Controlling air core meters with the 87C751 and SA5775 AN426

INTRODUCTION

Often, certain classes of microcontroller applications surface where large amounts of on-chip resources such as a large program memory space and numerous I/O pins are not required. These applications are typically cost sensitive and desirable attributes of the MCU include low cost and modest on-chip resources such as program and data memory, I/O, and timer-counters. Substantial benefits of reduced design cycle time can be realized by using an industry-standard architecture having software compatibility with existing popular microcontrollers.

THE 87C751

The Philips 87C751 is one such microcontroller that easily meets these requirements. This device, shown in Figure 1, has a 2k x 8 program memory, 64 bytes of RAM, 19 parallel I/O lines, and a 16-bit autoreload timer-counter. It also includes an I^{2}C serial interface and a fixed rate timer. The 87C751 is based on the 80C51 core and thus uses an industry-standard architecture and instruction set. The device is available in both ROM (83C751) and EPROM (87C751) versions. The EPROM version is available in both UV erasable and OTP packages. References to the 87C751 in this document also apply to the 83C751, unless explicitly stated.

TYPICAL APPLICATION

A typical example of such an application is the interface between the 87C751 and the Philips SA5775 Serial Gauge Driver, SGD, shown in Figure 2. This circuit includes the 87C751 microcontroller, the SA5775 Serial Gauge Driver, an NE555 timer, and discrete support components.

An air core meter differs from a conventional (d'Arsonval) meter movement in that it has no spring to return the needle to a predetermined position, no zeroing adjustment, and no permanent magnet in the classical sense. Instead, it consists of two coils of wire wound in quadrature with each other around a central core in which there is a disc magnetized along its diameter. A shaft is placed through the center of this disc so that the shaft rotates with the disc. An indicating needle attached to this shaft will rotate with it.

SA5775 Serial Gauge Driver

The SA5775 is a monolithic driver for controlling air core meters typically used in automotive instrument clusters and is shown in Figure 3. The SA5775 receives a 10-bit serial word and converts that word to four voltage outputs that appear at the SINE+, SINE–, COSINE+, and COSINE– outputs. The differential voltage at the SINE outputs are applied to one coil of the meter and the COSINE outputs are applied to the other coil of the meter.

The currents through these coils produce a resultant magnetic force which is the vector sum of the magnetic forces produced by each of the two coils. Since the currents through the coils are bidirectional this magnetic vector can rotate through a full 360 degrees. The magnetized disc within the air core meter will follow the rotating vector and the needle will indicate the vector's current position. Since 10 bits are used, there are 1024 discrete words available resulting in an angular displacement of 0.3516 degrees per bit. This is small enough to provide an apparently smooth movement of the needle. The smoothness of the motion will depend greatly on the damping factor of the meter movement.

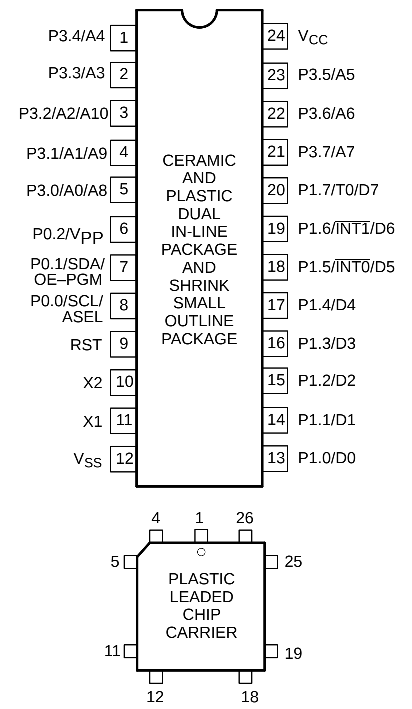
A simplified block diagram of the SA5775 is shown in Figure 4. This device consists of a serial-in/parallel-out shift register, a data latch, a D/A converter, a multiplexer, and output buffers.

A logic high must be present on the chip select (CS) input to clock in the data. Data appearing on the data input (DI) pin is clocked into the shift register on the rising edge of the clock (CLK) input. The data output (DO) pin is the overflow from the shift register, allowing the user to daisy chain multiple SA5775 devices. Note that data is clocked out of this pin on the falling edge of the clock. The CS pin is also used to latch the parallel outputs of the shift register into the data latch. The outputs of the data latch feed the inputs to the D/A converter. The D/A converter outputs are buffered to form the drive signals for the meter coils.

The D/A converter circuits, multiplexer and associated output buffers are purposely designed such that the span of these circuits do not include the power supply rails. This is to avoid inaccuracies that would otherwise occur if the output were to become very close to either supply rail. With a supply voltage of 14 volts (VIGN), the outputs will span a range of approximately 1 to 11 volts. The SA5775 is designed to drive air core meters having a minimum winding impedance of 180Ω at –40°C.

The clock high and low time requirements are 175ns minimum and the maximum data rate is 1.6 megabits per second. At this rate it would require approximately 6.4ms to ramp from zero to full scale if all binary codes were loaded into the SA5775. However, the air core meter cannot respond to such data rates. Both inertia of the movement and damping build into the design of typical air core meter movements limit their response speed.

A high on the output enable input pin (OE) is required to permit the SA5775 to drive the air core gauge. In Figure , OE is held low while the microcontroller is being reset to prevent the gauge from being driven.



Pin	Function	Pin	Function
1	P3.4/A4	15	P1.0/D0
2	P3.3/A3	16	P1.1/D1
3	P3.2/A2/A10	17	P1.2/D2
4	P3.1/A1/A9	18	P1.3/D3
5	NC*	19	P1.4/D4
6	P3.0/A0/A8	20	P1.5/$\overline{\text{INT0}}$/D5
7	P0.2/V_{PP}	21	NC*
8	P0.1/SDA/OE-PGM	22	NC*
9	P0.0//SCLASEL	23	P1.6/$\overline{\text{INT1}}$/D6
10	NC*	24	P1.7/T0/D7
11	RST	25	P3.7/A7
12	X2	26	P3.6/A6
13	X1	27	P3.5/A5
14	V_{SS}	28	V_{CC}

* DO NOT CONNECT

SU00315

Figure 1. Pin Configuration

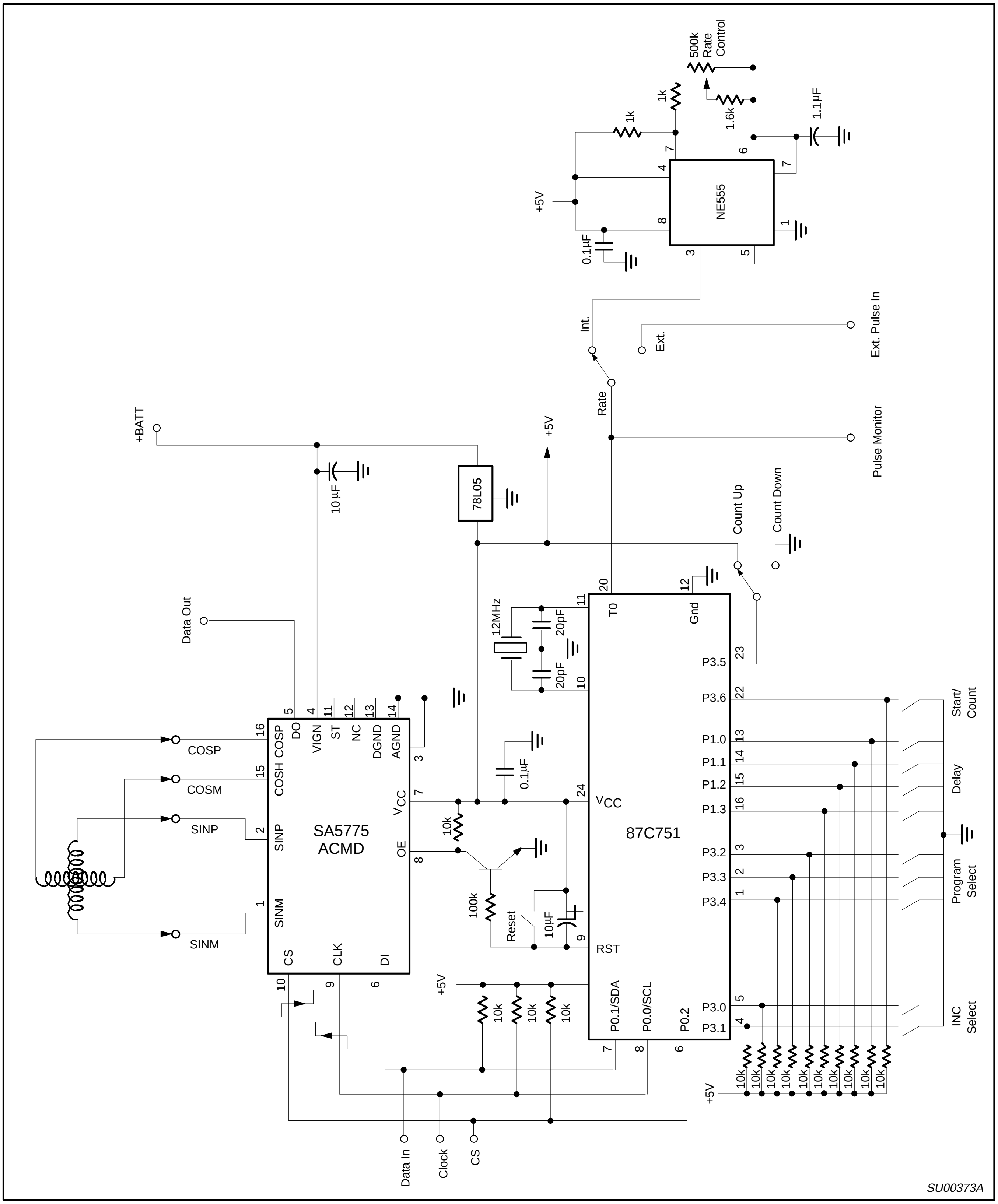
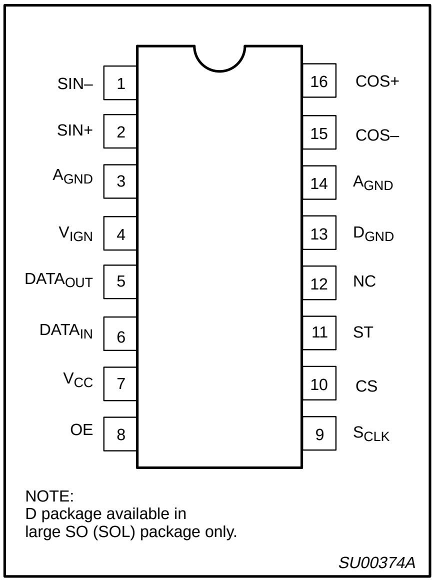


Figure 2. Interface Between the 87C751 and the Philips SA5775



NOTE:
D package available in
large SO (SOL) package only.

SU00374A

Figure 3. D and N Packages

87C751 Microcontroller

The 87C751 microcontroller provides all of the intelligence in this application. It samples various input ports to determine which demonstration programs to run, the incremental step sizes for angular displacement of the meter core, and the time delay between increments. In one of the demonstration modes, it also samples a variable frequency input and positions the meter core in response to the frequency of that input. The 87C751 also transmits the 10-bit serial data to the SA5775. Data input (DI), Clock (CLK), and Chip Select (CS) lines are driven from the 87C751.

Port 0 of the 87C751 is a 3-bit wide port and is used for communicating data to the SGD. Data is transmitted, MSB first, in a serial stream clocked into the DI of the SA5775 on the rising edge of the clock. In order to clock in data, the CS pin of the SA5775 must be high. The data in the input register is shifted into a latch that drives the DAC on the high to low transition of the CS line. As data is shifted into the SGD, it overflows through the Data Out (DO) pin on the falling edge of the clock. With this facility, multiple SGDs can be daisy-drained with DO of one SGD being connected to DI of the next one, and common clock and chip select lines may be used. This simplifies the interfacing to multiple meter drivers.

The 78L05 regulator (Q2) provides 5 Volt power for the board so that single supply of +14 volts can be applied to the board.

Three rotary switches are used on this board. The PROGRAM SELECT switch (S3) is used to select the program routine that is executed, the INC SELECT (S2) switch selects the incremental step sizes of two of the routines, and the DELAY switch (S4) is used to set the delay between successive word transmissions in one of the routines.

The START/COUNT button (S5) is used to begin execution of a routine, and to cause the next incremental step in Routine #1.

The COUNT UP/DOWN switch (S6) is used in Routine #1 to determine whether the count is increased or decreased with transmission of successive words.

NE555 Timer

The NE555 timer shown in this application example is used as a free running squarewave generator used to simulate sensor inputs such as those which might be found in an automobile, etc. The NE555 timer (U4) operates in the astable mode to produce an output frequency that can be varied from about 1Hz to about 200 Hz. Three of the program routines measure the input period and produce an output code that is proportional to the frequency present at pin 20 (TO) of the microcontroller. A RATE switch (S7) is used to select between the on board oscillator or an external source.

The program listing is included at the end of this application note.

Program Entry

The program starts at address 030(hex) on line 21 of the program listing. The first task is to write 1's to all pins of each port.

Lines 25 and 26 clear registers 6 and 7. These registers are used in this program only to hold the data that is sent out to the SGD. The registers are cleared to be sure that the starting value is zero.

At line 27 the program waits until the START/COUNT button (S5) is depressed before continuing. Lines 28 and 29 set the timer to overflow after 10ms. This is done by setting the timer registers for a count of 10,000 microseconds less than full scale. When the timer counter overflows the timer flag is set, and the timer is reloaded with the value in the timer register. By examining the timer flag we know when 10ms has expired.

Line 30 calls subroutine RPS (Read Port Selected), which reads Port 3 to determine which routine has been selected. Since the PROGRAM SELECT switch (S3) is connected to port pins P3.2 through P3.4, subroutine RPS (lines 507 through 511 at the end of the program) first reads Port 3 into the accumulator, then complements it because the switches used are complementary binary. The reading is then rotated right once and the upper nibble and the LSB (least significant bit) are masked off, leaving twice the value of the port selected in the accumulator. Twice the read value is needed for the next few main program lines that determine which routine to execute.

Line 31 moves the address of label JMPTBL (Jump Table) to the 16-bit Data Pointer (DPTR) register. Line 32 causes a program jump to the address that is the sum of the value in the accumulator (two times the routine number selected) plus the DPTR register. Since each of the commands on lines 33 through 40 are two byte commands, these addresses are all separated by two bytes; hence, the need for the accumulator to contain a number that is twice the number of the selected routine.

Routine 0

This routine begins on line 41 by incrementing the 10-bit word in registers 7 and 6 by the amount indicated by the setting of the INCREMENT SELECT switch, then sending that word to the SA5775. When a full scale overflow is detected, a full scale code (3FF hex) is sent out, followed by a delay of 500 ms, then successive output codes are sent out, decremented by an amount indicated by the INCREMENT SELECT switch. When an underflow is detected a code of zero scale is sent and the routine returns to the beginning of the program. This routine is implemented with a series of subroutine calls.

The SO subroutine begins on line 356 and starts by sending out whatever ten bits that in the two LSBs of register 7 (R7) plus the 8 bits of R6 by calling the SENDIT subroutine. Then it calls the UP subroutine, which increases the word value to be sent out. The program then jumps to the beginning of this subroutine, repeating the process of sending out a word and incrementing to the next word until an overflow from the tenth bit (bit 2 of R7) is detected at line 362.

The SENDIT subroutine (beginning on line 476) brings the CS line high, sets a bit counter (R1) to 2 (to send out two bits of R7), brings the value of R7 to the accumulator, rotates the accumulator to the right three times through the carry bit to bring the two LSBs to the position of the two MSBs, calls the SEND1 routine, which sends the number of bits in the accumulator, starting with the MSB, indicated by R1. Counter R1 is then set to 8 to send out all 8 bit of R6 and the accumulator is loaded with the contents of R6. The SEND1 routine is again called to send out the final 8 bits, and, on line 491, the CS line is brought low, loading the SA5775 internal parallel latch with the contents of the input shift register.

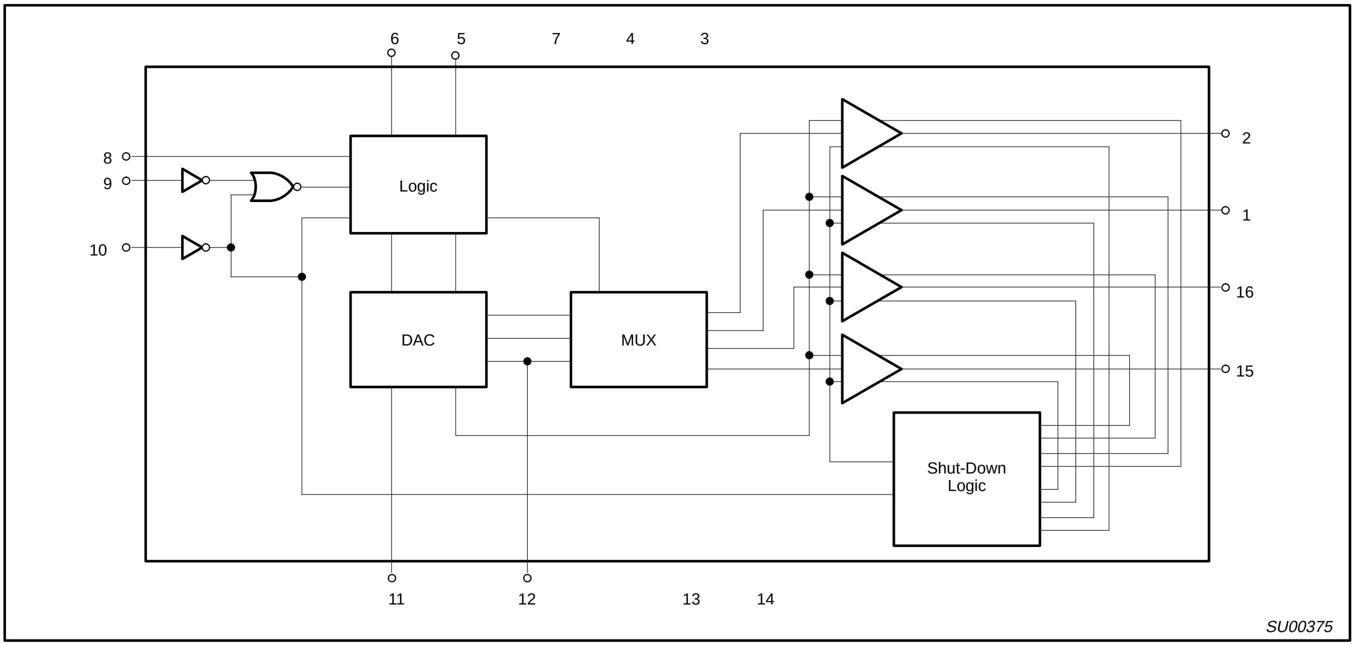


Figure 4. Block Diagram of the SA5775

The SEND1 routine rotates the accumulator left through the carry bit, moves the value of the carry bit to port pin PO.1 (SDA—Serial Data pin), waits to provide a setup time, brings the clock low, waits, brings the clock high, waits, then decrements bit counter sends the next bit if the counter is not zero. A return is executed when the counter becomes zero.

The UP subroutine, beginning at line 364, reads the delay selected by switch S4 at port pin P1, complements it (again, because the rotary switches are complementary binary), masks off the upper four bits (because the delay switch has just four positions and is connected to the lower four bits of the port), multiplies it by 4 (rotates left twice), then moves the result to R1. If R1 is not zero, the program jumps around line 376 and calls a 10ms delay (subroutine DLY10MS) the number of times entered into R1.

The 10ms delay subroutine (starting at line 436) sets the timer for 10ms, waits at line 446 for the timer flag to be set, clears the timer flag, stops the timer, and returns, in this case, to line 379, where the program decrements R1 and repeats the 10ms delay until R1 is zero.

If the selected delay was zero, the program jumps from line 376 to line 380 and reads port 3 to determine the amount the sent out word is to change from the value previously sent out. The accumulator is complemented and the upper 6 bits masked off to recover only the two bits of the selected increment amount. Since increments of 1, 2, 3, or 4 LSBs are hardly noticeable, the program then multiplies the result by 8 (rotate left three times). To insure a minimum change amount, the accumulator is increment by one at line 386. This all means that the increment amounts that can be selected are 1, 9, 17, or 25 LSBs. This amount is added, in lines 387 through 391, to the word previously send out and we return from this subroutine.

After calling the S0 subroutine, PR0GO call the FULLSC (full scale) subroutine, which sends out the full scale code of 3E8(hex). Although a 10-bit full scale code would be 3FF(hex), going only to 3E8 allows an easy distinction between zero scale and full scale when looking at the display. The FULLSC subroutine is found at line 352.

After advancing to full scale, there is a 500ms delay, found at line 464 and called from line 48, then 49 calls the S0D subroutine to send out decreasing word values.

The S0D subroutine begins at line 393 and begins by sending out the current word in R7 and R6 from line 398, then calling subroutine DOWN, which calculates the next (decreasing) word to send out. DOWN begins at line 402. It essentially does the same thing as the UP subroutine, but subtracts the INCREMENT SELECT value from the previously sent word rather than adding to it.

At line 50 subroutine ZEROSC is called to send a zero scale code to the SA5775, then the program branches back to the beginning.

Routine 1

This routine is selected with the PROGRAM SELECT switch is in position 1 or position 9. Routine 1 (PROG1) increments or decrements the word send out, depending upon the setting of the COUNT UP/COUNT DOWN switch, S6. The amount of change is determined by the setting of the INC SELECT switch, S2.

At line 63, the program examines S6 at port pin P3.6 and jumps to the decrement portion of the routine if the pin is low. If this pin is high, the UP subroutine is called from line 64 to increase the R7/R6 word value. The UP subroutine was previously described.

If pin P3.6 is low, the DOWN subroutine (line 402) decreases the previous word sent out by the amount determined from the INC SELECT switch setting.

To insure enough delay to allow the user time to release the START/COUNT button (S5), a delay of 200ms is included at line 66 before jumping to line 27, where another depression of the START/COUNT button is awaited. If S3 (PROGRAM SELECT) is still set to 1 or 9, depression of S5 will cause a jump back to line 52. If another program is selected, the program will jump to the selected routine.

Holding down S5 with PROGRAM SELECT set at position 1 or 9 will cause increasing or decreasing word values to be sent to the SA5775.

Routine 2
PROG2 is the most complex of all these routines. The purpose of this routine is to cause the air core meter deflection to represent the frequency presented at the timer/counter input to the microcontroller. This is done by measuring the period of the input square wave and taking the inverse of the period. The input here must be a square wave because a slow rise and fall time at this input will cause fluctuating readings. To determine the frequency by counting pulses for a time would require a much longer time and, therefore, is impractical.

The MEAS (measure) subroutine is called at line 79 to measure the period of the input waveform and the CALC (calculate) subroutine is called at line 80 to calculate the code to send to the SA5775. The SENDIT subroutine is then called to send the word to the SA5775 and the program jumps back to line 28.

The MEAS subroutine begins at line 83 by being sure the timer is not running and clearing the timer (overflow) flag, then entering zero into both high and low bytes of the timer and the timer register. The carry bit is then cleared (line 90) and the timer started and the timer interrupt enabled.

Lines 93 and 94 form a short loop that waits until either the carry bit is set or until the TO input is low. The carry bit is set when the timer has gone beyond one second. This is done by the timer interrupt subroutine, found at lines 16 through 19. If the TO input never goes low, we know the frequency is at or near zero and the program jumps to GZS (line 108) where R3 is loaded with a 1F (hex) to cause the CALC subroutine to load zero scale into R7/R6.

When (and if) TO is found to be low, the program jumps to line 95 and waits for that input to go high. Time out process is the same as above.

Now that the TO input is found high (if is is before the one second time out), the timer and carry bit are cleared in lines 97 through 100 (R3 is an extension of the timer).

At lines 101 through 107 we wait for one complete cycle at the TO input, with the timer/counter measuring that period, then return to line 80, where the CALC subroutine is called.

The CALC subroutine, starting at line 113, begins by initializing the word to send out (R7/R6) to zero, clearing the carry bit, checking to see if R3 indicates a time above one second, returning to line 81 if it does. Otherwise the program continues at line 26, where the program checks to see if the input frequency is beyond full scale (timer reading above 00 12 88 hex). If it is, R7/R6 is loaded with 12 88 hex (full scale of decimal 1,000). This value was chosen because it is sufficiently far from zero scale that it is easily discerned from zero scale on the display.

If the result is not to be full scale or zero scale, the program continues at line 140 with a shift and subtract divide routine. The dividend would be 1,000,000 (decimal) to convert back to frequency in Hertz (period measurements is in microseconds), but that would provide a maximum count of 200 at 200Hz, only one fifth of the full scale desired of 1,000. So we made the dividend to be 5,000,000 decimal, or 4C 4B 40 hex.

This algorithm is found in lines 156 through 192 and works as follows:

1. Clear a counter.
2. Rotate dividend until the first one is in the second MSB position. Since a code of 4C has already provides that, no shifting is necessary.
3. Rotate the divisor (the period in microseconds in this case) left until the first one is in the second MSB position, but the first byte is LESS THAN the first byte of the dividend. Increment the counter each time the divisor is rotated.
4. Initialize a counter to zero.
5. Rotate the quotient (answer) and dividend one bit left.
6. If first byte of quotient is smaller than the first byte of the quotient, jump to step 8.
7. Add one to the quotient and subtract the divisor from the dividend.
8. Decrement the counter and go to step 5 if it is not zero.

Once the CALC subroutine is completed, the program calls SENDIT from line 81 and jumps, ultimately, to the selected routine.

Routine 3
PROG3, beginning at line 194, measures the input period four times, then calculates the code to display that is the average of these four readings.

It starts by setting a counter for three readings, taking those three readings and storing them in memory, beginning at RAM address 20 hex, using register RO as an index register.

At line 212 the program takes a fourth reading, then adds the three previous readings to it in lines 213 through 227; and divides the sum by four (rotates right twice) in lines 229 through 239. The word to send out is then calculated from line 240 and sent to the SGD, after which the program then looks for and jumps to the selected routine.

Routine 4
PROG4 begins at line 243 and displays the average of the current and last three words sent out.

RAM space used is first initialized to zero and a new reading is taken and a new word is calculated and saved. At lines 264 through 284, the new word is added to the last three readings and the average calculated and stored in RAM locations 28 and 29 (hex), and the average word is sent out.

At line 286, the program reads for the program selected and jumps to line 254 if this routine is selected, otherwise it goes to line 28.

Routine 5
PROG5 begins at line 293 and, very simply, send in sequence the codes for 1/8 through full scale in 1/8 scale steps, with 500ms between steps. It then steps down to zero scale in 1/8 scale steps, then returns to line 28.

Routine 6
PROG6 begins at line 314 and does the same as PROG5, but steps in 1/4 scale increments.

Routine 7
PROG7 loads the code for 3/8 scale into R7/R6, sends it, waits 500ms, changes r& for 5/8 scale, sends it, waits for 500ms, then repeats this sequence 9 more times (for a total of ten times), waits 500ms, then returns the output to zero scale and the program jumps to line 28.

```
              ;                                SGD V3 DEMO                  TT.20
              ;                            PROCESSOR:  87C751
              ;                                                             7–29–89
              ;
              ;   The purpose of this program is to drive version 3 of the SGD (SA5775)
              ;   demonstration board.  The PROGRAM SELECT switch is used to select from
              ;   a choice of four routines.  Registers R7 and R6 contain the 10–bit word
              ;   that is send to the SA5775.
              ;
              $MOD751
0000                 ORG    0
              ;
0000 B02E            SJMP   START             ;RESET VECTOR
              ;
000B                 ORG    00BH              ;TIMER/COUNTER INTERRUPT ROUTINE
000B 0B              INC    R3                ;INCREMENT R3 (3rd BYTE OF TIMER)
000C 740F            MOV    A,#0FH            ;TEST FOR TIME OUT (R3 > 0F)
000E 9B              SUBB   A,R3              ;IF R3 > 0F, CARRY IS SET
000F 32              RETI
              ;
0030                 ORG    30H               ;START OF PROGRAM
0030 7580FF   START: MOV    P0,#0FFH          ;SET PORTS HIGH
0033 7590FF          MOV    P1,#0FFH
0036 75B0FF          MOV    P3,#0FFH
0039 7F00            MOV    R7,#0             ;CLEAR WORD TO SEND OUT
003B 7E00            MOV    R6,#0
003D 20B6FD   W:     JB     P3.6,W            ;WAIT FOR START BUTTON DEPRESS
0040 758BF0   READY: MOV    RTL,#LOW(0–10000) ;SET TIMER REGISTER
0043 758DD8          MOV    RTH,#HIGH(0–10000);FOR 10ms TIME
0046 51D2            ACALL  RPS               ;READ PORT 3 FOR PROG SELECT
0048 90004C          MOV    DPTR,#JMPTBL      ;JMP ADDRESS TO DATA POINTER
004B 73              JMP    @A+DPTR           ;GOTO APPROPRIATE ROUTINE
004C 015C     JMPTBL:AJMP   PROG0             ;RAMP UP AND BACK DOWN
004E 0168            AJMP   PROG1             ;STEP UP/DOWN W/ start PRESS
0050 017A            AJMP   PROG2             ;READ & DISPLAY SPEED
0052 2145            AJMP   PROG3             ;DISPLAY AVERAGE OF 4 NEW READINGS
0054 2186            AJMP   PROG4             ;DISPLAY AVERAGE OF LAST 4 READINGS
0056 21D3            AJMP   PROG5             ;ADVANCE TO FULL SCALE AND BACK IN 45 DEGREE STEPS
0058 21F3            AJMP   PROG6             ;ADVANCE TO FULL SCALE AND BACK IN 90 DEGREE STEPS
005A 4107            AJMP   PROG7             ;ALTERNATE DISPLAY BETWEEN 3/8 AND 5/8 SCALE TEN TIMES
005C          PROG0:
              ;   This routine increases word sent at the selected step size (INCREMENT SELECT)
              ;   and delay time (DELAY), up to full scale, waits 500ms, then decreases the
              ;   word sent at the selected step size and delay times until zero scale is reached.

005C 5128            ACALL  SO                ;SEND OUT INCREASING WORDS
005E 5121            ACALL  FULLSC            ;SET TO FULL SCALE
0060 51A5            ACALL  DLY500            ;WAIT 500ms
0062 5152            ACALL  SOD               ;SEND OUT DECREASING WORDS
0064 511B            ACALL  ZEROSC            ;RESET TO ZERO SCALE
0066 0130            AJMP   START             ;GO TO BEGINNING OF PROGRAM
006B          PROG1:
              ;
              ;        MANUAL INCREMENT/DECREMENT ROUTINE
              ;
              ;   This routine increases or decreases the sent out word, depending upon
              ;   the setting of the UP/DOWN switch, by an amount set by the INCREMENT
              ;   SELECT switch.  There is a wait of 200ms before again looking for
              ;   depression of the START/COUNT button to allow time to release this
              ;   button and switch bounce to settle.  The program then looks to see which
              ;   routine is selected and goes to that routine.
              ;
0068 30B50B          JNB    P3.5,DCX          ;GO AND COUNT DOWN IF SELECTED
006B 5130            ACALL  UP                ;INCREASE WORD
006D 51B5     DP1:   ACALL  SENDIT            ;SEND THE WORD
006F 519D            ACALL  DLY200            ;WAIT 200ms
0071 013D            AJMP   W                 ;WAIT FOR COUNT BUTTON DEPRESS & SELECTED ROUTINE
0073 20B5F2   DCX:   JB     P3.5,PROG1        ;GO AND COUNT UP IF SELECTED
0076 515A            ACALL  DOWN              ;DECREASE WORD
```

```
0078 80F3            SJMP   DP1
007A         PROG2:
             ;
             ;        READ TIME INPUT AND DISPLAY "SPEED"
             ;
             ;   This routine measures the period of the square wave at the T0 input and
             ;   sends out a word that is inversely proportional to 5 times that period,
             ;   providing a display proportional to frequency.
             ;
007A 1182            ACALL  MEAS                ;MEASURE THE INPUT PERIOD
007C 11C5            ACALL  CALC                ;CALCULATE THE WORD TO SEND
007E 51B5            ACALL  SENDIT              ;SEND OUT THE WORD
0080 0140            AJMP   READY
0082 C28C    MEAS:   CLR    TR                  ;HALT TIMER
0084 C28D            CLR    TF                  ;CLEAR TIMER FLAG
0086 758B00          MOV    RTL,#0              ;SET TIMER REGISTERS
0089 758D00          MOV    RTH,#0
008C 758A00          MOV    TL,#0               ;SET TIMER
008F 758C00          MOV    TH,#0
0092 7B00            MOV    R3,#0               ;CLEAR TIMER 3RD BYTE
0094 C3              CLR    C
0095 D28C            SETB   TR                  ;START TIMER
0097 75A882          MOV    IE,#82H             ;ENABLE TIMER INTERRUPT
009A 4021    W20:    JC     GZS                 ;JUMP IF R3 > 0F
009C 2097FB          JB     P1.7,W20            ;WAIT FOR T0 INPUT LOW
009F 401C    W21:    JC     GZS                 ;JUMP IF R3 > 0F
00A1 3097FB          JNB    P1.7,W21            ;WAIT FOR T0 INPUT HIGH
00A4 758A00          MOV    TL,#0               ;RESET TIMER
00A7 758C00          MOV    TH,#0
00AA 7B00            MOV    R3,#0
00AC C3              CLR    C                   ;CLEAR CARRY/BORROW
00AD 4008    W22:    JC     HT                  ;JUMP IF TIME UP (CARRY SET)
00AF 2097FB          JB     P1.7,W22            ;WAIT FOR T0 LOW
00B2 4003    W23:    JC     HT                  ;JUMP IF TIME UP (CARRY SET)
00B4 3097FB          JNB    P1.7,W23            ;WAIT FOR T0 HIGH AGAIN
00B7 C28C    HT:     CLR    TR                  ;HALT TIMER
00B9 75A800          MOV    IE,#0               ;DISABLE ALL INTERRUPTS
00BC 22              RET
00BD 7B1F    GZS:    MOV    R3,#1FH             ;SET FOR ZERO SCALE
00BF 22              RET
00C0 7F03    GFS:    MOV    R7,#03
00C2 7EE8            MOV    R6,#0E8H
OOC4 22              RET
00C5         CALC:
             ;
             ;   This subroutine calculates the 10-bit word to send as a function fo what
             ;   is in R3, TH & TL.  The 10-bit word is developed and left in registers
             ;   R7 and R6 for use by SENDIT subroutine.
             ;
00C5 7F00            MOV    R7,#0               ;INITIALIZE QUOTIENT
00C7 7E00            MOV    R6,#0
00C9 C3              CLR    C                   ;CLEAR CARRY/BORROW
00CA 740F            MOV    A,#0FH              ;CHECK FOR ZERO SCALE
00CC 9B              SUBB   A,R3
00CD 5001            JNC    NZS                 ;JUMP IF NOT ZERO SCALE
00CF 22              RET
00D0 E58A    NZS:    MOV    A,TL                ;CHECK FOR FULL SCALE
00D2 9488            SUBB   A,#88H
00D4 E58C            MOV    A,TH
00D6 9413            SUBB   A,#13H
00D8 EB              MOV    A,R3
00D9 9400            SUBB   A,#0
00DB 40E3            JC     GFS
00DD 752E4C          MOV    2EH,#4CH            ;SET DIVIDEND TO 5,000,000
00E0 752F4B          MOV    2FH,#4BH
00E3 753040          MOV    30H,#40H
00E6 7C00            MOV    R4,#0               ;CLEAR DIVIDE COUNTER
00E8 8B2B            MOV    2BH,R3              ;MOVE READING TO MEMORY (DIVISOR)
00EA 858C2C          MOV    2CH,TH
```

```
00ED 858A2D          MOV    2DH,TL
00F0 C3      ROTL:   CLR    C                ;BRING DIVISOR BE JUST LESS THAN DIVIDEND
00F1 E52E            MOV    A,2EH
00F3 952B            SUBB   A,2BH
00F5 4014            JC     DIV24            ;JUMP IF SHIFTING WOULD MAKE DIVISOR > DIVIDEND
00F7 6012            JZ     DIV24            ;JUMP IF DIVISOR & DIVIDEND MS BYTES EQUAL BEFORE SHIFT
00F9 E52D            MOV    A,2DH            ;SHIFT DIVISOR TO LEFT
00FB 33              RLC    A
00FC F52D            MOV    2DH,A
00FE E52C            MOV    A,2CH
0100 33              RLC    A
0101 F52C            MOV    2CH,A
0103 E52B            MOV    A,2BH
0105 33              RLC    A
0106 F52B            MOV    2BH,A
0108 0C              INC    R4
0109 80E5            SJMP   ROTL
010B C3      DIV24:  CLR    C
010C EE              MOV    A,R6             ;ROTATE QUOTIENT LEFT
010D 33              RLC    A
010E FE              MOV    R6,A
010F EF              MOV    A,R7
0110 33              RLC    A
0111 FF              MOV    R7,A
0112 C3              CLR    C                ;ROTATE DIVIDEND LEFT
0113 E530            MOV    A,30H
0115 33              RLC    A
0116 F530            MOV    30H,A
0118 E52F            MOV    A,2FH
011A 33              RLC    A
011B F52F            MOV    2FH,A
011D E52E            MOV    A,2EH
011F 33              RLC    A
0120 F52E            MOV    2EH,A
0122 C3              CLR    C                ;TEST SUBTRACT MOST SIGNIFICANT BYTES
0123 952B            SUBB   A,2BH
0125 401B            JC     ZERO             ;JUMP IF QUOTIENT MS BYTE < DIVISOR MS BYTE
0127 7401            MOV    A,#1             ;ADD 1 TO QUOTIENT
0129 2E              ADD    A,R6
012A FE              MOV    R6,A
012B EF              MOV    A,R7
012C 3400            ADDC   A,#0
012E FF              MOV    R7,A
012F C3              CLR    C                ;SUBTRACT DIVISOR FROM DIVIDEND
0130 E530            MOV    A,30H
0132 952D            SUBB   A,2DH
0134 F530            MOV    30H,A
0136 E52F            MOV    A,2FH
0138 952C            SUBB   A,2CH
013A F52F            MOV    2FH,A
013C E52E            MOV    A,2EH
013E 952B            SUBB   A,2BH
0140 F52E            MOV    2EH,A
0142 DCC7    ZERO:   DJNZ   R4,DIV24
0144 22              RET
0145         PROG3:
             ;
             ;            DISPLAY AVERAGE OF FOUR NEW READINGS
             ;
             ;   This routine reads the period of the T0 input four times, then displays the
             ;   “speed” corresponding to the average of these four readings.
             ;
0145 7903            MOV    R1,#3            ;SET FOR 3 READINGS
0147 7820            MOV    R0,#20H          ;SET INDEX REGISTER FOR BOTTOM
0149 1182    P30:    ACALL  MEAS             ;TAKE 3 READINGS AND SAVE THEM
014B EB              MOV    A,R3
014C F6              MOV    @R0,A
014D 08              INC    R0
014E A68C            MOV    @R0,TH
```

```
0150 08               INC    R0
0151 A68A             MOV    @R0,TL
0153 08               INC    R0
0154 D9F3             DJNZ   R1,P30
0156 1182             ACALL  MEAS              ;TAKE A 4TH READING, LEAVING IN R3,TH,TL
0158 7828             MOV    R0,#28H           ;SET INDEX REGISTER FOR TOP
015A 7903             MOV    R1,#3             ;SET COUNTER TO ADD FIRST 3 READINGS TO LAST ONE
015C E58A     P31:    MOV    A,TL              ;ADD FIRST THREE READINGS TO THE LAST ONE
015E 26               ADD    A,@R0
015F F58A             MOV    TL,A
0161 18               DEC    R0
0162 E58C             MOV    A,TH
0164 36               ADDC   A,@R0
0165 F58C             MOV    TH,A
0167 18               DEC    R0
0168 EB               MOV    A,R3
0169 36               ADDC   A,@R0
016A FB               MOV    R3,A
016B 18               DEC    R0
016C D9EE             DJNZ   R1,P31
016E 7902             MOV    R1,#2
0170 EB       P32:    MOV    A,R3              ;DIVIDE BY 4 (ROTATE RIGHT TWICE) FOR AVERAGE
0171 C3               CLR    C
0172 13               RRC    A
0173 FB               MOV    R3,A
0174 E58C             MOV    A,TH
0176 13               RRC    A
0177 F58C             MOV    TH,A
0179 E58A             MOV    A,TL
017B 13               RRC    A
017C F58A             MOV    TL,A
017E D9F0             DJNZ   R1,P32
0180 11C5             ACALL  CALC              ;CALCULATE THE WORD
0182 51B5             ACALL  SENDIT            ;SEND OUT THE WORD
0184 0140             AJMP   READY             ;GO TO SELECTED ROUTINE
0186          PROG4:
              ;
              ;        DISPLAY AVERAGE OF LAST FOUR WORDS SENT OUT
              ;
              ;   This routine sends out the average of the last four readings sent out.
              ;
0186 7827             MOV    R0,#27H
0188 7600     P4:     MOV    @R0,#0
018A 18               DEC    R0
018B B81FFA           CJNE   R0,#1FH,P4
018E 7820     P4A:    MOV    R0,#20H
0190 1182     P40:    ACALL  MEAS              ;MEASURE PERIOD
0192 11C5             ACALL  CALC              ;CALCULATE THE CODE
0194 EF               MOV    A,R7              ;SAVE THE CODE
0195 F6               MOV    @R0,A
0196 08               INC    R0
0197 EE               MOV    A,R6
0198 F6               MOV    @R0,A
0199 752800           MOV    28H,#0            ;INITIALIZE THE WORD TO SEND
019C 752900           MOV    29H,#0
019F 7927             MOV    R1,#27H
01A1 E529     P41:    MOV    A,29H             ;ADD TOGETHER LAST 4 RESULTS
01A3 C3               CLR    C
01A4 27               ADD    A,@R1
01A5 F529             MOV    29H,A
01A7 E528             MOV    A,28H
01A9 19               DEC    R1
01AA 37               ADDC   A,@R1
01AB F528             MOV    28H,A
01AD 19               DEC    R1
01AE B91FF0           CJNE   R1,#1FH,P41
01B1 7902             MOV    R1,#2
01B3 C3       P42:    CLR    C
01B4 E528             MOV    A,28H
```

```
01B6 13                RRC    A
01B7 F528              MOV    28H,A
01B9 E529              MOV    A,29H
01BB 13                RRC    A
01BC F529              MOV    29H,A
01BE D9F3              DJNZ   R1,P42
01C0 AF28              MOV    R7,28H
01C2 AE29              MOV    R6,29H
01C4 51B5              ACALL  SENDIT             ;SEND OUT THE WORD
01C6 51D2              ACALL  RPS                ;READ PROGRAM SELECT
01C8 B40806            CJNE   A,#8,N4            ;JUMP TO N4 (& “READY”) IF PROGRAM 4 NOT SELECTED
01CB 08                INC    R0
01CC B828C1            CJNE   R0,#28H,P40        ;GOTO P40 IF R0 NOT 28 (HEX)
01CF 80BD              SJMP   P4A
01D1 0140      N4:     AJMP   READY
               ;
               PROG5:
               ;
               ;   This routine advances the display in 45 degree steps to full scale, then steps down
               ;   to zero in 45 degree steps.  There is a 500ms delay between steps.
               ;
01D3 7F00              MOV    R7,#0
01D5 7E7F      P5:     MOV    R6,#07FH
01D7 51B1              ACALL  SD500              ;SEND THE WORD AND WAIT 500ms
01D9 7EFF              MOV    R6,#0FFH
01DB 51B1              ACALL  SD500              ;SEND THE WORD AND WAIT 500ms
01DD 0F                INC    R7
01DE BF04F4            CJNE   R7,#4,P5
01E1 7F03              MOV    R7,#3
01E3 7EFF      LP5:    MOV    R6,#0FFH
01E5 51B1              ACALL  SD500              ;SEND THE WORD AND WAIT 500ms
01E7 7E7F              MOV    R6,#7FH
01E9 51B1              ACALL  SD500
01EB 1F                DEC    R7
01EC BFFFF4            CJNE   R7,#0FFH,LP5
01EF 511B              ACALL  ZEROSC             ;RETURN TO ZERO
01F1 013D              AJMP   W                  ;WAIT FOR KEY PRESS
01F3           PROG6:
               ;
               ;   This routine advances the display in 90 degree steps to full scale, then steps down
               ;   to zero in 90 degree steps.  There is a 500ms delay between steps.
               ;
01F3 7EFF              MOV    R6,#0FFH
01F5 7F00              MOV    R7,#0
01F7 51B1      LP6:    ACALL  SD500              ;SEND THE WORD AND WAIT 500ms
01F9 0F                INC    R7
01FA BF04FA            CJNE   R7,#4,LP6
01FD 1F        LP6A:   DEC    R7
01FE 51B1              ACALL  SD500              ;SEND THE WORD AND WAIT 500ms
0200 BF00FA            CJNE   R7,#0,LP6A
0203 511B              ACALL  ZEROSC             ;RETURN TO ZERO
0205 013D              AJMP   W                  ;WAIT FOR KEY PRESS
0207           PROG7:
               ;
               :   This routine alternates between 3/8 and 5/8 scale ten times with 300ms delay
               ;   between steps, then waits 500ms before returning display to zero scale.
               ;
0207 7A0A              MOV    R2,#10             ;SET COUNTER
0209 7E7F      PR7:    MOV    R6,#07FH
020B 7F01              MOV    R7,#1
020D 51AD              ACALL  SD300              ;SEND OUT THE WORD AND WAIT 300ms
020F 7F02              MOV    R7,#2
0211 51AD              ACALL  SD300              ;SEND OUT THE WORD AND WAIT 300ms
0213 DAF4              DJNZ   R2,PR7             ;DO IT 10 TIMES
0215 51A5              ACALL  DLY500             ;WAIT 500ms
0217 511B              ACALL  ZEROSC             ;RESET TO ZERO SCALE
0219 0130              AJMP   START              ;LOOK FOR VALID PROGRAM
               ;
               ;
```

```
              ;             SUBROUTINES
              ;
              ;
021B 7F00     ZEROSC: MOV    R7,#0             ;RESET METER TO ZERO SCALE
021D 7E00             MOV    R6,#0
021F 4125             AJMP   RST
0221 7F03     FULLSC: MOV    R7,#03H           ;SET METER TO FULL SCALE
0223 7EFF             MOV    R6,#0FFH
0225 51B5     RST:    ACALL  SENDIT
0227 22
0228          SO:
              ;
              ;   This subroutine sends increasing 10–bit words in registers R7 & R6 to the SGD.
              ;
0228 51B5             ACALL  SENDIT            ;WRITE THE 10–BIT WORD TO SGD
022A 5130             ACALL  UP                ;INCREASE THE WORD VALUE
022C 30E2F9           JNB    ACC.2,SO          ;JUMP IF BIT 2 NOT SET
022F 22               RET
0230          UP:
              ;
              ;   This subroutine waits for a period of time = 10ms X DELAY read un, then
              ;   increases the 10–bit word by the INCREMENT SELECT amount.
              ;
0230 E590             MOV    A,P1              ;READ DELEY
0232 F4               CPL    A                 ;COMPLEMENT ACC
0233 540F             ANL    A,#0FH            ;MASK OFF UPPER 4 BITS
0235 23               RL     A
0236 23               RL     A
0237 F9               MOV    R1,A
0238 B90002           CJNE   R1,#0,D10         ;JUMP IF DELAY SET FOR ZERO
023B 8006             SJMP   NODLY
023D 7B01     D10:    MOV    R3,#1             ;SET FOR 1 X 10ms DELAY
023F 5195     D10A:   ACALL  DLY10MS           ;DELAY 10MS x DELAY
0241 D9FC             DJNZ   R1,D10A
0243 E5B0     NODLY:  MOV    A,P3              ;READ INCREMENT SELECT
0245 F4               CPL    A                 ;COMPLEMENT ACC
0246 5403             ANL    A,#3              ;MASK OFF UPPER 6 BITS
0248 23               RL     A
0249 23               RL     A
024A 23               RL     A
024B 04               INC    A
024C 2E               ADD    A,R6              ;ADD INCREMENT TO R6
024D FE               MOV    R6,A              ;SAVE IT
024E E4               CLR    A
024F 3F               ADDC   A,R7              ;ADD CARRY TO R7
0250 FF               MOV    R7,A              ;SAVE IT
0251 22               RET
0252          SOD:
              ;
              ;   This subroutine sends out decreasing words at the rate set by DELAY and
              ;   step size determined by INCREMENT SELECT.
              ;
0252 51B5             ACALL  SENDIT            ;SEND OUT THE PRESENT WORD
0254 515A             ACALL  DOWN              ;DECREASE THE WORD
0256 50FA             JNC    SOD               ;DO IT AGAIN IF CARRY NOT SET
0258 411B             AJMP   ZEROSC
025A          DOWN:
              ;
              ;   Waits for 10ms x DELAY pot setting, then sends out decreasing values of words
              ;   in step sizes of 8 x INCREMENT SELECT + 1.
              ;
025A E590             MOV    A,P1              ;READ DELAY
025C F4               CPL    A                 ;COMPLEMENT ACC
025D 540F             ANL    A,#0FH            ;MASK OFF UPPER FOUR BITS
025F 23               RL     A
0260 23               RL     A
0261 F9               MOV    R1,A              ;SAVE DELAY
0262 B90002           CJNE   R1,#0,D10S        ;JUMP IF DELAY SET FOR ZERO
0265 8004             SJMP   NDD
```

```
0267 5195    D10S:   ACALL  DLY10MS            ;DELAY 10ms x (DELAY +1)
0269 D9FC            DJNZ   R1,D10S
026B E5B0    NDD:    MOV    A,P3               ;READ INCREMENT SELECT
026D F4              CPL    A                  ;COMPLEMENT ACC
026E 5403            ANL    A,#3               ;MASK OFF UPPER 6 BITS
0270 23              RL     A                  ;MULTIPLY BY 8
0271 23              RL     A
0272 23              RL     A
0273 04              INC    A                  ;INSURE MINIMUM STEP
0274 C3              CLR    C                  ;CLEAR CARRY FOR SUBTRACTION
0275 CE              XCH    A,R6
0276 9E              SUBB   A,R6               ;SUBTRACT INCREMENT FROM R6
0277 CE              XCH    A,R6               ;SAVE IT
0278 E4              CLR    A                  ;CLEAR ACCUM FOR SUBTRACTION
0279 CF              XCH    A,R7
027A 9F              SUBB   A,R7               ;SUBTRACT BORROW FROM R7
027B 5403            ANL    A,#3               ;INSURE MAXIMUM WORD
027D CF              XCH    A,R7               ;SAVE IT
027E 22              RET
027F 00      DELAY:  NOP                       ;3υs DELAY
0280 22              RET
0281         DMS10:
             ;
             ;   Produces a delay of 10ms x the value in R3.
             ;   Destroys R3 and timer readings.
             ;
             ;
0281 758AF0          MOV    TL,#LOW,(0–10000) ;LOAD TIMER FOR 10ms DELAY
0284 758CD8          MOV    TH,#HIGH(0–10000)
0287 C28D            CLR    TF                 ;CLEAR TIMER FLAG
0289 D28C            SETB   TR                 ;START TIMER
028B 308DFD  MS10W:  JNB    TF,MS10W           ;WAIT FOR TIMER FLAG TO BE SET
028E C28D            CLR    TF                 ;CLEAR TIMER FLAG
0290 DBF9            DJNZ   R3,MS10W           ;WAIT RS x 10ms
0292 C28C            CLR    TR                 ;STOP TIMER
0294 22              RET
             ;
0295 7B01    DLY10MS: MOV   R3,#1              ;SET R3 FOR 10ms WAIT
0297 80EB            SJMP   DMS10              ;WAIT 10ms
             ;
0299 7B0A    DLY100: MOV    R3,#10             ;SET R3 FOR 100ms WAIT
029B 80E4            SJMP   DMS10              ;WAIT 100ms
             ;
029D 7B14    DLY200: MOV    R3,#20             ;SET R3 FOR 200ms WAIT
029F 80E0            SJMP   DMS10              ;WAIT 200ms
             ;
02A1 7B1E    DLY300: MOV    R3,#30             ;SET R3 FOR 300ms WAIT
02A3 80DC            SJMP   DMS10              ;WAIT 300ms
             ;
02A5 7B32    DLY500: MOV    R3,#50             ;SET R3 FOR 500ms WAIT
02A7 80D8            SJMP   DMS10              ;WAIT 500ms
             ;
02A9 51B5    SD200:  ACALL  SENDIT             ;SEND THE WORD
02AB 80F0            SJMP   DLY200             ;WAIT 200ms
             ;
02AD 51B5    SD300:  ACALL  SENDIT             ;SEND THE WORD
02AF 80F0            SJMP   DLY300             ;WAIT 200ms
             ;
02B1 51B5    SD500:  ACALL  SENDIT             ;SEND THE WORD
02B3 80F0            SJMP   DLY500             ;WAIT 500ms
             ;
02B5         SENDIT:
             ;
             ;   This subroutine sends out a single word locate4d in R7 and R6.
             ;   Accumulator, R0 and R1 are destroyed.
             ;
02B5 D282            SETB   P0.2               ;SET CS HIGH
02B7 7902            MOV    R1,#02             ;SET COUNTER FOR 2 BITS OF R7
02B9 EF              MOV    A,R7               ;MOVE R7 TO A FOR SEND OUT
```

```
02BA 13            RRC    A                 ;ALIGN R7 FOR SEND OUT
02BB 13            RRC    A
02BC 13            RRC    A
02BD 51C7          ACALL  SEND1             ;SEND OUT UPPER TWO BITS
02BF 7908          MOV    R1,#8             ;SET COUNTER FOR R6 SEND OUT
02C1 EE            MOV    A,R6              ;MOVE R6 TO ACCUM
02C2 51C7          ACALL  SEND1             ;SEND OUT LOWER 8 BITS
02C4 C282          CLR    P0.2              ;LOAD SGD
02C6 22            RET
02C7       SEND1:
           ;
           ;   This subroutine sends [R1] number of bits of the accumulator, starting
           ;   with the MSB over the IIC port.
           ;   Accumulator, R0 and R1 are destroyed.
           ;
02C7 33            RLC    A                 ;ROTATE BIT TO CARRY
02C8 9281          MOV    P0.1,C            ;MOVE CARRY TO DATA OUT
02CA C280          CLR    P0.0              ;CLOCK LOW
02CC 00            NOP
02CD D280          SETB   P0.0              ;CLOCK HIGH
02CF D9F6          DJNZ   R1,SEND1          ;SEND NEXT BIT TILL DONE
02D1 22            RET
           ;
02D2 E5B0  RPS:    MOV    A,P3              ;READ PORT 3 FOR PROGRAM SELECT
02D4 F4            CPL    A                 ;COMPLEMENT ACC
02D5 03            RR     A                 ;ROTATE TO LSB’s & MULT BY 2
02D6 540E          ANL    A,#0EH            ;MASK FOR PROGRAM SELECT * 2
02D8 DD            RET
           END

ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

```
ACC ............ D ADDR   00E0H     PREDEFINED
CALC ........... C ADDR   00C5H
D10 ............ C ADDR   023DH
D10A ........... C ADDR   023FH
D10S ........... C ADDR   0267H
DCX ............ C ADDR   0073H
DELAY .......... C ADDR   027FH     NOT USED
DIV24 .......... C ADDR   010BH
DLY100 ......... C ADDR   0299H     NOT USED
DLY10MS ........ C ADDR   0295H
DLY200 ......... C ADDR   029DH
DLY300 ......... C ADDR   02A1H
DLY500 ......... C ADDR   02A5H
DMS10 .......... C ADDR   0281H
DOWN ........... C ADDR   025AH
DP1 ............ C ADDR   006DH
FULLSC ......... C ADDR   0221H
GFS ............ C ADDR   00C0H
GZS ............ C ADDR   00BDH
HT ............. C ADDR   00B7H
IE ............. D ADDR   00A8H     PREDEFINED
JMPTBL ......... C ADDR   004CH
LP5 ............ C ADDR   01E3H
LP6 ............ C ADDR   01F7H
LP6A ........... C ADDR   01FDH
MEAS ........... C ADDR   0082H
MS10W .......... C ADDR   028BH
N4 ............. C ADDR   01D1H
NDD ............ C ADDR   026BH
NODLY .......... C ADDR   0243H
NZS ............ C ADDR   00D0H
P0 ............. D ADDR   0080H     PREDEFINED
P1 ............. D ADDR   0090H     PREDEFINED
P3 ............. D ADDR   00B0H     PREDEFINED
P30 ............ C ADDR   0149H
P31 ............ C ADDR   015CH
P32 ............ C ADDR   0170H
P4 ............. C ADDR   0188H
P40 ............ C ADDR   0190H
P41 ............ C ADDR   01A1H
P42 ............ C ADDR   01B3H
P4A ............ C ADDR   018EH
P5 ............. C ADDR   01D5H
PR7 ............ C ADDR   0209H
PROG0 .......... C ADDR   005CH
PROG1 .......... C ADDR   0068H
PROG2 .......... C ADDR   007AH
PROG3 .......... C ADDR   0145H
PROG4 .......... C ADDR   0186H
PROG5 .......... C ADDR   01D3H
PROG6 .......... C ADDR   01F3H
PROG7 .......... C ADDR   0207H
READY .......... C ADDR   0040H
ROTL ........... C ADDR   00F0H
RPS ............ C ADDR   02D2H
RST ............ C ADDR   0225H
RTH ............ D ADDR   008DH     PREDEFINED
RTL ............ D ADDR   008BH     PREDEFINED
SD200 .......... C ADDR   02A9H     NOT USED
SD300 .......... C ADDR   02ADH
SD500 .......... C ADDR   02B1H
SEND1 .......... C ADDR   02C7H
SENDIT ......... C ADDR   02B5H
SO ............. C ADDR   0228H
SOD ............ C ADDR   0252H
START .......... C ADDR   0030H
TF ............. B ADDR   008DH     PREDEFINED
TH ............. D ADDR   008CH     PREDEFINED
TL ............. D ADDR   008AH     PREDEFINED
TR ............. B ADDR   008CH     PREDEFINED
UP ............. C ADDR   0230H
W .............. C ADDR   003DH
W20 ............ C ADDR   009AH
W21 ............ C ADDR   009FH
W22 ............ C ADDR   00ADH
W23 ............ C ADDR   00B2H
ZERO ........... C ADDR   0142H
ZEROSC ......... C ADDR   021BH
```