USE of the ISD4000 with a Low Cost Motorola Microcontroller

ISD Applications Note #1, **How to use the ISD4000 with a Microcontroller,** described code written for the COPS family of microcontrollers to do basic record and playback in the ISD4000 device family. The ISD-ES302 demo board uses that microcontroller and runs the listed software from Application Note #1.

This Applications Note describes how the Motorola 6805 series of microcontrollers may be used to perform the same function. Specifically, the attached code runs in a MC68HC705J1A 20 pin device and plugs into the ISD-ES302 demo board via an adapter board. Only one change was made on the ES-302 board itself, the addition of a diode. The adapter board and modification to the ES-302 will be described following the software discussion. Much of this design is covered in Applications Note #1, available in the ISD Data Book, 1996 2nd Edition and later data book. Refer to that Application Note for the main board schematic and related explanations. The flow chart for the software is essentially the same for both notes and will not be repeated here.

The one major difference between the code in Applications Note #1 and this one is that the COPS processor has a hardware SPI port and the 'J1A Motorola microcontroller does not. This software, therefore, has routines written to replace the hardware SPI port.

Major software routines:

The list file of this software will be broken up into pieces and each routine described. Many notes can also be found in the software listing itself. An unassembled source file of this software is available as an Email attachment from ISD Applications Department. Send your request to *apps@isd.com* and ask for the Applications Note Number 3 Source code.

Program Set Up and Listing Header: This software was assembled using a 6805 cross assembler purchased from 2500 A.D. Software.

```
2500 A.D. 6805 Macro Assembler  -  Version 4.01b
               ------------------------------------------------
                       Input  Filename : DEMO1.ASM

                              *
                              *
                              *
                              *
                              *
                                     LIST ON

```

Microcontroller Pin Out and I/O definition, Register definition: The following shows the pin out of the microcontroller and defines the device I/O. It also defines the memory map and interrupt vectors as well as the control registers used in the device.

```
0000                          .ABSOLUTE
                             ************************************************************
                             *Port Definitions       MC68HC705J1A DIP or SOIC Package
                             *PORT A
                             ************************************************************
       0000                  porta    equ     0
       0040                  portaDDR equ     %01000000

       0000                  L2       equ     $00     pin 18 - Go to Begin pushbutton
       0001                  L5       equ     $01     pin 17 - Skip to Next pushbutton
       0002                  L6       equ     $02     pin 16 - Play Next pushbutton
       0003                  L1       equ     $03     pin 15 - REC pushbutton
       0004                  L4       equ     $04     pin 14 - STOP pushbutton
       0005                  RAC      equ     $05     pin 13 - Row Address Clock Input
       0006                  LOWBAT   equ     $06     pin 12 - LED control output
       0007                  L7       equ     $07     pin 11 - Play Last pushbutton
                             *
                             ************************************************************
                             *PORT B
                             ************************************************************
       0001                  portb    equ     1
       001F                  portbDDR equ     %00011111

       0000                  PD       equ     $00     pin 8  - PD pin to ISD2500
       0001                  PLAREC   equ     $01     pin 7  - PLAY/REC pin to ISD2500
       0002                  SSBAR    equ     $02     pin 6  - Slave Select SPI Output
       0003                  SK       equ     $03     pin 5  - SPI Clock output
       0004                  SO       equ     $04     pin 4  - Serial Out
       0005                  SI       equ     $05     pin 3  - Serial In
                             *
```

```
                       ************************************************************
                       * Other Pin definitions
                       ************************************************************
                       *Pin 10 = VSS (GND)
                       *Pin 20 = RESET\ (active low)
                       *Pin 19 = IRQ\ ISD4000 Interupt output
                       *Pin  9 = VCC (+5)
                       *Pin  1 = OSC1
                       *Pin  2 = OSC2

                       *SOFTWARE ASSUMES A 3.579545 MHz Crystal
                       *PERIOD IS .279365 uSEC, OR INTERNAL CLOCK OF .55873 uSEC
PER
                       *CYCLE.                                        ------
                       *
                       *
                       ************************************************************
                       *area definitions
                       ************************************************************
       00C0            RamArea         equ     $00C0
       00FF            StkTop          equ     $00FF
       0300            RomArea         equ     $0300
       07F8            IntVects        equ     $07F8
                       *

                       ************************************************************
                       *Vector area
                       ************************************************************
07F8                       org     IntVects
07F8   0300                fdb     TIMESVC
07FA   0302                fdb     extsvc
07FC   0301                fdb     swisvc
07FE   0324                fdb     reset
                       *
                       *

                       ************************************************************
                       *Control Register Definitions
                       ************************************************************
       0008            TSCR            equ     $8       TIMER STATUS AND CONTROL REG
       0009            TCR             equ     $9       TIMER COUNTER REGISTER
       000A            ISCR            equ     $A       IRQ STATUS AND CONTROL
       0010            PDRA            equ     $10      PULL DOWN REGISTER A
       0011            PDRB            equ     $11      PULL DOWN REGISTER B
       07F1            MOR             equ     $07F1
       07F0            COPR            equ     $07F0


```

Flag Registers and System Equates: The following listing shows the definition of the RAM area in the microcontroller as well as locations of flags, registers and equates. Note that there are two bytes reserved for flags when only one was necessary. Since this program was well within the memory boundaries of the microcontroller, it was not necessary to eliminate this unneeded byte.

```
                       ************************************************************
                       *Ram flags and registers
                       ************************************************************


00C0                                 org     RamArea
                       ************************************************************
00C0                   R00             rmb     1
                       ***************************
       0000            PLAYING         equ     $00      Indicates we are Playing
back
       0001            RECDING         equ     $01      Indicates we are Recording
       0002            MAKEIT0         equ     $02      Make it zero flag
       0003            OPERATE         equ     $03
                       ******          equ     $04
                       ******          equ     $05
                       ******          equ     $06
                       ******          equ     $07
                       ***************************
00C1                   R01             rmb     1
                       ***************************
                       ******          equ     $10
                       ******          equ     $11
                       ******          equ     $12
                       ******          equ     $13
                       ******          equ     $14
                       ******          equ     $15
                       ******          equ     $16
                       ******          equ     $17
                       ***************************

                       ***************************
                       *NOW SET UP THE REGISTERS USED IN THE PROGRAM
                       **********************************************
00C2                  STRTREGS
00C2                  ADDRL           rmb      1    Address reg LOW
00C3                  ADDRH           rmb      1    Address reg HIGH
00C4                  RECADDL         rmb      1    holds addr of last record (LOW)
00C5                  RECADDH         rmb      1    holds addr of last record (HIGH)
00C6                  SPIONE          rmb      1    SPI Low byte - with control bits
00C7                  SPITWO          rmb      1    SPI High byte - address
00C8                  SEQCNT          rmb      1    Real Time Interrupt counter
00C9                  TOFCNT          rmb      1    Timer Overflow Flag counter
00CA                  TEMP            rmb      1    USED FOR VARIOUS STUFF
```

```
00CB                 TEMP1           rmb      1      USED IN DELAY TIMERS
00CC                 TEMP2           rmb      1      USED only in 16.879 mSEC TIMER
00CD                 TEMPOUT         rmb      1      Store of acc during SPI activity
00CE                 TEMPIN          rmb      1      Store of acc during SPI activity
                     *
00CF                 ENDREG
                     *
          *NOTE THAT THIS POINT CANNOT GO PAST $ WITHOUT LOOKING AT THE STACK USAGE
                       **********************************************************
                       *Define equates and other things
                       **********************************************************
       0004            DDR             equ     4
       0020            POWERUP         equ     $20
       00A0            SETREC          equ     $A0
       0000            STOPPWRDN       equ     $00
       0030            STOP            equ     $30     (IAB also set)
       00E0            SETPLAY         equ     $E0
       00B0            REC             equ     $B0
       00E8            SETMC           equ     $E8
       00F8            MC              equ     $F8
       00F0            PLAY            equ     $F0
       0030            RINT            equ     $30     (IAB also set)
       0070            READADR         equ     $70     (no RUN, IAB set)


                       *
```

Interrupt Routines: The interrupt routines are defined below. Note that the timer interrupt service routine (TIMESVC) and the software interrupt service routine (SWISVC) are not needed in this software. An accidental interrupt to those routines results in an immediate return from interrupt instruction (rti).

The external interrupt routine shuts off the ISD2500 device on the board that is used as a microphone pre-amp and speaker driver. It also sends a STOP opcode to the ISD4000 that is probably not needed because an interrupt from the ISD4000 always results from the end of an operation. The external interrupt routine then checks to see that the interrupt has been cleared, i.e. that the INT pin of the ISD4000 is back HIGH. If it is not, then an early chip revision of the ISD4000 is in the system and the device is in the Overflow interrupt state. To insure compatibility with this early revision device, the routine then sends a SETPLAY opcode with an address of zero and then a STOP opcode to clear the address counter in the ISD4000 and clear the Overflow interrupt.

```
                       **********************************************************
                       *Interrupt servce routines and subroutine
                       **********************************************************
00CF                        .CODE
00CF                        .RELATIVE
0300                        org     RomArea
                       *
                       *
0300                   TIMESVC
                       *
                       *NOTE:  Fop=.5587302 uSec (for a color burst xtal)
                       *
                       *

0300
0300   80                   rti
0301
0301                   swisvc
0301   80                   rti
                       *

                       *-----------------------------------------------------------
-
                       *               START OF EXTERNAL INTERRUPT SERVICE ROUTINE
                       *-----------------------------------------------------------
-
                       *

                       *The external int comes from the ISD4000. We first execute a
                       *STOP command to try to clear the interrupt.
0302                   extsvc
0302   10 01                bset    PD,portb        power down ISD2500 audio chip

0304   1D 00                bclr    LOWBAT,porta    turn off the LED
0306   17 C0                bclr    OPERATE,R00     clear the OPERATE flag

0308   A6 30                lda     #STOP
030A   CD 04 5E             jsr     SPI_8           stop the operation

030D   CD 04 C7             jsr     TPBN            delay

                       *We should have a cleared int by now.  If it's not cleared,
                       *we must be in Overflow.

0310   2F 11                bih     ENDINT          Branch if interrupt line is
HIGH

                       *We in Overflow. Start play at zero then immediately stop
it.
                       *This should clear the interrupt in a beta ISD4000 device

0312   3F C2                clr     ADDRL
0314   3F C3                clr     ADDRH
0316
0316   A6 E0                lda     #SETPLAY        set up to play message with
addr
```

```
243   0318   CD 04 2E             jsr     ADDADDR          add the address to it
244   031B
245   031B   CD 04 6B             jsr     SPI_16           start the playback
246
247                          *Now do the STOP
248   031E   A6 30                lda     #STOP
249   0320   CD 04 5E             jsr     SPI_8            stop the operation
250
251   0323                   ENDINT
252   0323   80                   rti
253   0324
254
255                      *
256                      *----------------------------------------------------------------
- 257                       *                END OF EXTERNAL INTERRUPT SERVICE ROUTINE
258                      *---------------------------------------------------------------
259                      *
260                      ;
261
```

Initialization: The reset initialization routine begins at address 324 (all address references are in hexadecimal). Note that the Mask Option Register (MOR) programming is shown for completeness but is “dummied out” so that this otherwise useless code is not executed. The Mask Option Register must be programmed by the EPROM programmer and must be set up manually in most programmers. Note also that the STOP command in line #321 is dummyed out; the sleep instruction of the microcontroller is not used in this program.

```
262     ;****************************************************************************
263     ;       INITIALIZE REGISTERS
264     ;       POWER UP AND PRESET ROUTINE
265     ;****************************************************************************
266
267   0324                   reset
268                          *
269   0324   A6 40           USER     lda     #portaDDR    set up to read push buttons
270   0326   B7 04                    sta     porta+DDR
271   0328   A6 1F                    lda     #portbDDR
272   032A   B7 05                    sta     portb+DDR
273
274                          *        lda     #$20
275                          *        sta     MOR     enable Port A IRQs, no COP, enable
276                          *                        pull down resistors, enable OSC res.
277
278
279   032C   A6 00                lda     #$00         Turn the LED off
280   032E   B7 00                sta     porta
281
282   0330   A6 07                lda     #$07          SI=LOW+SO=LOW+SK=LOW+SSBAR=HIGH+
283                          *                         PLAYREC=HIGH + PD=HIGH
284   0332   B7 01                sta     portb
285
286   0334   A6 FC                lda     #$FC
287   0336   B7 10                sta     PDRA          enable pull-down res PA0 & PA1
288   0338   A6 3F                lda     #$3F
289   033A   B7 11                sta     PDRB          enable no pull downs on port B
290   033C   3F C0                clr     R00
291   033E   3F C2                clr     ADDRL
292   0340   3F C3                clr     ADDRH
293   0342   3F C4                clr     RECADDL
294   0344   3F C5                clr     RECADDH
295   0346   3F CA                clr     TEMP
296
297   0348   5F                   clrx
298
299   0349   A6 0C                lda     #$0C             Clear and disable timer int.
300   034B   B7 08                sta     TSCR
301
302
303                   ;************ INITIALIZE ISD3300 ******************************
304   034D
305   034D   A6 20                lda     #POWRERUP
306   034F   CD 04 5E             jsr     SPI_8            power up the ISD4000
307
308   0352   A6 20                lda     #POWRERUP
309   0354   CD 04 2E             jsr     ADDADDR          add pwr up info to the addr
info
310   0357
311   0357   CD 04 6B             jsr     SPI_16           send in an address of all zeros
312
313   035A   CD 04 A1             jsr     FLASH1           blink the led once
314
315   035D   9C                   rsp                     reset the stack pointer
316   035E   9A                   cli                      enable interrups and GO!
317   035F
318
319   035F                   STOPPIT
320
321                           *        stop
322
323
```

Main Loop: The main loop simply looks for push-button closures on 6 switches. When a switch is closed, the routine branches to the proper point down in the code to execute the indicated operation. The microcontroller spends most of it’s time in this loop waiting for a switch closure.

```
 *************************************************************************
 *               TOP OF THE LOOP.  EVERYTHING STARTS FROM HERE           *
 *                                                                       *
 *************************************************************************
 ********************  WAIT FOR AN INPUT HERE ********************************
 *
 *              MAIN LOOP WAITING FOR A SWITCH CLOSURE
 *****************************************************************************
                     * Read the pushbuttons and branch if they are "alive".
035F                   READ
035F   00 00 11             brset   L2,porta,GOTOBEG
0362   02 00 1E             brset   L5,porta,SKIP2NXT
0365   04 00 3E             brset   L6,porta,PLAYNXT
0368   06 00 60             brset   L1,porta,RECIT
036B   08 00 0F             brset   L4,porta,STOPITX
036E   0E 00 0F             brset   L7,porta,PLAYLASX

0371   20 EC                bra     READ


```

GOTOBEG: The “Go To Beginning” routine only sets the flag bit MAKIT0 (for Make It Zero) located in the R00 flag register. The MAKIT0 bit will be used to indicate to other routines that record or playback should start from address zero of the ISD4000 when next executed. After setting the flag bit, this routine branches to the READX routine which runs a debounce timer. The READX routine is executed after any push-button sequence to effectively debounce the push buttons.

Also, in each of the next 3 routines, SKIP2NXT (Skip to the Next message), PLAYNXT (Play the Next message) and RECIT (Record a message) look at the MAKIT0 bit to see if the address counter should be first be cleared to all zeros before executing the operation. Thus the RECIT routine will be told to begin at zero if the GOTOBEG push button was pressed before the RECIT routine is called. If the push button was not pressed, RECIT will record starting at the end of the last record or play operation without resetting the address counter.

```
******************************* GO TO Beginning ****************************

                        *The Go To Beginning Button is pushed
0373                   GOTOBEG
0373   06 C0 53             brset   OPERATE,R00,READZ
0376   14 C0                bset    MAKEIT0,R00


0378   CD 04 AC             jsr     FLASH2          blink the led twice

037B   20 4C                bra     READZ
****************************************************************************

```

Misc: A couple of the branches are too long. An intermediate jump is needed.

```
037D                   STOPITX
037D   CC 03 F9             jmp     STOPIT

0380                   PLAYLASX
0380   CC 04 05             jmp     PLAYLAST

```

SKIP2NXT: This routine sends a SETMC[1] opcode to the ISD4000 with an address of zero if the MAKIT0 bit is set or sends a MC opcode if it is not set. This routine does not start a record or play operation, but merely causes the address counter in the ISD4000 to be modified so that it is left pointed at the “next” message in the device.

```
 ******************************* Message Cueing *****************************

       *Now do a message cueing cycle . . . but first, do we do it from zero?
0383                   SKIP2NXT
0383   06 C0 43             brset   OPERATE,R00,READZ

0386   05 C0 13             brclr   MAKEIT0,R00,NOCUE0  if this flag clear,
                           *                            no cue from zero

                       *We do a message cueing cycle from address zero
0389   15 C0                bclr    MAKEIT0,R00
038B
038B   3F C2                clr     ADDRL
038D   3F C3                clr     ADDRH
038F
038F   CD 04 A1             jsr     FLASH1     blink the LED once
```

[1] See the Opcode Summary in the ISD4000 Data Sheet for an explanation of the Instruction Opcodes

```

0392   A6 E8            lda     #SETMC      set up to do a msg cue + addr
0394   CD 04 2E         jsr     ADDADDR     add the address to it
0397

0397   CD 04 6B         jsr     SPI_16      do msg cue (but don't play)


039A   20 2D            bra     READZ       debounce and leave

                   *we do a message cueing cycle from the last address
039C               NOCUE0
039C   CD 04 A1         jsr     FLASH1      blink the LED once

039F   A6 F8            lda     #MC
03A1   CD 04 5E         jsr     SPI_8       do msg cue (but don't play)

03A4   20 23            bra     READZ       debounce and leave

  ****************************************************************************
03A6


```

PLAYNXT: This routine sends a SETPLAY opcode to the ISD4000 with an address of zero if the MAKIT0 bit is set. This causes playback to start at address zero. If the MAKIT0 flag bit is not set, a playback operation begins at whatever address is currently in the ISD4000's internal address pointer. While playback is occurring, only the STOP push button will be recognized by the software.

```
  ***************************** Play Next Message ****************************

             *Play the next message . . . unless Go to Beginning has been pressed
03A6                  PLAYNXT
03A6   06 C0 79            brset   OPERATE,R00,READX  check if we already oper.
03A9   16 C0               bset    OPERATE,R00

03AB   12 01               bset    PLAREC,portb         Put the ISD2500 into Play
03AD   11 01               bclr    PD,portb             Power up the ISD2500

03AF   05 C0 10            brclr   MAKEIT0,R00,NOPLA0  if flag clear, no play fm
zero

                       *We play a message from address zero

03B2   15 C0               bclr    MAKEIT0,R00      clear the makit it zero flag
03B4   3F C2               clr     ADDRL
03B6   3F C3               clr     ADDRH
03B8
03B8   A6 E0               lda     #SETPLAY        set up to play a msg with addr
03BA   CD 04 2E            jsr     ADDADDR         add the address to it
03BD
03BD   CD 04 6B            jsr     SPI_16          start the playback

03C0   1C 00                 bset    LOWBAT,porta    turn on the LED

*Now fall through from starting at zero and send 8 more bits to continue play

                        *we really play the "next message"
03C2                   NOPLA0
03C2   A6 F0                 lda     #PLAY           load for play with IAB bit set
03C4   CD 04 5E              jsr     SPI_8          start the playback

03C7   1C 00                 bset    LOWBAT,porta    turn on the LED

03C9   20 57           READZ    bra     READX           debounce and leave

  ****************************************************************************
03CB


```

RECIT: This routine sends a SETREC opcode to the ISD4000 with an address of zero if the MAKIT0 bit is set. This causes record to start at address zero. If the MAKIT0 flag bit is not set, a record operation begins at whatever address is currently in the ISD4000's internal address pointer. While record is occurring, only the STOP push button will be recognized by the software.

An additional function is performed during the RECIT routine and before recording begins. If the MAKIT0 bit is not set, i.e. the internal address is to be used for record, then this address is read out of the ISD4000 using the SPIIN subroutine (which will be explained later). This address is stored in a set of registers[2] so that the PLAYLAST routine can make use of it. If the MAKIT0 bit is set, then a zero is stored in these registers.

```
  **************************** Record Next Message ***************************

             *Record the next message . . . unless Go to Beginning has been pressed
03CB                   RECIT
```

[2] Since the ISD4000 family uses an address longer than 8 bits, it takes two 8 bit registers to store the address. In the software, these registers are called RECADDL and RECADDH.

```
03CB   06 C0 54                 brset   OPERATE,R00,READX  check to see if already oper
03CE   16 C0                    bset    OPERATE,R00

03D0   13 01                    bclr    PLAREC,portb    Put the ISD2500 into Record
03D2   11 01                    bclr    PD,portb        Power up the ISD2500

03D4   05 C0 16                 brclr   MAKEIT0,R00,NOREC0  if flag clear, no rec from
0


                           *We record message from address zero
03D7   15 C0                    bclr    MAKEIT0,R00      clear the makit it zero flag
03D9   3F C2                    clr     ADDRL
03DB   3F C3                    clr     ADDRH

                           *Also clear the "record next" address bytes


03DD   3F C4                    clr     RECADDL
03DF   3F C5                    clr     RECADDH
03E1
03E1   A6 A0                    lda     #SETREC     set up to record a message with
addr
03E3   CD 04 2E                 jsr     ADDADDR     add the address to it
03E6
03E6   CD 04 6B                 jsr     SPI_16      start the recording

03E9   1C 00                    bset    LOWBAT,porta    turn on the LED
03EB
03EB   20 04                    bra     NOREC1

                           *Now send 8 more bits to continue record

                           *we really Record the "next message"
03ED                       NOREC0
03ED   A6 70                    lda     #READADR
03EF   AD 4C                    bsr     SPIIN        go read the current addr before
rec


03F1                       NOREC1
03F1   A6 B0                    lda     #REC
03F3   AD 69                    bsr     SPI_8           start the recording

03F5   1C 00                    bset    LOWBAT,porta    turn on the LED

03F7   20 29                    bra     READX           debounce and leave

  ****************************************************************************
03F9


```

STOPIT: This routine sends a STOP opcode to the ISD4000. This interrupts any operation in progress. If a record operation is interrupted, an EOM flag bit is stored in the device to mark the end of the message.

```
  ************************* Stop Recording or Playing ***********************

           *Stop whatever we are doing . . . but don't corrupt the address register
03F9                       STOPIT
03F9   A6 30                    lda     #STOP
03FB   AD 61                    bsr     SPI_8           stop the operation

03FD   1D 00                    bclr    LOWBAT,porta    turn off the LED

03FF   17 C0                    bclr    OPERATE,R00     turn off the "Operate" flag

0401   10 01                    bset    PD,portb        Power down the ISD2500

0403   20 1D                    bra     READX

  ****************************************************************************
0405

```

PLAYLAST: This routine retrieves the bytes that hold the address stored at the beginning of the last RECIT operation. It then sends a SETPLAY opcode and the address to the ISD4000 so that playback now begins at the address where the "last" recording started from.

```

  ****************************** Play Next Message ***************************

                           *Play the last message we recorded
0405                       PLAYLAST
0405   06 C0 1A                 brset   OPERATE,R00,READX  check if we already oper
0408   16 C0                    bset    OPERATE,R00

040A   12 01                    bset    PLAREC,portb    Put the ISD2500 into Playback
040C   11 01                    bclr    PD,portb        Power up the ISD2500


                     *Go get the address we stored off just before we recorded (It's in
                           *RECADDH + RECADDL).  Put it in the address registers and
GO!

040E   B6 C4                    lda     RECADDL
```

```
0410    B7 C2                  sta     ADDRL

0412    B6 C5                  lda     RECADDH
0414    B7 C3                  sta     ADDRH

0416    A6 E0                  lda     #SETPLAY      set up to play a message with
addr
0418    CD 04 2E               jsr     ADDADDR         add the address to it
041B
041B    CD 04 6B               jsr     SPI_16          start the playback

041E    1C 00                  bset    LOWBAT,porta    turn on the LED

0420    20 A0                  bra     NOPLA0          Send 2nd command to set IAB bit

 ****************************************************************************
0422


 *after doing any and all the stuff,you arrive here to debounce the switches
0422    B6 00          READX   lda     porta
0424    A4 9F                  and     #$9F          only look at the push buttons
0426    26 FA                  bne     READX         loop until the button is released

                         *OK, the button has been released
0428    CD 04 C7               jsr     TPBN            go wait for awhile
042B    CC 03 5F               jmp     READ
```

ADDADDR Subroutine: Whenever we do any of the "SET" operations, we have to add a 10 bit address to the 5 bit ISD4000 opcode. The address starts with A0 in the LSB of the 16 bit word, with the opcode in the left-most 5 bits of the word. Graphically this looks like:

MOSI	->	C4	C3	C2	C1	C0	x	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

The two bytes to be shifted into the SPI port must be loaded such that the first bit shifted in is A0, the second is A1, etc. with the very last bit shifted into the SPI port being the C4 RUN bit. At that point the Slave Select pin will go HIGH to end the data input and start the operation as indicated by the bytes just shifted in.

In the ADDADDR subroutine, the accumulator brings in the opcode and the address is defined by the data in the ADDRL and ADDRH bytes. ADDRL contains A7 thru A0 and ADDRH holds A9 and A8 in the two LSB locations. This subroutine uses an AND and an OR operation to combine the opcode with bits A9 and A8 of the address and load SPIONE with this data. ADDRL is next written to SPITWO. SPIONE and SPITWO are the two byes that will be sent out of the SPI port of the micro.

```
         ************************************************************
         ************************************************************
         *    SUBROUTINES START HERE!
         ************************************************************
         ************************************************************


         ************************************************************
         *SUBROUTINE ADDADDR
         ************************************************************
                            *This Subroutine adds the address to the control bits already
                            *present in SPIONE.  In doing so, it adds a second byte for
                            *16 bit transfers into the ISD4000.  This subroutine is
                            *called with a control byte in the accumulator.

042E                         ADDADDR
042E    B7 C6                  sta     SPIONE   put the accumulator into SPI one

0430    B6 C3                  lda     ADDRH    get the upper byte of the address
0432    A4 03                  and     #$03
0434    BA C6                  ora     SPIONE
0436    B7 C6                  sta     SPIONE
0438
0438    B6 C2                  lda     ADDRL    Go get the lower byte
043A    B7 C7                  sta     SPITWO

043C    81                     rts
```

SPIIN Subroutine: This subroutine is used to read the address data out of the ISD4000 MISO pin before starting a record operation. This saves the "current address" so that a "play last record" operation is possible. Graphically the output side of the ISD4000 SPI port (MISO) looks like:

MISO	<--	OVF	EOM	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	x	0	0	0

The OVF bit is presented to the MISO pin as soon as the Slave Select pin goes LOW. The first clock input to the ISD4000 SPI shifts the EOM to the MISO pin, and subsequent clocks shift the address data out, LSB first. Since we only read the MISO pin after the clock cycle, the OVF bit gets thrown away automatically. Note that the SHIFTIT routine (explained later) shifts the data into TEMPIN byte left to right. This causes the sense of the data to be inverted so that the EOM bit will be in the LSB position of the byte.

After the data is shifted out of the MISO pin, we are left with a byte holding P6 thru P0 plus the EOM bit and a second byte holding P9, P8 and P7. One more linked shift to the right of these two bytes dumps the EOM bit and positions the 10 bit address correctly for later use in the PLAYLAST routine.

It should be noted that whenever you read the ISD4000's SPI port data, you also are shifting data into the device. Accordingly, the calling routine must make sure the data written does not inadvertently start an unwanted operation, or interrupt an operation in progress. When SPIIN is called, the accumulator brings in the 5 bit opcode into the subroutine that is to be written to the MOSI port of the SPI. This routine assumes that an address will not be written into the SPI port at this time. Consequently the first byte written will be all zeros, followed by the second byte containing the opcode.

```
                       *****************************************************************
                       *SUBROUTNE SPIIN
                       *****************************************************************
                       * Read the SPI data from the chip, throw out the OVF and EOM
                       * bits, and store the address just retreived in RECADDL and
                       * RECADDH.  The accumulator comes in with the proper 5 bits
                       * of control so that nothing is disturbed.


043D                   SPIIN
043D   17 01                bclr    SK,portb        Make sure the clock starts LOW
043F   15 01                bclr    SSBAR,portb     Drop Slave Select

0441   B7 C6                sta     SPIONE  Save the accumulator for later

                       *we shift zeros in first

0443   3F CA                clr     TEMP    put all zeros in temp
0445   AD 39                bsr     SHIFTIT

0447   B6 CE                lda     TEMPIN  get the data just shifted in
0449   B7 C4                sta     RECADDL

                       * note that EOM bit is still in RECADDL.  We will
                       * fix this later.

044B   B6 C6                lda     SPIONE  go get the original accumulator
044D   B7 CA                sta     TEMP

044F   AD 2F                bsr     SHIFTIT go do the second byte


                       * TEMPIN now has upper 3 bits of the address in it.  We wil
                       * shift 1 of those bits into RECADDL.  The last 2 bits
remain
                       * in TEMPIN and this becomes RECADDH.

0451   34 CE                lsr     TEMPIN
0453   36 C4                ror     RECADDL

0455   B6 CE                lda     TEMPIN
0457   B7 C5                sta     RECADDH

0459   14 01                bset    SSBAR,portb     Slave Select goes HIGH to end
                       *                            cycle
045B   19 01                bclr    SO,portb        Leave this LOW when finished
045D   81                   rts
```

SPI_8 Subroutine: This subroutine is used to shift 8 bits into the MOSI pin of the ISD4000. The accumulator brings in the byte to be shifted. The SHIFTIT subroutine does the actual shifting of the data.

```
                       *****************************************************************
                       *SUBROUTNE SPI_8
                       *****************************************************************
                       *SPI driver subroutine - outputs 8 bits.  This routine sends
                       *the SPIONE byte out the bit banged SPI port.

045E                   SPI_8
045E   17 01                bclr    SK,portb        Make sure the clock starts LOW
0460   15 01                bclr    SSBAR,portb     Drop Slave Select

0462   B7 CA                sta     TEMP
0464   AD 1A                bsr     SHIFTIT
```

```
0466   14 01                bset    SSBAR,portb     Slave Select goes HIGH to end
                       *                                cycle
0468   19 01                bclr    SO,portb        Leave this LOW when finished
046A   81                   rts

```

SPI_16 Subroutine: This subroutine is used to shift 16 bits into the MOSI pin of the ISD4000. The SPITWO byte is shifted in first and the SPIONE bit is shifted in second. The SHIFTIT subroutine does the actual shifting of the data.

```
                       *************************************************************
                       *SUBROUTNE SPI_16
                       *************************************************************
                       *SPI driver subroutine - outputs 16 bits.  This routine
sends
                       *the SPIONE and SPITWO bytes out the bit banged SPI port.

046B                   SPI_16
046B   17 01                bclr    SK,portb        Make sure the clock starts LOW
046D   15 01                bclr    SSBAR,portb     Drop Slave Select

046F   B6 C7                lda     SPITWO
0471   B7 CA                sta     TEMP

0473   AD 0B                bsr     SHIFTIT

0475   B6 C6                lda     SPIONE
0477   B7 CA                sta     TEMP

0479   AD 05                bsr     SHIFTIT
047B
047B   14 01                bset    SSBAR,portb     Slave Select goes HIGH to end
                       *                                cycle
047D   19 01                bclr    SO,portb        Leave this LOW when finished
047F   81                   rts


```

SHIFTIT Subroutine: This subroutine does all the shifting of data into and out of the SPI port of the ISD4000. The TEMP register brings in the 8 bit data to be shifted out and the TEMPIN register brings data out of the routine. SHIFTIT talks directly to the ports of the microcontroller.

```
                       *************************************************************
                       *SUBROUTNE SHIFTIT
                       *************************************************************
                       *This subroutine shifts out 8 bits from the SPI Port

0480                   SHIFTIT
0480   A6 08                    lda     #8
0482   B7 CB                    sta     TEMP1           put an 8 count into temp 1

0484   B6 CA                    lda     TEMP

0486   44              OUTAGN   lsra                    shift the LSB into carry bit
0487   24 04                    bcc     OUTZERO

0489   18 01           OUTONE   bset    SO,portb        output a "1"
048B   20 02                    bra     CONTSPI

048D   19 01           OUTZERO bclr    SO,portb        output a "0"


                       *Now toggle the clock to shift the data
048F   16 01           CONTSPI  bset    SK,portb
0491   17 01                    bclr    SK,portb

                       *now look at incoming data

0493   0A 01 03                 brset   SI,portb,INONE
0496   98                       clc                     it's a 0, clear carry bit
0497   20 01                    bra     CONTIN

0499   99              INONE    sec                     it's a 1, set the carry bit

049A   36 CE           CONTIN   ror     TEMPIN          bring the bit into the acc

049C   3A CB                    dec     TEMP1
049E   26 E6                    bne     OUTAGN
04A0
04A0   81                       rts
```

FLASH1 Subroutine: The FLASH1 subroutine blinks the LED once.

```
                       *************************************************************
                       *SUBROUTNE FLASH1
                       *************************************************************
                       *This subroutine flashes the LED once
```

```

04A1                   FLASH1
04A1   1C 00                bset    LOWBAT,porta    turn on the LED
04A3
04A3   A6 96                lda     #150
04A5   B7 CB                sta     TEMP1           use 150 at the delay

04A7   AD 22                bsr     KEYDLY          delay
04A9   1D 00                bclr    LOWBAT,porta    turn off the LED
04AB   81                   rts
```

FLASH2 Subroutine: The FLASH2 subroutine blinks the LED twice.

```
                       ************************************************************
                       *SUBROUTNE FLASH2
                       ************************************************************
                       *This subroutine flashes the LED twice
04AC                   FLASH2
04AC   1C 00                bset    LOWBAT,porta    turn on the LED

04AE   A6 96                lda     #150
04B0   B7 CB                sta     TEMP1           use 150 at the delay

04B2   AD 17                bsr     KEYDLY          delay
04B4   1D 00                bclr    LOWBAT,porta    turn off the LED

04B6   A6 96                lda     #150
04B8   B7 CB                sta     TEMP1           use 150 at the delay

04BA   AD 0F                bsr     KEYDLY          delay
04BC   1C 00                bset    LOWBAT,porta    turn on the LED

04BE   A6 96                lda     #150
04C0   B7 CB                sta     TEMP1           use 150 at the delay

04C2   AD 07                bsr     KEYDLY          delay
04C4   1D 00                bclr    LOWBAT,porta    turn off the LED
04C6   81                   rts
```

TPBN Subroutine: This subroutine is used to generate delays in the program. Calling TPBN directly gives approximately a 1/4 second delay. KEYDLY can also be called as a subroutine. In this case, the accumulator brings in a variable that initiates a delay shorter than TPBN.

```
                       ************************************************************
                       *SUBROUTNE TPBN
                       ************************************************************
                       *USED FOR TIMING VARIOUS STUFF IN THE PROGRAM
                       *

                       *TPBN IS THE PUSH BUTTON DELAY IN PUSH BUTTON MODE
04C7   A6 FA           TPBN    lda     #250         DELAY 250 MILLISECONDS (approx)
04C9   B7 CB                    sta     TEMP1

04CB   B6 CB           KEYDLY  lda     TEMP1           (3)
04CD   27 0A                    beq     FINDLY          (3)
04CF   3A CB                    dec     TEMP1           (5)



                        ***********************************************************
                        *WAIT1     GENERATES A DELAY (1.154 mSEC)
                        ***********************************************************
         *WAIT1 GENERATES A 1.154 MILLISECOND DELAY.  IT DOES NOT DISTURB THE
         *ACCUMULATOR.  FROM A BSR (CALLING THIS ROUTINE) THROUGH THE RTS (ENDING
         *THIS ROUTINE, IT TAKES 2065 CYCLES x 4.3656 uSEC = 1.154 MSec.  THIS
THING
                      *LOOPS 256 TIMES.

04D1   3F CC           WAIT1   clr     TEMP2            (5)
04D3   3C CC           WAIT2   inc     TEMP2            (5)
04D5   26 FC                   bne     WAIT2            (3)

04D7   20 F2                   bra     KEYDLY           (3)

04D9   81              FINDLY  rts



                       *






                       * NOTE: THE LAST USABLE ADDRESS IS $7CF
                       ;
```

```
814
815
```

Push Button Operation Notes:

```
816       *Instructions:
817       *Six of the 9 push-buttons are labeled on the PCB.  Pushing the Yellow
818       *"Go to Beg" button will reset the address pointer to the front of the
819       *chip or address 000.  The indication that this has been done will be a
820       *double flash of the LED, D11, in the bottom left corner of the PCB.
821       *
822       *Pushing the Red "Record" button will turn on the Red LED to indicate
823       *that the chip is now recording anything it hears at the microphone M1
824       *near the top center of the board.  The board will continue to Record
825       *until the end of the chip is reached or the Black "Stop" button is pressed.
826       *At that time the LED will go out and the board will stop Recording.
827       *
828       *Pushing the White "Play_Last" button will playback what you have just
829       *recorded. This message will play through to its end or stop when you
830       *press the Black "Stop" button
831       *
832       *Pushing the Yellow "Go_to_Beg" button, and then the White "Play_Next"
833       *button will play messages from the beginning of memory through to its
834       *end or stop/pause when you press the Black "Stop" button. To resume
835       *playback push play-next again if playback is paused. To play the next
836       *message press play next again.
837       *
838       *The Blue "Skip_to_Next" button will let you bypass a message and play
839       *the one after that message.  For example, had you recorded three messages,
840       *beginning at the front of the chip, and gone back to the beginning after
841       *the last message then you could press the White "Play_Next" button to play
842       *message #1, press the Blue "Skip_to_Next" button to bypass message #2 and
843       *then Press the White "Play_Next" to play message #3.
844
845
846
846
```

Cross Reference Listing:

```
Defined            Symbol Name            Value              References

575    ADDADDR                           042E     243 309 381 421 468 536
158    ADDRH                             00C3     240 292 376 418 459 533 578
157    ADDRL                             00C2     239 291 375 417 458 530 583
Pre    CODE                              00C0     195
716    CONTIN                            049A     712
705    CONTSPI                           048F     699
120    COPR                          =   07F0
Pre    DATA                              0000
177    DDR                           =   0004     270 272
251    ENDINT                            0323     234
171    ENDREG                            00CF
798    FINDLY                            04D9     779
728    FLASH1                            04A1     313 378 391
744    FLASH2                            04AC     352
347    GOTOBEG                           0373     334
714    INONE                             0499     710
116    ISCR                          =   000A
 97    IntVects                      =   07F8     103
778    KEYDLY                            04CB     734 750 756 762 796
 57    L1                            =   0003     337
 54    L2                            =   0000     334
 58    L4                            =   0004     338
 55    L5                            =   0001     335
 56    L6                            =   0002     336
 61    L7                            =   0007     339
 60    LOWBAT                        =   0006     223 425 434 472 488 503 540 729 735
745
                                                  751 757 763
134    MAKEIT0                       =   0002     349 369 373 412 416 453 457
185    MC                            =   00F8     393
119    MOR                           =   07F1
804    MSGTABL                           04DA
390    NOCUE0                            039C     369
430    NOPLA0                            03C2     412 542
479    NOREC0                            03ED     453
484    NOREC1                            03F1     474
135    OPERATE                       =   0003     224 348 367 406 407 447 448 505 519
520
695    OUTAGN                            0486     719
698    OUTONE                            0489
701    OUTZERO                           048D     696
Pre    PAGE0                             0000
 69    PD                            =   0000     221 410 451 507 523
117    PDRA                          =   0010     287
118    PDRB                          =   0011     289
 70    PLAREC                        =   0001     409 450 522
186    PLAY                          =   00F0     431
132    PLAYING                       =   0000
518    PLAYLAST                          0405     361
360    PLAYLASX                          0380     339
405    PLAYNXT                           03A6     336
178    POWRERUP                      =   0020     305 308
806    PTTABL                            04DA
130    R00                               00C0     224 290 348 349 367 369 373 406 407
412
                                                  416 447 448 453 457 505 519 520
```

```
141    R01                                00C1
 59    RAC                             =  0005
333    READ                               035F     341 555
188    READADR                         =  0070     480
549    READX                              0422     406 436 447 490 509 519 551
436    READZ                              03C9     348 354 367 387 396
183    REC                             =  00B0     485
160    RECADDH                            00C5     294 465 532 630
159    RECADDL                            00C4     293 464 529 611 627
133    RECDING                         =  0001
445    RECIT                              03CB     337
187    RINT                            =  0030
 94    RamArea                         =  00C0     128
 96    RomArea                         =  0300     197
163    SEQCNT                             00C8
184    SETMC                           =  00E8     380
182    SETPLAY                         =  00E0     242 420 535
179    SETREC                          =  00A0     467
689    SHIFTIT                            0480     608 619 649 670 675
 74    SI                              =  0005     710
 72    SK                              =  0003     600 645 664 705 706
366    SKIP2NXT                           0383     335
 73    SO                              =  0004     634 653 679 698 701
599    SPIIN                              043D     481
161    SPIONE                             00C6     576 580 581 603 616 672
162    SPITWO                             00C7     584 667
663    SPI_16                             046B     245 311 384 423 470 538
644    SPI_8                              045E     227 249 306 394 432 486 501
 71    SSBAR                           =  0002     601 632 646 651 665 677
181    STOP                            =  0030     226 248 500
499    STOPIT                             03F9     358
357    STOPITX                            037D     338
319    STOPPIT                            035F
180    STOPPWRDN                       =  0000
156    STRTREGS                           00C2
 95    StkTop                          =  00FF
115    TCR                             =  0009
165    TEMP                               00CA     295 607 617 648 668 673 693
166    TEMP1                              00CB     691 718 732 748 754 760 776 778 780
167    TEMP2                              00CC     792 793
169    TEMPIN                             00CE     610 626 629 716
168    TEMPOUT                            00CD
200    TIMESVC                            0300     104
164    TOFCNT                             00C9
775    TPBN                               04C7     229 554
114    TSCR                            =  0008     300
269    USER                               0324
792    WAIT1                              04D1
793    WAIT2                              04D3     794
220    extsvc                             0302     105
 51    porta                           =  0000     223 270 280 334 335 336 337 338 339
425
                                                   434 472 488 503 540 549 729 735 745
751
                                                   757 763
 52    portaDDR                        =  0040     269
 66    portb                           =  0001     221 272 284 409 410 450 451 507 522
523
                                                   600 601 632 634 645 646 651 653 664
665
                                                   677 679 698 701 705 706 710
 67    portbDDR                        =  001F     271
267    reset                              0324     107
209    swisvc                             0301     106


         Lines Assembled :  846              Assembly Errors :  0
```