0411 C21E                                        clr     genp2           ;0=chk R
0413 8004                                        sjmp    ss_93
0415 8DF0                    ss_10:              mov     b,r5
0417 D21E                                        setb    genp2           ;1=chk L

0419 E5F0                    ss_93:              mov     a,b
041B 5400      F                                 anl     a,#sens_pat     ;masking for cmp, 3 high 4 low
041D B40804                                      cjne    a,#08h,ss_90    ;check for aligned condition
0420 D205                                        setb    aligned         ;perfectly on wall
0422 8144                                        ajmp    ss_ret
0424 E5F0                    ss_90:              mov     a,b             ;get wall info again
0426 5403                                        anl     a,#00000011b    ;check for too close center if a > 0
0428 600B                                        jz      ss_91           ;if no wall on ones then not too_
042A 201E04                                      jb      genp2,ss_101
042D D201                                        setb    too_l
042F 8144                                        ajmp    ss_ret
0431 D200                    ss_101:             setb    too_r
0433 8144                                        ajmp    ss_ret
0435 E5F0                    ss_91:              mov     a,b             ;get wall info again
0437 5460                                        anl     a,#01100000b    ;check for too close wall if a > 0
0439 6009                                        jz      ss_ret
043B 201E04                                      jb      genp2,ss_102
043E D200                                        setb    too_r
0440 8144                                        ajmp    ss_ret
0442 D201                    ss_102:             setb    too_l
0444 22                      ss_ret:             ret

                             ;********************************************************************************
                             pause:              ;pause loop using R7 as the loop counter
0445 C28C                                        clr     tr0
0447 903EFE                                      mov     dptr,#03efeh
044A D582FD                                      djnz    dpl,$
044D D583FA                                      djnz    dph,$-3
0450 DFF3                                        djnz    r7,pause
0452 22                                          ret
```

```
MCS-51 MACRO ASSEMBLER    751MAIN                                  04/16/92   PAGE   17

LOC  OBJ            SOURCE
                    ;*****************************************************************************
                    halt:           ;This S.R. brings the thing to a halt - mostly used for debug
0453 C2A9                           clr     et0
0455 D290                           setb    mot_en
0457 80FE                           sjmp    $
                    ;*****************************************************************************
                    EXTRN CODE              (acc_tab)  ;from the DOS file 751acc.asm

                                    end
```