



# Using the ISD33000 Device with Motorola Microcontrollers

A repeat of Applications Note 1, using a different microcontroller. This microcontroller does not have a built in SPI port but can be programmed through firmware to provide one.

ISD application note "Using the ISD33000 with a Microcontroller," described code written for the COPS family of microcontrollers to do basic record and playback in the ISD33000 device series. The ISD-ES302 demo board uses that microcontroller and runs the listed software in that application note.

This application 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 ISD-ES302 board; the addition of a diode. The adapter board and ISD-ES302 modification are described following the software discussion. Much of this design is covered in the section "Using the

ISD33000 with a Microcontroller," refer to it 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.

The main difference between the code in the previous application note and this one is that the COPS processor has a hardware SPI port and the MC68HC705J1A 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 E-mail attachment from ISD Applications Department. Send a request to apps@isd.com and ask for the Applications Note No. 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
                Output Filename : DEMO1.obj

1          LIST ON
2          PL 120
3*****
4*This Software is Copyright 1996 by Information Storage Devices
5*
6  * ISD
7*
```

```

8*
9*
10*
11* *****          ***** *      *          *****
12* *      *      *          **    **      *      *
13*      *      *      *          * *    * *      *      *
14*      *      *      *****    * *    * *      *      *
15*      *      *      *          * *    * *      *      *
16*      *      *      *          * **    * *      *      *
17* *****          ***** *      *      *      *****
18*
19*
20*
21*
22*
23*
24;*****
25;*   ISD33000  D E M O          P R O G R A M          *
26;*
27;*
28;*   (C) 1996 I N F O R M A T I O N   S T O R A G E   D E V I C E S   *
29;*
30;*
31;*   MICROCONTROLLER          : MC68HC705J1A          *
32;*   DATE OF LAST REVISION: 06.03.96          *
33;*   NAME OF SOURCE FILE   : demol.asm          *
34;*   CLOCK SPEED          : 3.579545 MHz          *
35;*
36;*****
37;*****
38
39
40
41
42
43
44
45
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.

```

46  0000          .ABSOLUTE
47
48          *Port Definitions          MC68HC705J1A DIP or SOIC Package
49          *PORT A
50
51          0000          porta          equ          0
52          0040          portaDDR      equ          %01000000
53
54          0000          L2            equ          $00          pin 18 - Go to Begin pushbutton
55          0001          L5            equ          $01          pin 17 - Skip to Next pushbutton
56          0002          L6            equ          $02          pin 16 - Play Next pushbutton

```

```

57          0003          L1      equ      $03      pin 15 - REC pushbutton
58          0004          L4      equ      $04      pin 14 - STOP pushbutton
59          0005          RAC      equ      $05      pin 13 - Row Address Clock Input
60          0006          LOWBAT  equ      $06      pin 12 - LED control output
61          0007          L7      equ      $07      pin 11 - Play Last pushbutton
62          *
63          *****
64          *PORT B
65          *****
66          0001          portb   equ      1
67          001F          portbDDR equ      %00011111
68
69          0000          PD       equ      $00      pin 8 - PD pin to ISD2500
70          0001          PLAREC  equ      $01      pin 7 - PLAY/REC pin to ISD2500
71          0002          SSBAR   equ      $02      pin 6 - Slave Select SPI Output
72          0003          SK       equ      $03      pin 5 - SPI Clock output
73          0004          SO       equ      $04      pin 4 - Serial Out
74          0005          SI       equ      $05      pin 3 - Serial In
75          *
76          *****
77          * Other Pin definitions
78          *****
79          *Pin 10 = VSS (GND)
80          *Pin 20 = RESET\ (active low)
81          *Pin 19 = IRQ\ ISD33000 Interupt output
82          *Pin 9 = VCC (+5)
83          *Pin 1 = OSC1
84          *Pin 2 = OSC2
85
86          *SOFTWARE ASSUMES A 3.579545 MHz Crystal
87          *PERIOD IS .279365 uSEC, OR INTERNAL CLOCK OF .55873 uSEC PER
88          *CYCLE.          -----
89          *
90          *
91          *****
92          *area definitions
93          *****
94          00C0          RamArea   equ      $00C0
95          00FF          StkTop    equ      $00FF
96          0300          RomArea   equ      $0300
97          07F8          IntVects  equ      $07F8
98          *
99
100         *****
101         *Vector area
102         *****
103         07F8          org      IntVects
104         07F8 0300          fdb      TIMESVC
105         07FA 0302          fdb      extsvc
106         07FC 0301          fdb      swisvc
107         07FE 0324          fdb      reset
108         *
109         *
110
111         *****

```

```

112                 *Control Register Definitions
113                 *****
114 0008    TSCR    equ    $8      TIMER STATUS AND CONTROL REG
115         0009    TCR      equ    $9      TIMER COUNTER REGISTER
116         000A    ISCR     equ    $A      IRQ STATUS AND CONTROL
117         0010    PDRA     equ    $10     PULL DOWN REGISTER A
118         0011    PDRB     equ    $11     PULL DOWN REGISTER B
119         07F1    MOR      equ    $07F1
120         07F0    COPR     equ    $07F0
121
122

```

**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.

```

123                 *****
124                 *Ram flags and registers
125                 *****
126
127
128 00C0                org    RamArea
129                 *****
130 00C0                R00      rmb    1
131                 *****
132 0000    PLAYING equ    $00    Indicates we are Playing back
133 0001    RECDING equ    $01    Indicates we are Recording
134 0002    MAKEIT0 equ    $02    Make it zero flag
135 0003    OPERATE equ    $03
136         ***** equ$04
137         ***** equ$05
138         ***** equ$06
139         ***** equ$07
140         *****
141 00C1                R01      rmb    1
142                 *****
143         *****      equ    $10
144         *****      equ    $11
145         *****      equ    $12
146         *****      equ    $13
147         *****      equ    $14
148         *****      equ    $15
149         *****      equ    $16
150         *****      equ    $17
151                 *****
152
153                 *****
154                 *NOW SET UP THE REGISTERS USED IN THE PROGRAM
155                 *****
156 00C2STRREGS
157 00C2ADDRl rmb 1    Address reg LOW
158 00C3ADDRH rmb 1    Address reg HIGH
159 00C4RECADDL rmb1  holds addr of last record (LOW)

```

```

160 00C5RECADDHrmb1 holds addr of last record (HIGH)
161 00C6SPIONErmb1 SPI Low byte - with control bits
162 00C7SPITWOrmb1 SPI High byte - address
163 00C8SEQCNTrmb1 Real Time Interrupt counter
164 00C9TOFCNTrmb1 Timer Overflow Flag counter
165 00CATEMPrmb 1 USED FOR VARIOUS STUFF
166 00CBTEMP1rmb 1 USED IN DELAY TIMERS
167 00CCTEMP2rmb 1 USED only in 16.879 mSEC TIMER
168 00CDTEMPOUTrmb1 Store of acc during SPI activity
169 00CETEMPINrmb1 Store of acc during SPI activity
170 *
171 00CFENDREG
172 *
173 *NOTE THAT THIS POINT CANNOT GO PAST $ WITHOUT LOOKING AT THE STACK USAGE
174 *****
175 *Define equates and other things
176 *****
177 0004 DDR equ 4
178 0020 POWRERUP equ $20
179 00A0 SETREC equ $A0
180 0000 STOPPWRDN equ $10(IAB also set)
181 0030 STOP equ $30 (IAB also set)
182 00E0 SETPLAY equ $E0
183 00B0 REC equ $B0
184 00E8 SETMC equ $E8
185 00F8 MC equ $F8
186 00F0 PLAY equ $F0
187 0030 RINT equ $30 (IAB also set)
188 0070 READADR equ $70 (no RUN, IAB set)
189
190
191 *
```

**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 ISD33000 that is probably not needed because an interrupt from the ISD33000 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 ISD33000 is back HIGH. If it is not, then an early chip revision of the ISD33000 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 SET-PLAY opcode with an address of zero and then a STOP opcode to clear the address counter in the ISD33000 and clear the overflow interrupt.

```

192 *****
193 *Interrupt servce routines and subroutine
194 *****
195 00CF.CODE
196 00CF.RELATIVE
197 0300org RomArea
198 *
```

```

199          *
200 0300          TIMESVC
201          *
202          *NOTE: Fop=.5587302 uSec (for a color burst xtal)
203          *
204          *
205
206 0300
207 0300 80          rti
208 0301
209 0301swisvc
210 0301 80          rti
211          *
212
213 *-----
214 * START OF EXTERNAL INTERRUPT SERVICE ROUTINE
215 *-----
216 *
217
218 *The external int comes from the ISD33000. We first execute a
219 *STOP command to try to clear the interrupt.
220 0302extsvc
221 0302 10 01bsetPD,portbpower down ISD2500 audio chip
222
223 0304 1D 00bclrLOWBAT,portaturn off the LED
224 0306 17 C0bclrOPERATE,R00clear the OPERATE flag
225
226 0308 A6 30lda#STOP
227 030A CD 04 5EjsrSPI_8stop the operation
228
229 030D CD 04 C7jsrTPBNdelay
230
231 *We should have a cleared int by now. If it's not cleared,
232 *we must be in Overflow.
233
234 0310 2F 11bihENDINTBranch if interrupt line is HIGH
235
236 *We in Overflow. Start play at zero then immediately stop it.
237 *This should clear the interrupt in a beta ISD33000 device
238
239 0312 3F C2clrADDRL
240 0314 3F C3clrADDRH
241 0316
242 0316 A6 E0lda#SETPLAYset up to play message with addr
243 0318 CD 04 2EjsrADDADDRadd the address to it
244 031B
245 031B CD 04 6BjsrSPI_16start the playback
246
247 *Now do the STOP
248 031E A6 30lda#STOP
249 0320 CD 04 5EjsrSPI_8stop the operation
250
251 0323ENDINT
252 0323 80rti
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 “dummed out” so that this otherwise useless code is not executed. The mask option register must be programmed by the EEPROM programmer and must be set up manually in most programmers. Note also that the STOP command in line #321 is dummed 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 0324reset
268 *
269 0324  A6 40USERlda#portaDDRset up to read push buttons
270 0326  B7 04staporta+DDR
271 0328  A6 1Flda#portbDDR
272 032A  B7 05staportb+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 00lda#$00Turn the LED off
280 032E  B7 00staporta
281
282 0330  A6 07lda#$07SI=LOW+SO=LOW+SK=LOW+SSBAR=HIGH+
283 *      PLAYREC=HIGH + PD=HIGH
284 0332  B7 01staport b
285
286 0334  A6 FClda#$FC
287 0336  B7 10staPDRAenable pull-down res PA0 & PA1
288 0338  A6 3Flda#$3F
289 033A  B7 11staPDRBenable no pull downs on port B
290 033C  3F C0clrR00
291 033E  3F C2clrADDRL
292 0340  3F C3clrADDRH
293 0342  3F C4clrRECADDL
294 0344  3F C5clrRECADDH
295 0346  3F CAclrTEMP
296
297 0348  5Fclrx
298
299 0349  A6 0Clda#$0CClear and disable timer int.

```

```

300 034B B7 08staTSCR
301
302
303 ;***** INITIALIZE ISD3300 *****
304 034D
305 034D A6 20lda#POWRERUP
306 034F CD 04 5EjsrSPI_8power up the ISD33000
307
308 0352 A6 20lda#POWRERUP
309 0354 CD 04 2EjsrADDADDRadd pwr up info to the addr info
310 0357
311 0357 CD 04 6BjsrSPI_16send in an address of all zeros
312
313 035A CD 04 A1jsrFLASH1blink the led once
314
315 035D 9Crspreset the stack pointer
316 035E 9Aclienable interrups and GO!
317 035F
318
319 035FSTOPPIT
320
321 * stop
322
323

```

**Main Loop:** The main loop simply looks for push-button closures on six 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 its time in this loop waiting for a switch closure.

```

324 *****
325 * TOP OF THE LOOP. EVERYTHING STARTS FROM HERE *
326 * *
327 *****
328 ***** WAIT FOR AN INPUT HERE *****
329 *
330 * MAIN LOOP WAITING FOR A SWITCH CLOSURE
331 *****
332 * Read the pushbuttons and branch if they are "alive".
333 035READ
334 035F 00 00 11brsetL2,porta,GOTOBEG
335 0362 02 00 1EbrsetL5,porta,SKIP2NXT
336 0365 04 00 3EbrsetL6,porta,PLAYNXT
337 0368 06 00 60brsetL1,porta,RECIT
338 036B 08 00 0FbrsetL4,porta,STOPITX
339 036E 0E 00 0FbrsetL7,porta,PLAYLASX
340
341 0371 20 ECbraREAD
342
343

```

**GOTOBEG:** The "Go To Beginning" routine only sets the flag bit MAKITO (for Make It Zero) located in the R00 flag register. The MAKITO bit will be used to indicate to other routines that record or playback should start



from address zero of the ISD33000 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 three 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.

```

344 ***** GO TO Beginning *****
345
346 *The Go To Beginning Button is pushed
347 0373GOTOBEG
348 0373 06 C0 53brsetOPERATE,R00,READZ
349 0376 14 C0bsetMAKEIT0,R00
350
351
352 0378 CD 04 ACjsrFLASH2blink the led twice
353
354 037B 20 4CbraREADZ
355 *****
356

```

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

```

357 037DSTOPITX
358 037D CC 03 F9jmpSTOPIT
359
360 0380PLAYLASX
361 0380 CC 04 05jmpPLAYLAST
362

```

**SKIP2NXT:** This routine sends a SETMC<sup>1</sup> opcode to the ISD33000 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 ISD33000 to be modified so that it is left pointed at the "next" message in the device.

```

363 ***** Message Cueing *****
364
365 *Now do a message cueing cycle . . . but first, do we do it from zero?
366 0383SKIP2NXT
367 0383 06 C0 43brsetOPERATE,R00,READZ
368
369 0386 05 C0 13brclrMAKEIT0,R00,NOCUE0if this flag clear,
370 * no cue from zero
371

```

1. See the "Opcode Summary" in the ISD 33000 Data Sheet for an explanation of the instructive opcodes.

```

372 *We do a message cueing cycle from address zero
373 0389 15 C0bclrMAKEIT0,R00
374 038B
375 038B 3F C2clrADDRL
376 038D 3F C3clrADDRH
377 038F
378 038F CD 04 A1 jsrFLASH1blink the LED once
379
380 0392 A6 E8lda#SETMCset up to do a msg cue + addr
381 0394 CD 04 2EjsrADDADDRadd the address to it
382 0397
383
384 0397 CD 04 6BjsrSPI_16do msg cue (but don't play)
385
386
387 039A 20 2DbraREADZdebounce and leave
388
389 *we do a message cueing cycle from the last address
390 039CNOUCUE0
391 039C CD 04 A1jsrFLASH1blink the LED once
392
393 039F A6 F8lda#MC
394 03A1 CD 04 5EjsrSPI_8do msg cue (but don't play)
395
396 03A4 20 23braREADZdebounce and leave
397
398 *****
399 03A6
400
401

```

---

**PLAYNXT:** This routine sends a SETPLAY opcode to the ISD33000 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 ISD33000's internal address pointer. While playback is occurring, only the STOP push button will be recognized by the software.

---

```

402 ***** Play Next Message *****
403
404 *Play the next message . . . unless Go to Beginning has been pressed
405 03A6PLAYNXT
406 03A6 06 C0 79brsetOPERATE,R00,READXcheck if we already oper.
407 03A9 16 C0bsetOPERATE,R00
408
409 03AB 12 01bsetPLAREC,portbPut the ISD2500 into Play
410 03AD 11 01bclrPD,portbPower up the ISD2500
411
412 03AF 05 C0 10brclrMAKEIT0,R00,NOPLA0if flag clear, no play fm zero
413
414 *We play a message from address zero
415
416 03B2 15 C0bclrMAKEIT0,R00clear the makit it zero flag
417 03B4 3F C2clrADDRL
418 03B6 3F C3clrADDRH

```

```

419 03B8
420 03B8 A6 E0lda#SETPLAYset up to play a msg with addr
421 03BA CD 04 2EjsrADDADDRadd the address to it
422 03BD
423 03BD CD 04 6BjsrSPI_16start the playback
424
425 03C0 1C 00bsetLOWBAT,portaturn on the LED
426
427 *Now fall through from starting at zero and send 8 more bits to continue play
428
429 *we really play the "next message"
430 03C2 NOPLA0
431 03C2 A6 F0lda#PLAYload for play with IAB bit set
432 03C4 CD 04 5EjsrSPI_8start the playback
433
434 03C7 1C 00bsetLOWBAT,portaturn on the LED
435
436 03C9 20 57READZbraREADXdebounce and leave
437
438 *****
439 03CB
440
441

```

**RECIT:** This routine sends a SETREC opcode to the ISD33000 with an address of zero if the MAKITO bit is set. This causes record to start at address zero. If the MAKITO flag bit is not set, a record operation begins at whatever address is currently in the ISD33000'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 MAKITO bit is not set, i.e., the internal address is to be used for record, then this address is read out of the ISD33000 using the SPIIN subroutine (which will be explained later). This address is stored in a set of registers<sup>1</sup> so that the PLAYLAST routine can make use of it. If the MAKITO bit is set, then a zero is stored in these registers.

```

442 ***** Record Next Message *****
443
444 *Record the next message . . . unless Go to Beginning has been pressed
445 03CBRECIT
446
447 03CB 06 C0 54brsetOPERATE,R00,READXcheck to see if already oper
448 03CE 16 C0bsetOPERATE,R00
449
450 03D0 13 01bclrPLAREC,portbPut the ISD2500 into Record
451 03D2 11 01bclrPD,portbPower up the ISD2500
452
453 03D4 05 C0 16brclrMAKEIT0,R00,NOREC0if flag clear, no rec from 0
454
455
456 *We record message from address zero
457 03D7 15 C0bclrMAKEIT0,R00clear the makit it zero flag
458 03D9 3F C2clrADDRL

```

1. Since the ISD33000 family uses an address longer than 8 bits, it takes two 8-bit registers to share the address. In the software, these registers are called RECADDL and RECADDH.

```

459 03DB 3F C3clrADDRH
460
461 *Also clear the "record next" address bytes
462
463
464 03DD 3F C4clrRECADDL
465 03DF 3F C5clrRECADDH
466 03E1
467 03E1 A6 A0lda#SETRECset up to record a message with addr
468 03E3 CD 04 2EjsrADDADDRadd the address to it
469 03E6
470 03E6 CD 04 6BjsrSPI_16start the recording
471
472 03E9 1C 00bsetLOWBAT,portaturn on the LED
473 03EB
474 03EB 20 04braNOREC1
475
476 *Now send 8 more bits to continue record
477
478 *we really Record the "next message"
479 03EDNOREC0
480 03ED A6 70lda#READADR
481 03EF AD 4CbsrSPIINGo read the current addr before rec
482
483
484 03F1NOREC1
485 03F1 A6 B0lda#REC
486 03F3 AD 69bsrSPI_8start the recording
487
488 03F5 1C 00bsetLOWBAT,portaturn on the LED
489
490 03F7 20 29braREADXdebounce and leave
491
492 *****
493 03F9
494
495

```

---

**STOPIT:** This routine sends a STOP opcode to the ISD33000. 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.

---

```

496 ***** Stop Recording or Playing *****
497
498 *Stop whatever we are doing . . . but don't corrupt the address register
499 03F9 STOPIT
500 03F9 A6 30lda#STOP
501 03FB AD 61bsrSPI_8stop the operation
502
503 03FD 1D 00bclrLOWBAT,portaturn off the LED
504
505 03FF 17 C0bclrOPERATE,R00turn off the "Operate" flag
506
507 0401 10 01bsetPD,portbPower down the ISD2500
508

```

```

509 0403 20 1DbraREADX
510
511 *****
512 0405
513

```

---

**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 ISD33000 so that playback now begins at the address where the "last" recording started from.

---

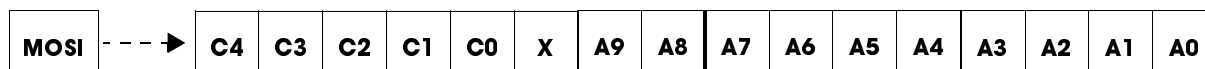
```

514
515 ***** Play Next Message *****
516
517 *Play the last message we recorded
518 0405 PLAYLAST
519 0405 06 C0 1A brsetOPERATE,R00,READXcheck if we already oper
520 0408 16 C0 bsetOPERATE,R00
521
522 040A 12 01bsetPLAREC,portbPut the ISD2500 into Playback
523 040C 11 01bclrPD,portbPower up the ISD2500
524
525
526 *Go get the address we stored off just before we recorded (It's in
527 *RECADDH + RECADDL). Put it in the address registers and GO!
528
529 040E B6 C4ldaRECADDL
530 0410 B7 C2staADDRL
531
532 0412 B6 C5ldaRECADDH
533 0414 B7 C3staADDRH
534
535 0416 A6 E0lda#SETPLAYset up to play a message with addr
536 0418 CD 04 2EjsrADDADDRadd the address to it
537 041B
538 041B CD 04 6BjsrSPI_16start the playback
539
540 041E 1C 00bsetLOWBAT,portaturn on the LED
541
542 0420 20 A0braNOPLA0Send 2nd command to set IAB bit
543
544 *****
545 0422
546
547
548 *after doing any and all the stuff,you arrive here to debounce the switches
549 0422 B6 00READXldaporta
550 0424 A4 9Fand#$9Fonly look at the push buttons
551 0426 26 FAbneREADXloop until the button is released
552
553 *OK, the button has been released
554 0428 CD 04 C7jsrTPBNgo wait for awhile
555 042B CC 03 5Fjmp READ
556

```

557  
558  
559

**ADDADDR Subroutine:** Whenever we do any of the “SET” operations, we have to add a 10 bit address to the 5 bit ISD33000 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:



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 ADDR\_L and ADDR\_H bytes. ADDR\_L contains A7–A0 and ADDR\_H 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. ADDR\_L is next written to SPITWO. SPIONE and SPITWO are the two bytes that will be sent out of the SPI port of the micro.

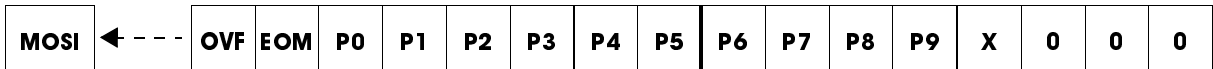
```

560 *****
561 *****
562 *   SUBROUTINES START HERE!
563 *****
564 *****
565
566
567 *****
568 *SUBROUTINE ADDADDR
569 *****
570             *This Subroutine adds the address to the control bits already
571             *present in SPIONE. In doing so, it adds a second byte for
572             *16 bit transfers into the ISD33000. This subroutine is
573             *called with a control byte in the accumulator.
574
575 042EADDADDR
576 042E   B7 C6staSPIONEput the accumulator into SPI one
577
578 0430   B6 C3ldaADDRHget the upper byte of the address
579 0432   A4 03and#$03
580 0434   BA C6oraSPIONE
581 0436   B7 C6staSPIONE
582 0438
583 0438   B6 C2ldaADDRLGo get the lower byte
584 043A   B7 C7staSPITWO
585
586 043C   81rts
587
588

```

589

**SPIIN Subroutine:** This subroutine is used to read the address data out of the ISD33000 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 ISD33000 SPI port (MISO) looks like:



The OVF bit is presented to the MISO pin as soon as the Slave Select pin goes LOW. The first clock input to the ISD33000 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 SHIFIT 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–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 ISD33000’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.

```

590                                     *****
591 *SUBROUTNE SPIIN
592                                     *****
593 * Read the SPI data from the chip, throw out the OVF and EOM
594 * bits, and store the address just retrieved in RECADDL and
595 * RECADDH. The accumulator comes in with the proper 5 bits
596 * of control so that nothing is disturbed.
597
598
599 043DSPIIN
600 043D 17 01bclrSK,portbMake sure the clock starts LOW
601 043F 15 01bclrSSBAR,portbDrop Slave Select
602
603 0441 B7 C6staSPIONESave the accumulator for later
604
605                                     *we shift zeros in first
606
607 0443 3F CAclrTEMPput all zeros in temp
608 0445 AD 39bsrSHIFITIT
609
610 0447 B6 CEldaTEMPINget the data just shifted in
611 0449 B7 C4staRECADDL
612
613                                     * note that EOM bit is still in RECADDL. We will

```

```

614                 * fix this later.
615
616 044B B6 C6ldaSPIONEgo get the original accumulator
617 044D B7 CAstaTEMP
618
619 044F AD 2FbsrSHIFTTITgo do the second byte
620
621
622 * TEMPIN now has upper 3 bits of the address in it. We will
623 * shift 1 of those bits into RECADDL. The last 2 bits remain
624 * in TEMPIN and this becomes RECADDH.
625
626 0451 34 CElsrTEMPIN
627 0453 36 C4rorRECADDL
628
629 0455 B6 CEldaTEMPIN
630 0457 B7 C5staRECADDH
631
632 0459 14 01bsetSSBAR,portbSlave Select goes HIGH to end
633                 *cycle
634 045B 19 01bclrSO,portbLeave this LOW when finished
635 045D 8lrts
636
637

```

---

**SPI\_8 Subroutine:** This subroutine is used to shift 8 bits into the MOSI pin of the ISD33000. The accumulator brings in the byte to be shifted. The SHIFTTIT subroutine does the actual shifting of the data.

---

```

638                 *****
639 *SUBROUTNE SPI_8                 *****
640                 *****
641                 *SPI driver subroutine - outputs 8 bits. This routine sends
642 *the SPIONE byte out the bit banded SPI port.
643
644 045ESPI_8
645 045E 17 01bclr SK,portb Make sure the clock starts LOW
646 0460 15 01bclr SSBAR,portb Drop Slave Select
647
648 0462 B7 CAsta TEMP
649 0464 AD 1Absr SHIFTTIT
650
651 0466 14 01bset SSBAR,portb Slave Select goes HIGH to end
652 * cycle
653 0468 19 01bclr SO,portb Leave this LOW when finished
654 046A 8lrts
655
656

```

---

**SPI\_16 Subroutine:** This subroutine is used to shift 16 bits into the MOSI pin of the ISD33000. The SPITWO byte is shifted in first and the SPIONE bit is shifted in second. The SHIFTTIT subroutine does the actual shifting of the data.

---

```

657                 *****

```



```

658             *SUBROUTNE SPI_16
659             *****
660             *SPI driver subroutine - outputs 16 bits. This routine sends
661 *the SPIONE and SPITWO bytes out the bit banged SPI port.
662
663 046B SPI_16
664 046B 17 01bclr SK,portb      Make sure the clock starts LOW
665 046D 15 01bclr SSBAR,portb   Drop Slave Select
666
667 046F B6 C7lda SPITWO
668 0471 B7 CAsta TEMP
669
670 0473 AD 0Bbsr SHIFTIT
671
672 0475 B6 C6lda SPIONE
673 0477 B7 CAsta TEMP
674
675 0479 AD 05bsr SHIFTIT
676 047B
677 047B 14 01bset SSBAR,portb   Slave Select goes HIGH to end
678             *cycle
679 047D 19 01bclr SO,portb      Leave this LOW when finished
680 047F 81rts
681
682
683

```

---

**SHIFTIT Subroutine:** This subroutine does all the shifting of data into and out of the SPI port of the ISD33000. 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.

---

```

684 *****
685 *SUBROUTNE SHIFTIT
686 *****
687 *This subroutine shifts out 8 bits from the SPI Port
688
689 0480SHIFTIT
690 0480 A6 08lda#8
691 0482 B7 CBstaTEMPlput an 8 count into temp 1
692
693 0484 B6 CAldaTEMP
694
695 0486 44OUTAGNlsrshift the LSB into carry bit
696 0487 24 04bccOUTZERO
697
698 0489 18 01OUTONEbsetSO,portboutput a "1"
699 048B 20 02braCONTSPI
700
701 048D 19 01OUTZERObclrSO,portboutput a "0"
702
703
704 *Now toggle the clock to shift the data
705 048F 16 01CONTSPIbsetSK,portb
706 0491 17 01bclrSK,portb

```

```

707
708 *now look at incoming data
709
710 0493 0A 01 03brsetSI,portb,INONE
711 0496 98clcit's a 0, clear carry bit
712 0497 20 01braCONTIN
713
714 0499 99INONEsecit's a 1, set the carry bit
715
716 049A 36 CECONTINrorTEMPINbring the bit into the acc
717
718 049C 3A CBdecTEMP1
719 049E 26 E6bneOUTAGN
720 04A0
721 04A0 81rts
722

```

---

**FLASH1 Subroutine:** The FLASH1 subroutine blinks the LED once.

---

```

723 *****
724 *SUBROUTNE FLASH1
725 *****
726 *This subroutine flashes the LED once
727
728 04A1FLASH1
729 04A1 1C 00bsetLOWBAT,portaturn on the LED
730 04A3
731 04A3 A6 96lda#150
732 04A5 B7 CBstaTEMP1use 150 at the delay
733
734 04A7 AD 22bsrKEYDLYdelay
735 04A9 1D 00bclrLOWBAT,portaturn off the LED
736 04AB 81rts
737
738
739

```

---

**FLASH2 Subroutine:** The FLASH2 subroutine blinks the LED twice.

---

```

740 *****
741 *SUBROUTNE FLASH2
742 *****
743 *This subroutine flashes the LED twice
744 04AC FLASH2
745 04AC 1C 00bset LOWBAT,porta turn on the LED
746
747 04AE A6 96lda #150
748 04B0 B7 CBsta TEMP1 use 150 at the delay
749
750 04B2 AD 17bsr KEYDLY delay
751 04B4 1D 00bclr LOWBAT,porta turn off the LED
752
753 04B6 A6 96lda #150

```

```

754 04B8 B7 CBsta TEMP1 use 150 at the delay
755
756 04BA AD 0Fbsr KEYDLY delay
757 04BC 1C 00bset LOWBAT,porta turn on the LED
758
759 04BE A6 96lda #150
760 04C0 B7 CBsta TEMP1 use 150 at the delay
761
762 04C2 AD 07bsrKEYDLYdelay
763 04C4 1D 00bclrLOWBAT,portaturn off the LED
764 04C6 81rts
765
766
767

```

---

**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.

---

```

768 *****
769 *SUBROUTINE TPBN
770 *****
771 *USED FOR TIMING VARIOUS STUFF IN THE PROGRAM
772 *
773
774 *TPBN IS THE PUSH BUTTON DELAY IN PUSH BUTTON MODE
775 04C7A6 FA TPBNlda #250 DELAY 250 MILLISECONDS (approx)
776 04C9 B7 CBstaTEMP1
777
778 04CB B6 CBKEYDLYldaTEMP1(3)
779 04CD 27 0AbeqFINDLY(3)
780 04CF 3A CBdecTEMP1(5)
781
782
783
784 *****
785 *WAIT1 GENERATES A DELAY (1.154 mSEC)
786 *****
787 *WAIT1 GENERATES A 1.154 MILLISECOND DELAY. IT DOES NOT DISTURB THE
788 *ACCUMULATOR. FROM A BSR (CALLING THIS ROUTINE) THROUGH THE RTS (ENDING
789 *THIS ROUTINE, IT TAKES 2065 CYCLES x 4.3656 uSEC = 1.154 MSec. THIS THING
790 *LOOPS 256 TIMES.
791
792 04D1 3F CC WAIT1clrTEMP2(5)
793 04D3 3C CC WAIT2incTEMP2(5)
794 04D5 26 FCbneWAIT2(3)
795
796 04D7 20 F2bra KEYDLY(3)
797
798 04D9 81FINDLYrts
799
800
801

```

802 \*  
 803  
 804  
 805  
 806  
 807  
 808  
 809 \* NOTE: THE LAST USABLE ADDRESS IS \$7CF  
 810  
 811  
 812  
 813  
 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=   00F8393
119   MOR=   07F1
804   MSGTABL04DA
390   NOCUE0 039C369
430   NOPLA0 03C2412 542
479   NOREC0 03ED453
484   NOREC1 03F1474
135   OPERATE =0003224 348 367 406 407 447 448 505 519 520
695   OUTAGN0486719
698   OUTONE 0489
701   OUTZERO048D696
Pre   PAGE00000
 69   PD=   0000221 410 451 507 523
117   PDRA=  0010287
118   PDRB=  0011289
 70   PLAREC= 0001409 450 522
186   PLAY=  00F0431
132   PLAYING= 0000
518   PLAYLAST0405361
360   PLAYLASX0380339
405   PLAYNXT03A6336
178   POWRERUP= 0020305 308
806   PTTABL04DA
130   R0000C0224 290 348 349 367 369 373 406 407 412
    416 447 448 453 457 505 519 520
141   R0100C1
 59   RAC=   0005
333   READ035F341 555
188   READADR= 0070480
549   READX0422406 436 447 490 509 519 551
436   READZ03C9348 354 367 387 396
183   REC =  00B0485

```

```

160 RECADDH00C5294 465 532 630
159 RECADDL00C4293 464 529 611 627
133 RECDING= 0001
445 RECIT03CB337
187 RINT= 0030
 94 RamArea = 00C0128
 96 RomArea= 0300197
163 SEQCNT00C8
184 SETMC= 00E8380
182 SETPLAY= 00E0242 420 535
179 SETREC = 00A0 467
689 SHIFTTIT 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

```

## APPENDIX

Figure 1 is used to adapt the COPS socket on the ISD-ES302 demo board to the Motorola MC68HC705J1A microcontroller. Plug J2 plugs into the COPS socket and the Motorola MC68HC705J1A plugs into the J1 socket. Also, R7 ( $4.7\ \Omega$ ), which connects to the Interrupt output of the ISD33000, must be shunted with a 1N914 signal diode with the cathode toward the ISD chip. The Motorola microcontroller cannot see interrupts unless this is done.

**Figure 1: Adapter Board Schematic**

