APPENDICES

APPENDIX A - Stating the facts

A1..... Hazard Assessment Form *(hand-written form – not in PDF)*

A2..... Initial and final objectives contrasted

A3..... Equipment used

A4..... Bibliography

APPENDIX B - Data Sheets *(All photo-copied from data books – not in PDF)*

B1..... LM1812 Ultrasonic Transceiver

B2..... 74HC4511 BCD-to-Seven-Segment Latch/ Decoder / Display driver

B3..... CA3240 Dual BiMOS operational Amplifier

B4..... CA5260 Dual Microprocessor MOSFET operational Amplifier

B5..... TLC339 Quad Micropower LinCMOS Comparators

B6..... 74HC4351 8-Channel analog multiplexer/demux. with latch

B7..... CA3306 CMOS Video-speed 6-Bit Flash Analog to Digital converter

B8..... Additional data on the MA40S2S/R ultrasonic transducer pair

APPENDIX C - Extra info related to the Mini-Mobot

C1..... The MEMMOD memory expansion board for the MIni-mobot

C2..... An Example pulse-echo receiving program

C3..... PCBs and Pinouts *(not all in PDF yet)*

Comparison of initial objectives with final **Proposal document date 11 oct' 91**

I propose to design and build an all-round ultra-sonic proximity detection system for the mini-mobot robots and to investigate the system's possible duel use as a simple inter-robot communications system.

My proposed method for dealing with the above is as follows:-

Firstly, to investigate, generally, different systems of ultrasonic sensing, ranging and proximity detection, and to write a set of requirements for my system. Then to select the most suitable system, giving reasons for the rejection of the other systems, and from it, design a possible circuit.

Investigation and selection of design presented in this report

Following this, I will build and test a simple prototype circuit, to see if it produces the correct results and to alter the test circuit accordingly until it meets the required specification. This will require the designing and building of various test set-ups as required, and the evaluation of the results produced. *Ditto*

At this point, I will investigate if the circuit is suitable for use as a communications system as well as a proximity detector and if so alter the circuit accordingly.

The communication system was built in from the outset, as it involved very little circuit alteration

From this circuit design, I will design and build the robot mounted version, with full 360° sensing, and investigate it's correct functioning and limitations as a proximity detector and as a communications system. From this a Mk2 version might require building, depending on the test results.

Note that range-finding (the actual measurement of distance to objects) was never mentioned at this stage, and was added to the objectives later

Finally, I will write software for the micro-controller, which runs the mini-mobots, to make best use of the designed ultrasonics in various single and multiple mini-mobot tasks eg. to skirt objects at a set distance, to send a mayday communication if the mobot encounters problems and for other mobots to react accordingly, etc. *Time did not permit!, so no software was written for the system*

If time permits, I will write software to investigate if the mobot can map it's environment, using the ultrasonic system, with any degree of accuracy.

A log-book will be kept through-out the project. *(It was)*

Equipment used

For testing;

- Hameg HM 205.2 Digital Storage oscilloscope (No. 16008)
- Hewlett Packard Logic Analyser
- Various Farnell single and dual power supplies
- Various Signal generators
- Polaroid 667 oscilloscope camera

For the project in general;

- PCB's laid out using BoardMaker
- Circuit diagrams drawn using ORCAD
- Report written using WordPerfect 5.1
- Diagrams produced using drawperfect, paintbrush etc.

Bibliography

1)..The Complete Bat (1990)

Author - James Robertson

Published by - Chatto and Windus Ltd.

2)..Eight-Bit 80C51 Embedded Processors (1990)

Author & Publisher - Advanced Micro Devices.

3)..Ultrasonics Applications Manual, S15E

Author & Publisher - Murata Electronics (UK) Ltd.

4)..NATO Advanced Research Workshop on Sensor Devices and Systems for Robotics, Oct.1987

Edited by - Alicia Casals

Published by - Springer-Verlag

5)..Microelectronic Circuits

Authors - Sedra & Smith

Published by - ?

6)..Electronic Communication Techniques (second ed.)

Author - P. Young

Published by - Macmillan

7)..The Art of Electronics (second ed.)

Authors - Horowitz and Hill

Published by - Cambridge University Press

Taken from P4,5,9,10,11 - Murata Ultrasonics Applications manual S15E

Overview

In it's original design, the Mini-Mobot has no RAM, other than the 256 bytes built into the 80C32 micro-controller. This was seen to be a problem from the start, and for this reason, a general purpose memory board was designed to fill the following requirements;

1). Provide enough extra memory such that lack of memory would never be a problem again.

2). Provide the memory in a form that could be fitted to the standard mobot setup, without the need for a whole new PCB level.

3). Be compatible with all future sensor boards.

As all bar 2 of the 28 signals required are to be found at the EPROM socket, a daughter board was designed that fits this socket. On the daughter board both the EPROM and a 32Kbyte x 8 bit SRAM are mounted. Although the micro-controller can address 64K of RAM, I/O addresses for the sensor boards etc. are in the top 32K of memory, thus limiting the maximum actual memory to 32K which is more than enough. The first MEMory MODification (MEMMOD) board used a standard DIL SRAM sat next to the EPROM, but it was too high, required extension legs to be fitted to the inter-board connectors; Thus MEMMOD MK2 was designed with the SRAM surface mounted under the EPROM. Only MEMMOD MK2 is presented here as MEMMOD1 was only a partial solution (see log-book for MEMMOD1 details)

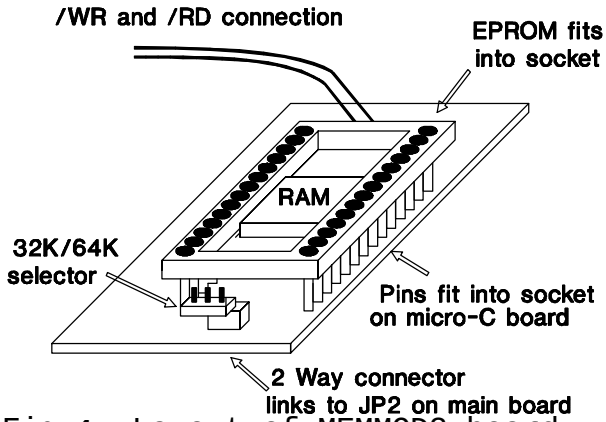

Fig.1; Layout of MEMMOD2 board

Construction Details

Equipment Required; 1).Solder paste and a hot-air gun OR

Lower wattage soldering iron with a tip 1mm or smaller, and 22 SWG or finer solder.

2).Standard Anti-static measures; wrist band and conductive work-bench

Construction method

Question 1; are your hands steady???...if not, go have a pint before starting the construction procedure.

Question 2; are you working on an anti-static bench, and have the wrist band on and connected - remember, if you zap the chip after you've started soldering, the whole assembly is scrap!

First check the PCB for burrs, broken tracks and short-circuits (Find them now or the assembly will be useless and irreparable),them polish up copper surface with a polishing block or very fine glasspaper. Check that the PCB will slide at least half way up the pins of the 28 way socket, then remove.

The first component to be soldered onto the board is the Memory IC; check no pins are bent. If you are using solder paste and a hot air gun, then follow the procedure laid out in the gun's manual. If you are stuck with the good ol' hot poker, follow instructions in the next paragraph.

Remove the IC from it's protection and place on the board in the correct position. Is it the correct way around??..CHECK!. Then use a bulldog clip or bent paper clip to hold the chip to the board, while you solder ONE pin in place. Gently remove the clip and check that all the pins line up, before soldering the rest of then.

REMEMBER..Hold the soldering iron on each pin for less than 3 seconds.

After this is done,solder wire through the 4 vias, the surface mount capacitor to the underside on the board, the 2 wires for /WR and /RD to the top of the board, the 3 way jumper in the bottom row of 3 holes facing up, and the altered 2 way housing to the underside, aligned with the top row of holes on the board.

Lastly, fit the assembly onto the 28 pin socket, and push it up as far as it will go, before soldering both sides of each pin.

Now visually check the assembly for solder bridges (if you find any, don't try to clear then with the soldering iron - you'll cook the chip - instead use a sharp scalpel and patience.)

Item List

1. Memmod2 PCB;
2. SRAM Memory; Hitachi HM62256LFP10 (Farnell) or similar 32k x 8bit, 28 pin Plastic SOP package
3. 28 pin Socket; Harwin Part No. D75 028-01 (Farnell P739 code 176-368) or similar Extended socket with turned pin contacts
4. SMT Capacitor; Kemet type C1206 100nF 50V Z5U 20% Cap (STC code 030757A) or similar surface mount style 1206
5. 3 Pin jumper; 3 way 0.1" pin Jumper
6. 2 Way shell; 0.1" connector shell cut down to 5mm depth
7. 2 skt pins; bent 90° 4mm from end
8. 2 pieces wire; to connect /RD and /WR

PCB layout

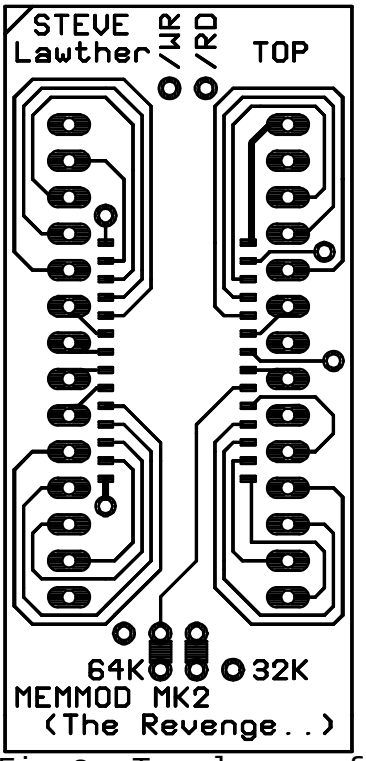

Fig.2; Top layer of MEMMOD2 PCB

Both viewed from above (Chip side)

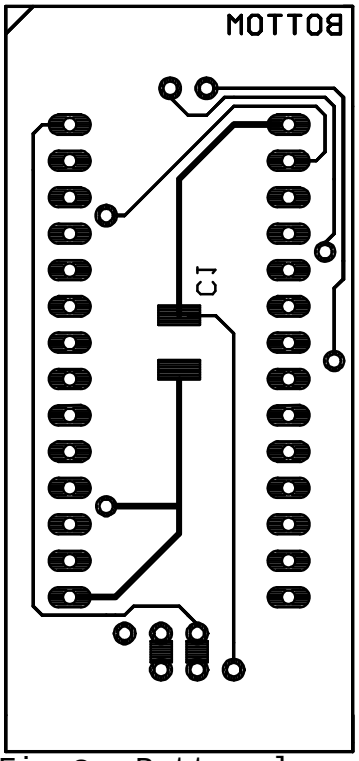

Fig.3; Bottom layer of MEMMOD pcb

Example Program

```
; MAIN TRANSDUCTOR READING,STORING AND PROCESSING ROUTINE

; stores raw data in external memory addressed by DPTR
; stores distances to probable objects in internal memory
; addressed by R1 - to get actual distance of object in
; centimetres, multiply stored value by 0.43 (assuming
; speed of sound to be 344m/s i.e. temp = 20deg.C)
; stores number of overflow values counted, in R4

;DPTR - has address where 1st byte of raw data is to be stored
;          Must have DPL = 0 therefore start DPTR value = XX00h
;R1     - has address (in internal RAM) where the first object
;          distance number is to be placed - must be 128 bytes away
;          from top of internal memory to save fault conditions from
;          overwriting the 1st register bank.(but see note at end)
;          ASSUMING R1 = 128 at mo.
;R2     - register used to check if there is a dip from overflow or
;          that a new overflow pulse is on the end of another
;R3     - store for the last value  from the A/D, for comparasion
;          to present value
;R4     - stores the number of times the value was overflow
;R6     - general temporary use ie wait routines-doesn't need to be
;          set to any particular value
;R7     - error number.. of binary code X0010NNN where
;          X indicates whether error is fatal (X=1) or not
;          0010 indicates the UltraSonic reading routine
;          NNN indicates the exact error
;
;NOT altered - R0,R5
;
;NOTE - any register bank can be used (as far as i can see!!!)

        MOV R4,#0

MAIN_STORE_LOOP:
;1st read value of a/d from port 1 and store it in external
;memory for further processing,

        MOV A,P1                    1       1
        MOVX @DPTR,A                2       3
```

```
;next, Is value an overflow, ie bit 6 (E6) of ACC set?? - If not
;jump to NO_OVER

        JNB ACC.6, NO_OVER          2       5

;now in overflow routine....1st increase R4 (no. of overflows
;counted), then check if it is a new overflow pulse (R2=0) or
;just a continuation of one already logged (R2>0)

        INC R4                      1       6
        MOV A,R2                    1       7
        JNZ SAME_OVER               2       9

;it is a new overflow pulse,so note that there is a pulse
;overflowing (by setting R2 to 2), then wait 2 ticks so the
;total cycle time is 25, then log it's position by jumping
;to RECORD_POSN below (saves repeating the code)

        MOV R2,#2                   1       10
        NOP                         1       11
        NOP                         1       12
        SJMP RECORD_POSN            2       14

SAME_OVER:
;just the same pulse overflowing...so no more to do except
;to note this (in R2) then wait a while so that the whole loop
;equals 25 cycles,increase the memory position, then run next
;loop..unless that was the last

        MOV R2,#2                   1       10
        MOV R6,#4                   2       12
        DJNZ R6,$                   2       14,16,18,20
        NOP                         1       21
        INC DPTR                    2       23
        DJNZ R0,MAIN_STORE_LOOP     2       25
        SJMP END_STORE_LOOP         -       --
```

```
NO_OVER:
;Ok, so the value from the A/D is 0-63,so first decrease R2,
;which indicates whether it is just a slight dip from overflow
;or is the end of any pulse (R2=0). All this time the a/d value
;is safe in R2's cubby hole!

        XCH A,R2                        1       6
        CLR C                   1       7
        RRC A                   1       8
        XCH R2,A                        1       9

;now check if the a/d value is more than the last value,
;indicating a new pulse edge. If not jump to NO_OBJECT
;note that the xch operand also stores the present value in R3

        XCH A,R3                        1       10
        CLR C                   1       11
        SUBB A,R3                       1       12
        JNC NO_OBJECT                   2       14

RECORD_POSN
;ok, so we have a new pulse edge (or something!), so record the
;position...check if the pulse caused the position to be recorded ;last cycle (in memory address
R1)..If not jump to DIFFER_PULSE

        MOV A,@R1               1       15
        INC A                   1       16
        CJNE A,DPL,DIFFER_PULSE         2       18

;right, so it's part of the same pulse edge, so update the value
;already stored and wait til the cycle is up

        MOV @R1,A               1       19
        NOP                             1       20
        NOP                             1       21
        INC DPTR                        2       23
        DJNZ R0,MAIN_STORE_LOOP         2       25
        SJMP END_STORE_LOOP             -       --
```

```
DIFFER_PULSE:
;it's a different pulse so the memory location in R1 needs to be
;increased and the position stored

        INC R1                          1       19
        MOV @R1,DPL                     2       21
        INC DPTR                        2       23
        DJNZ R0,MAIN_STORE_LOOP         2       25
        SJMP END_STORE_LOOP             -       --
NO_OBJECT:
;it's objects you're looking for, so if there isn't any evidence
;of one this cycle, then waste time 'til the next cycle

        MOV R6,#3                       1       15
        DJNZ R6,$                       2       17,19,21
        INC DPTR                        2       23
        DJNZ R0,MAIN_STORE_LOOP         2       25

END_STORE_LOOP:
;ok, all the rushing about is over - no worries about timing now
;Time for a sanity check on the values.
; 1st... there shouldn't be more than 25 objects counted (say)
;If there is, thats a fatal error as other info. in internal
; memory between 155 (say) and 255 might have been overwritten
;therefore place the error number 91h in R7 and jump to END

        MOV R7,10010001B
        MOV A,R1
        CLR C
        SUBB A,#128
;number in A is number of objects recorded
        CLR C
        SUBB A,#25
        JNC END

;2nd, there shouldn't be overflow more than 100 times (say)
;otherwise sometime's up, therefore give error 12h - nonfatal

        MOV R7,00010010B
        MOV A,R4
        CLR C
        SUBB A,#100
        JNC END   ;3rd, etc.....
```

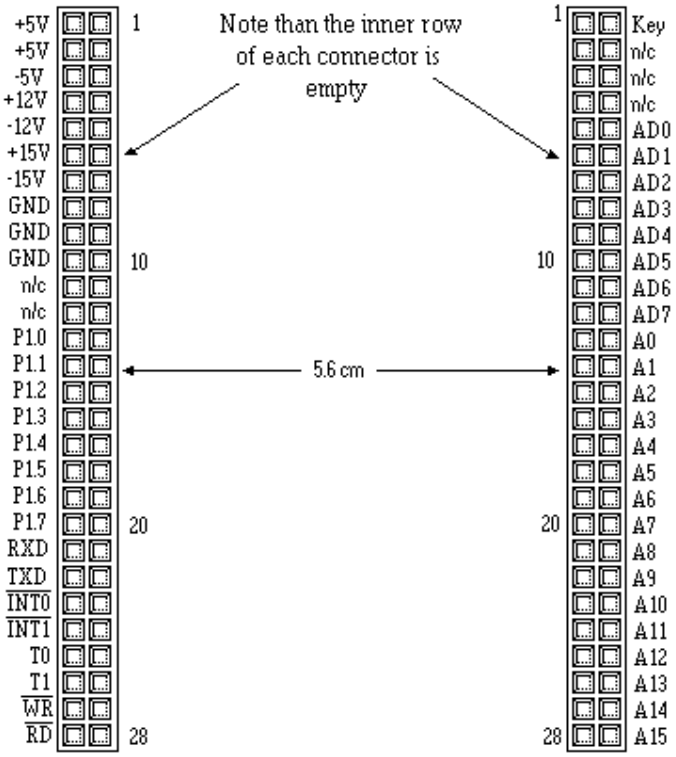

Figure 1 - Pinout of mobot spine