Building An HP-71B IR Module

Visitors to the HP Corvallis plant during the August 1988 conference will remember seeing an HP-71B Infrared module driving an 82240A IR printer. Here are a few details. The output is controlled by an HP-71 LEX file which implements a keyword: "PIR". This stands for "PRINTER IS IR" but is easier to implement than the parsing required to handle that verbose syntax. After PIR is executed a poll handler will trap all output directed to the print device (eg, PRINT, PRINT USING, PLIST). The next "PRINTER IS" statement will terminate this operation.

The circuit enclosed is annotated to show both options that we have used here at HP. The basic operation of the circuit is to use the OR4 line (pin 8 in card reader slot) to gate a 32 KHZ oscillator to an IR LED. When OR4 is low, no IR is output; when OR4 is high, IR is pulsed out as a 32KHz square wave. A flip-flop is added to guarantee that only entire pulses are sent out.

The crystal draws noticable power. Battery life will be slightly reduced if left running all the time. This is not a problem if the calculator is plugged into an AC adapter, but could be a problem for battery operated situations. The part of the circuit in the dashed-line area is designed to lower battery usage by leeching power from the strobe line, instead of using the 5.0V supply. Thus when strobe is active (i.e. when the calculator is running), this quickly charges up the capacitor (actually 100 pF may be too small), which will enable the oscillator to do it's thing. When the calculator shuts down (either light sleep or deep sleep), the resistor drains the capacitor and the input to the NAND gate drops, thus disabling the crystal and saving power. The big problem with this circuit is that if the machine is shut down for a long time, the crystal may be slow in coming up (crystals are that way); this may cause output sent to the IR port immediately after wakeup to be trashed.

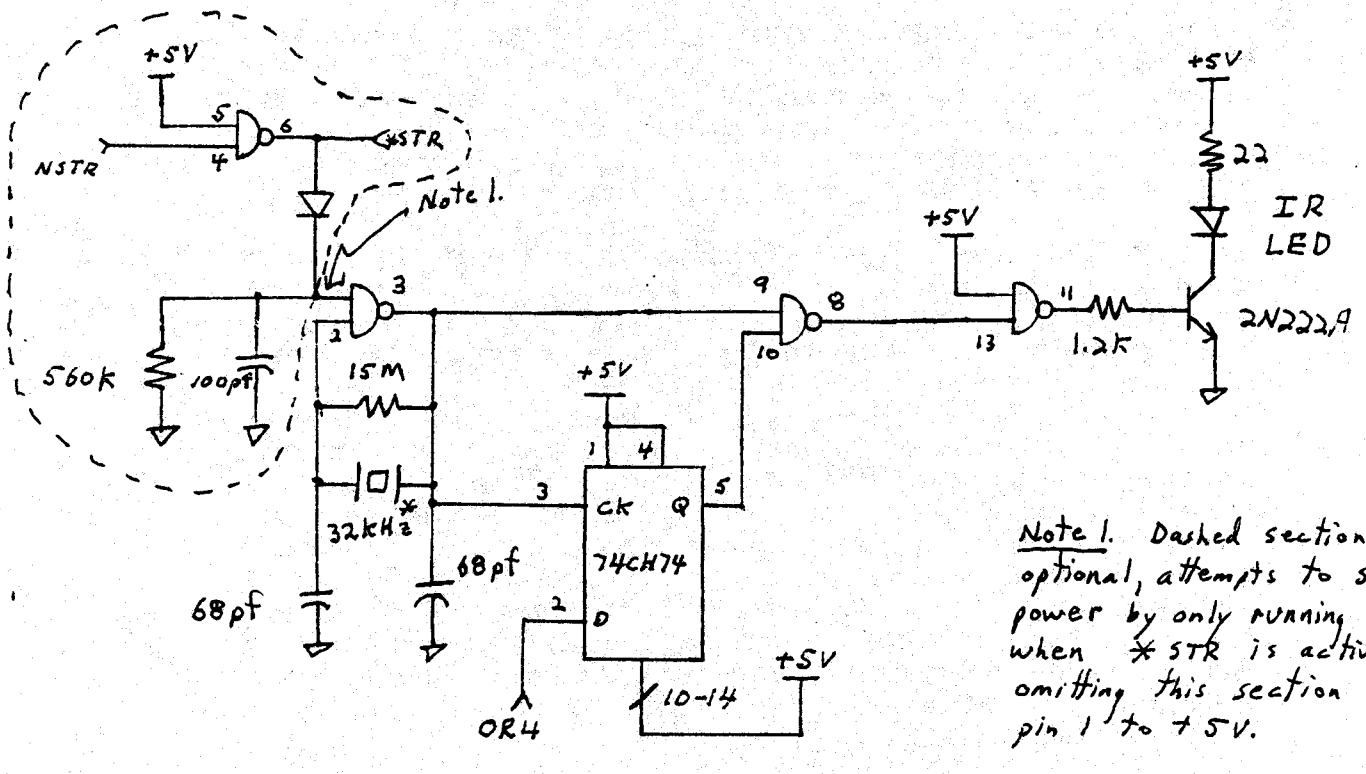
There are other ways of doing oscillator circuits that have lower power demands and come up quicker. We never had the time to design such a circuit.

Another approach would be to eliminate the "battery saver" section, and just pull that NAND gate input high. That is the way my IR module is wired. An improvement might be to put a manual switch to completely cut the Vdd (+5V) supply when the printer is not in use. I don't know if this is feasible for general use.

The prototypes that we have built have been very ugly. They have been made out of recycled card reader PC boards and module casings. There are two IC's and several discrete components that must be crammed into the module. If anyone designs their own PC board it would be much easier and reliable. We have been turning the IC's "belly-up" and soldering wires and components between the leads. Like I say, it's ugly.

I would suggest a breadboard be built just to make sure you have a circuit that works, then figure out how to make it fit in the module. If you end up making a special PC board, or find a better way of putting all this circuitry in the module I would apapreciate a couple of boards wo we can make some better looking units for our own use.

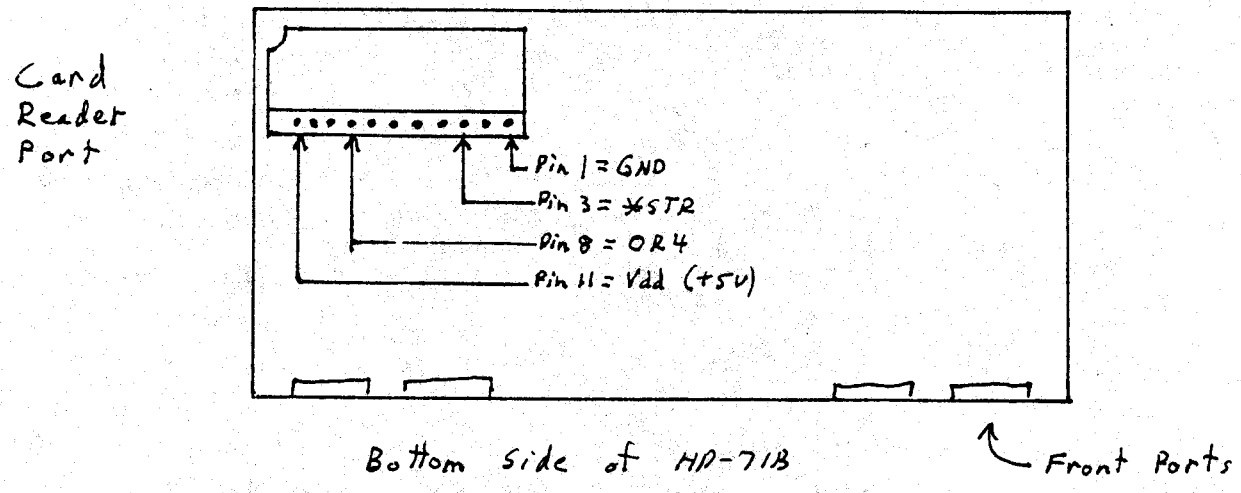
HP-71B IR MODULE



Note 1. Dashed sectional optional, attempts to save power by only running crystal when *STR is active. If omitting this section tie pin 1 to +5V.

1 - 74HC74
1 - 74HC00
1 - 2N222A
1 - *32,768 Hz Crystal
1 - 68 pf caps.
1 - 15MΩ Res.
1 - 22Ω Res.
1 - 1.2kΩ Res.
1 - IR LED

HP-71B Card Reader Pin Locations



```
Saturn Assembler    LEXFILE -- Wed Mar 19 17:02:13 PST Wed Apr 26 09:15:06 1989
Ver. 1.54, 01/18/89                                    pir.a          Page    1

   1                        TITLE LEXFILE -- Wed Mar 19 17:02:13 PST 1986
   2 00008                  REL     8
   3                  *  This file was generated on Wed Mar 19 17:02:15 1986
   4                  *  File Header
   5 00008 05942            NIBASC \PIRLEX  \       File Name
           5C454
           850202
   6 00018 0000             CON(4) =fLEX            File Type
   7 0001C 00               NIBHEX 00               Flags
   8 0001E 2071             NIBHEX 2071             Time
   9 00022 91306            NIBHEX 913068           Date
           8
  10 00028 CD200            REL(5) FILEND           File Length
  11                 *
  12 0002D C5               NIBHEX C5               Id
  13 0002F 38               CON(2) 131              Lowest Token
  14 00031 38               CON(2) 131              Highest Token
  15 00033 00000            NIBHEX 00000            End of lex table chain
  16                 *
  17 00038 F                NIBHEX F                Speed table omitted
  18 00039 7100             CON(4) (TxTbSt)+1-(*)   Offset to text table
  19 0003D 0000             CON(4) 0                No message table
  20 00041 A1000            REL(5) POLHND           Offset to poll handler
  21                        STITLE M a i n   T a b l e

Saturn Assembler    LEXFILE -- Wed Mar 19 17:02:13 PST Wed Apr 26 09:15:06 1989
Ver. 1.54, 01/18/89 M a i n   T a b l e              pir.a          Page    2

  22                 *  Main Table
  23 00046          =xrom5C
  24                 *
  25 00046 000              CON(3)    0           83  PIR
  26 00049 BF100            REL(5) =PIR
  27 0004E D                NIBHEX D
  28                        STITLE T e x t   T a b l e

Saturn Assembler    LEXFILE -- Wed Mar 19 17:02:13 PST Wed Apr 26 09:15:06 1989
Ver. 1.54, 01/18/89 T e x t   T a b l e              pir.a          Page    3

  29                 *  Text Table
  30 0004F          TxTbSt                          Text table start
  31                 *
  32 0004F 5                NIBHEX 5                PIR
  33 00050 05942            NIBASC \PIR\
           5
  34 00056 38               NIBHEX 38
  35 00058 1FF      TxTbEn  NIBHEX 1FF              Text termination
  36                 *
  37                 * Poll handler goes here.  Handler for VER$ poll is
  38                 * provided
  39                 *
  40 0005B 969      POLHND ?B=0   B                 VER$ poll?
  41 0005E 50               GOYES  hVER$0           Yes.
  42 00060 503              GONC   hVER$2           No.  To hVER$2 w/carry clear.
  43 00063 11B      hVER$0 C=R3
  44 00066 135              D1=C
  45 00069 112              A=R2
  46 0006C 1CB              D1=D1- (VER$en)-(VER$st)-2
  47 0006F 137              CD1EX
  48 00072 8B6              ?A>C   A
  49 00075 A1               GOYES  hVER$1
  50 00077 135              D1=C
  51 0007A 10B              R3=C
  52                 *
  53                 **!! LCASC text to be returned for VER$ here
  54                 * Include a leading blank!!
  55                 *
  56 0007D 3B24A    VER$st LCASC  \ PIR:B\
           32594
           0502
  57 0008B 15DB     VER$en DAT1=C (VER$en)-(VER$st)-2
  58 0008F 00       hVER$1 RTNSXM
  59                 *
  60                 **!! Continue poll handler here: Carry is clear, VER$ poll
  61                 *  has been handled.
  62 00091 3100     hVER$2 LC(2)   =pPRTIS
  63 00095 961              ?B=C   B
  64 00098 40               GOYES  hPRTIS
  65 0009A 00               RTNSXM
  66                 *
  67                 *
  68                 **!! LEXFILE code goes here
  69                 *
  70                 *
  71                 **!! LEXFILE code goes here
  72                 *
  73 0009C 1B000    hPRTIS D0=(5)  =IS-PRT
           00
  74 000A3 D0               A=0    A
  75 000A5 15A3             A=DAT0 4
  76 000A9 34F94            LCHEX  0F49F
           F0
  77 000B0 8A2              ?A=C   A
  78 000B3 40               GOYES  hPRTIS2
  79 000B5 00               RTNSXM
  80 000B7          hPRTIS2

Saturn Assembler    LEXFILE -- Wed Mar 19 17:02:13 PST Wed Apr 26 09:15:06 1989
Ver. 1.54, 01/18/89 T e x t   T a b l e              pir.a          Page    4

  81 000B7 6010             GOTO   GetAddr
  82 000BB          hPRTIS3
  83 000BB 07               C=RSTK
  84 000BD DA               A=C    A
  85 000BF D2               C=0    A
  86 000C1 305              LC(1)  5
  87 000C4 CA               A=A+C  A               Skip Part3 offset
  88 000C6          HandlerDone
  89 000C6 03       HPart3 RTNCC
  90                 *
  91 000C8          GetAddr
  92 000C8 7FEF             GOSUB  hPRTIS3         Push Handler address and
  93 000CC AFFFF            REL(5) HPart3                  jump to hPRTIS3
  94 000D1          Handler
  95                 * Entry:
  96                 *       A(A)=Length of buffer in bytes
  97                 *       D(A)=Start address of buffer
  98                 *       P=0
  99                 * Exit:
 100                 *       D1 points past last character sent
 101                 *       P=0
```

```
102                     *
103 000D1 D6                    C=A     A
104 000D3 06                    RSTK=C                  Save number of bytes on stack
105 000D5 1F000         00      D1=(5)  (=IS-PRT)+2
106 000DC 147                   C=DAT1  A               Read BITTIM
107 000DF DF                    CDEX    A               D=Timing info,C=Buffer Ptr
108 000E1 135                   D1=C                    D1 points to bytes
109 000E4               HandlerLoop
110 000E4 07                    C=RSTK                  Recall number of bytes
111 000E6 CE                    C=C-1   A               Decrement count
112 000E8 4DD                   GOC     HandlerDone     Exit loop when done
113 000EB 06                    RSTK=C                  Save decremented count
114 000ED 14B                   A=DAT1  B               Read byte to output
115 000F0 171                   D1=D1+  2               Move pointer past byte
116                     *
117                             STITLE  OUTBYTE
```



```
118                     *
119                     *                               BA98 7654 3210
120                     *
121                     * Bit 1 - B,6,5,4,3             1000 0111 1000
122                     MASK1   EQU     #878
123                     * Bit 2 - A,7,6,5,2,1           0100 1110 0110
124                     MASK2   EQU     #4E6
125                     * Bit 3 - 9,7,6,4,2,0           0010 1101 0101
126                     MASK4   EQU     #2D5
127                     * Bit 4 - 8,7,3,1,0             0001 1000 1011
128                     MASK8   EQU     #18B
129                     *
130                     * OUTBYTE -
131                     *       Entry:  A(B)=Byte to output
132                     *               D(3)=Extra cycle count
133                     *               D(2)=Loop count
134 000F3               =OUTBYTE
135
136 000F3 D2                    C=0     A
137 000F5 AE6                   C=A     B
138 000F8 0B                    CSTEX
139 000FA 06                    RSTK=C                  Save ST on RSTK
140 000FC 3187                  LC(2)   MASK1
141 00100 7901                  GOSUB   CountBits
142 00104 450                   GOC     HAM10
143 00107 85B                   ST=1    11
144 0010A 316E          HAM10   LC(2)   MASK2
145 0010E 7BF0                  GOSUB   CountBits
146 00112 450                   GOC     HAM20
147 00115 85A                   ST=1    10
148 00118 315D          HAM20   LC(2)   MASK4
149 0011C 7DE0                  GOSUB   CountBits
150 00120 450                   GOC     HAM30
151 00123 859                   ST=1    9
152 00126 31B8          HAM30   LC(2)   MASK8
153 0012A 7FD0                  GOSUB   CountBits
154 0012E 450                   GOC     HAM40
155 00131 858                   ST=1    8
156 00134               HAM40
157 00134 D2                    C=0     A
158 00136 09                    C=ST
159 00138 DA                    A=C     A
160 0013A 07                    C=RSTK
161 0013C 0A                    ST=C                    Restore status
```

```
162 0013E 2F                    P=      15
163 00140 30B                   LC(1)   11              Twelve bits (plus
164 00143 20                    P=      0                   start "bit")
165 00145 808F                  INTOFF
166 00149 7850                  GOSUB   SendStBit
167 0014D 7450                  GOSUB   SendStBit
168 00151 7050                  GOSUB   SendStBit       Send out start "bit"
169 00155               SendNextBit
170 00155 A34                   A=A+A   X               ; 6
171 00158 441                   GOC     SendOne         ; 10 or 3
172 0015B D2                    C=0     A               ; 7  Filler
173 0015D               SendZero
174 0015D 7570                  GOSUB   HalfBitZero     ; 12
```



```
175 00161 7830                  GOSUB   Cycles41        ; 41  Filler (25+16)
176 00165 7570                  GOSUB   HalfBitOne      ; 12
177 00169 6310                  GOTO    BitDone         ; 11
178 0016D               SendOne
179 0016D 7D60                  GOSUB   HalfBitOne      ; 12
180 00171 7820                  GOSUB   Cycles41
181 00175 7D50                  GOSUB   HalfBitZero     ; 12
182 00179 6300                  GOTO    BitDone         ; 11
183 0017D               BitDone
184 0017D A4E                   C=C-1   S               ; 4
185 00180 54D                   GONC    SendNextBit     ; 10/3
186
187 00183 8080                  INTON
188 00187 32F00                 LC(3)   #00F
189 0018C 801                   OUT=C                   ; return OUT to normal
190 0018F 31D1                  LC(2)   650*671000/15/1000000   Assume fast CPU (671KHz)
191 00193               =SlowPrint                      Add 650us delay between bytes
192 00193 A6E                   C=C-1   B               ; 5
193 00196 5CF                   GONC    SlowPrint       ; 10/3
194 00199 6A4F                  GOTO    HandlerLoop
195
196
197                     *       GOSUB   Cycles41        ; 12
198 0019D                Cycles41
199 0019D AD2                   C=0     M               ; 15
200 001A0 AE2                   C=0     B               ; 5
201 001A3 01                    RTN                     ; 9
202
203                      uSec/HalfBit   EQU     425
204                      OutPulseBit    EQU     #010
205                      Cyc/Sec        EQU     617000
206                      Cyc/Lp         EQU     3+10
207                      uSec/StBitHi   EQU     244-15          Do not simply change this since
208                     *                                       the difference between it and
209                     *                                       uSec/BitHi is embedded in cycle
210                     *                                  count of loop overhead low.
211                      Cyc/StBitHi    EQU     ((Cyc/Sec)*(uSec/StBitHi)+500000)/1000000
212                      StBitOvrhdPr   EQU     12+7+7+2+6+2+6
213                     *       GOSUB;C=0 A;C=0 A; P=0; LC(3);P=;OUT=C
214
215                      StBitOvrhdHi   EQU     6+3-10+6
216                     *                       C=0 X;GONC(no),-GONC(yes), OUT=C
217
218                      StBitOvrhdLo   EQU     7+7+6+4+6+3-10+9
219                     *       C=0 A;C=D A;P=C 3;B=0 WP;P=C 2;GONC(no),-GONC(yes),RTN
220
```

```
221             StBitHiLps      EQU     ((Cyc/StBitHi)-(StBitOvrhdHi)+(Cyc/Lp)/2)/(Cyc/Lp)
222             StBitHiCyc      EQU     (StBitHiLps)*(Cyc/Lp)+StBitOvrhdHi
223             StBitTotal      EQU     (StBitOvrhdPr)+(StBitHiCyc)+(StBitOvrhdLo)
224                            *
225 001A5     SendStBit
226 001A5 D2            C=0     A                       ; 7  Filler
227 001A7 D2            C=0     A                       ; 7  Filler
228 001A9 20            P=      0                       ; 2   Filler
229
230 001AB 32010         LC(3)   (OutPulseBit)           ; 6
231 001B0 26            P=      16-StBitHiLps           ; 2


Saturn Assembler    LEXFILE -- Wed Mar 19 17:02:13 PST Wed Apr 26 09:15:06 1989
Ver. 1.54, 01/18/89 OUTBYTE                         pir.a          Page    7

232 001B2 801           OUT=C                           ; 6  Start Pulse
233 001B5 AB2           C=0     X                       ; 6
234 001B8       StBitLoopHi
235 001B8 0C            P=P+1                           ; 3   Decrement loop counter
236 001BA 5DF           GONC    StBitLoopHi             ; 10/3  Loop back until done
237 001BD 801           OUT=C                           ; 6  Stop Pulse
238 001C0 D2            C=0     A                       ; 7  Filler
239
240 001C2 DB            C=D     A                       ; 7   Copy Lo loop counter
241 001C4 80D3          P=C     3                       ; 6
242 001C8 A91           B=0     WP                      ; 4 (+N where N=extra cycles not
243                                                             counted in low overhead)
244 001CB 80D2          P=C     2                       ; 6
245 001CF       StBitLoopLo
246 001CF 0C            P=P+1                           ; 3   Decrement loop counter
247 001D1 5DF           GONC    StBitLoopLo             ; 10  Loop back until done
248 001D4 01            RTN                             ; 9
249
250
251             uSec/BitHi      EQU     183+15          Do not simply change this since
252             *                                       the difference between it and
253             *                                       uSec/StBitHi is embedded in cycle
254             *                                       count of loop overhead low.
255
256             Cyc/BitHi       EQU     ((Cyc/Sec)*(uSec/BitHi)+500000)/1000000
257             BitOvrhdPr      EQU     6+3+7+12+6+2+6
258             *       A=A+A X; GOC(no); C=0 A; GOSUB ;LC(3);P=;OUT=C
259
260             BitOvrhdHi      EQU     6+3-10+4+4+6
261             *                       C=0 X;GONC(no),-GONC(yes),C=0 XS,C=0 XS, OUT=C
262
263             BitOvrhdLo      EQU     7+6+4+6+3-10+9+11+4+10
264             *       C=D A; P=C 3;B=0 WP; P=C 2; GONC(no);
265             *       -GONC(yes);RTN;GOTO;C=C-1 S;GONC
266
267             BitHiLps        EQU     ((Cyc/BitHi)-(BitOvrhdHi)+(Cyc/Lp)/2)/(Cyc/Lp)
268             BitHiCyc        EQU     (BitHiLps)*(Cyc/Lp)+BitOvrhdHi
269             BitTotal        EQU *   (BitOvrhdPr)+(BitHiCyc)+(BitOvrhdLo)
270
271
272 001D6       HalfBitZero
273 001D6 D2            C=0     A                       ; 7   Filler
274 001D8 28            P=      16-BitHiLps             ; 2
275 001DA 6010          GOTO    BitLoopHi               ; 11
276 001DE       HalfBitOne
277 001DE 32010         LC(3)   (OutPulseBit)           ; 6
278 001E3 28            P=      16-BitHiLps             ; 2

279 001E5 801           OUT=C                           ; 6  Start Pulse
280 001E8 AB2           C=0     X                       ; 6
281 001EB       BitLoopHi
282 001EB 0C            P=P+1                           ; 3   Decrement loop counter
283 001ED 5DF           GONC    BitLoopHi               ; 10/3  Loop back until done
284 001F0 AA2           C=0     XS                      ; 4   Filler
285 001F3 AA2           C=0     XS                      ; 4   Filler
286
287 001F6 801           OUT=C                           ; 6  Stop Pulse
288 001F9 DB            C=D     A                       ; 7   Copy Lo loop counter

Saturn Assembler    LEXFILE -- Wed Mar 19 17:02:13 PST Wed Apr 26 09:15:06 1989
Ver. 1.54, 01/18/89 OUTBYTE                         pir.a          Page    8

289                                                             and extra cycles
290 001FB 80D3          P=C     3                       ; 6
291 001FF A91           B=0     WP                      ; 4 (+N where N=extra cycles not
292                                                             counted in low overhead)
293 00202 80D2          P=C     2                       ; 6
294 00206       BitLoopLo
295 00206 0C            P=P+1                           ; 3   Decrement loop counter
296 00208 5DF           GONC    BitLoopLo               ; 10  Loop back until done
297 0020B 01            RTN                             ; 9
298 0020D       CountBits
299 0020D 0E62          C=C&A   B
300 00211 D1            B=0     A
301 00213       CountBits00
302 00213 96A           ?C=0    B
303 00216 E0            GOYES   CountBits20
304 00218       CountBits10
305 00218 A66           C=C+C   B
306 0021B 5CF           GONC    CountBits10
307 0021E B65           B=B+1   B
308 00221 51F           GONC    CountBits00     (B.E.T.)
309 00224       CountBits20
310 00224 822           SB=0
311 00227 81D           BSRB
312 0022A 832           ?SB=0
313 0022D 00            RTNYES
314 0022F 03            RTNCC
315                     STITLE  PIR statement execution

Saturn Assembler    LEXFILE -- Wed Mar 19 17:02:13 PST Wed Apr 26 09:15:06 1989
Ver. 1.54, 01/18/89 PIR statement execution         pir.a          Page    9

316                    *
317 00231           PIRd
318 00231 8D000             GOVLNG  =OUTELA
          00
319 00238           PIRp
320 00238 03                RTNCC
321                    *
322 0023A 7FFFF             REL(5)  PIRd
323 0023F 9FFFF             REL(5)  PIRp
324 00244           PIR
325 00244 1B000             D0=(5)  =IS-PRT
          00
326 0024B 33F94             LCHEX   F49F
```

65

```
                F
327 00251 15C3          DAT0=C  4
328                 *Calculate loop and cycle counters
329                     STITLE  InitIRbaud


Saturn Assembler    LEXFILE -- Wed Mar 19 17:02:13 PST Wed Apr 26 09:15:06 1989
Ver. 1.54, 01/18/89 InitIRbaud                          pir.a          Page  10

330                 *
331                 *       Entry:  P=    0, Hex mode
332                 *
333                 *  Counts loops for 16th second.  The loop counter is 2 plus
334                 *  the number of loops actually taken.  Since each loop is 36 cycles,
335                 *  we should subtract 72 (2*36) cycles from the amount indicated by
336                 *  the loop counter.  Also, there are 53 cycles of overhead for the
337                 *  case where the loop is never made (?A=C B/GOYES CLKS10; C=DAT0 B;
338                 *  P=1; C=C-1 P; C=C-1 P; B=B+1 X; A=DAT0 B).
339                 *  Thus the formula for the total number of cycles in the 16th of a
340                 *  second is the number in the loop counter times the number of loops
341                 *  per cycle (36), subtract 72 for the loops that weren't really done
342                 *  and add 53 for the overhead.
343                 *
344                 *       L=loops in 16th second (62500uSec)
345                 *
346                 *       cycles/62500uSec = L * 36 - 19
347                 *
348                 *  We need to calculate the number of CPU cycles in a specified period of
349                 *  time.  From above, we can set up a ratio:
350                 *
351                 *        C       L * 36 - 19
352                 *       ---  =   -----------
353                 *        P          62500
354                 *
355                 *  where:
356                 *
357                 *       P=number of uSec per period
358                 *       C=number of cycles per period
359                 *
360                 *  Now, solving for C,
361                 *
362                 *             P * ( L * 36cyc - 19cyc )     L * ( P * 36cyc ) - P * 19cyc
363                 *       C  =  -------------------------  =  ----------------------------
364                 *                    62500uSec                        62500uSec
365                 *
366                 *  To time out a period, P, by controlling the number of CPU cycles we
367                 *  use a loop which has two properties,
368                 *
369                 *               c  =  number of cycles per loop, and
370                 *               o  =  number of cycles of overhead
371                 *
372                 *  The total loops (l) in a period, is calculated as follows:
373                 *
374                 *                     C - o
375                 *               l  =  -----
376                 *                       c
377                 *
378                 *  Substituting for C in this equation gives
379                 *
380                 *                L * ( P * 36cyc ) - P * 19cyc
381                 *                ----------------------------  -  o
382                 *                         62500uSec
383                 *       l  =  ------------------------------------
384                 *                              c
385                 *
386                 *
```

```
                *           L * ( P * 36cyc ) - ( ( P * 19cyc) + ( o * 62500uSec ))
                *       =   ---------------------------------------------------
                *                          62500uSec * c
                *
                *  Filling in some real life values from the case of the IR output
                *  routine:
                *
                *      P = 425.5 uSec/Period
                *      o = 209 cycles
                *      c = 13 cycles
                *
                *  allows the calculation to be reduced to:
                *
                *                      L * 15318 - 13070585
                *              l =    ----------------------
                *                          62500 * 13
                *
                *  This calculation should be carried out as follows to preserve
                *  as much accuracy as possible using integer arithmetic:
                *
                *      l = ( ( (L*15318)-13039335) div 62500 ) div 13
                *
                *  In order to get the result of the integer division by 62500 to
                *  round correctly, 62500/2 is added to the dividend.
                *
                *  Note that the remainder of the last integer division is the number
                *  of odd cycles that should be squandered to perform the delay down
                *  to the nearest cycle.
                *
00255           CLKSPD
00255 20        P=    0
00257 D1        B=0   A           Counter
00259 1B000     DO=(5) =TIMER2
      00
00260 808F      INTOFF
00264 D2        C=0   A
00266 10C       R4=C
00269 14A       A=DAT0 B          Read lower 2 nibs of timer
0026C 14E   CLKS10 C=DAT0 B       Wait until kerchunk
0026F 902       ?A=C  P           Kchunk?
00272 AF        GOYES CLKS10      Not yet--7 cycles if fall through
00274 14E       C=DAT0 B          In case kchunk in 1st digit--15 c
00277 21        P=    1           Scan for another kerchunk--2 cycs
00279 A0E       C=C-1 P           Time in 1/32 secs--4 cycles
0027C A0E       C=C-1 P           Time in 1/16 secs--4 cycles
0027F B35   CLKS20 B=B+1 X        Inc counter--6 cycles
00282 14A       A=DAT0 B          Read timer again--15 cycles
00285 966       ?A#C  B           Kerchunk again?--15 cycs if not
00288 7F        GOYES CLKS20      Fall thru 1/32 secs after ?a=c
0028A D0        A=0   A           Kill 7 cycles
0028C B35       B=B+1 X           Duplicate above loop--6 cycles
0028F 14A       A=DAT0 B          Read timer again--15 cycles
00292 966       ?A#C  B           False alarm?--15 if yes
00295 AE        GOYES CLKS20
00297 11C       C=R4
0029A 8AE       ?C#0  A
0029D 8B        GOYES CLKSPD
```



```
0029F 8080      INTON             B <= FFF
002A3 20        P=    0
002A5 D9        C=B   A
002A7 AF0       A=0   W
002AA D4        A=B   A
002AC AF2       C=0   W
002AF 336DB     LC(4) 15318
      3
002B5 8F000     GOSBVL =MPY
      00
002BC AF2       C=0   W
002BF 357E6     LC(6) 13039335
      F6C
002C7 B7A       A=A-C W
002CA AF2       C=0   W
002CD 33424     LC(4) 62500
      F
002D3 7620      GOSUB  idiv
```

```
457 002D7 20          P=    0
458 002D9 AF2         C=0   W
459 002DC 30D         LC(1) 13
460 002DF 7A10        GOSUB idiv
461 002E3 20          P=    0
462 002E5 F2          CSL   A
463 002E7 B88         A=-A  P          Complement Loop counter in A(0)
464 002EA C2          C=A+C A          Combine A(0) with C(1)
465           *Save loop and cycle counters
466 002EC 1B000       DO=(5) (=IS-PRT)+4
          00
467 002F3 14C         DAT0=C B
468 002F6 8D000       GOVLNG =NXTSTM
          00
469 002FD 8D000 idiv  GOVLNG =IDIV
          00
470
471           *
472           * End of LEXFILE
473           *
474 00304      FILEND
475 00304            END
```

```
MASK2          Abs   1254 #000004E6 -   124   144
MASK4          Abs    725 #000002D5 -   126   148
MASK8          Abs    395 #0000018B -   128   152
MPY            Ext                -   450
NXTSTM         Ext                -   468
=OUTBYTE       Rel    243 #000000F3 -   134
OUTELA         Ext                -   318
OutPulseBit    Abs     16 #00000010 -   204   230   277
PIR            Rel    580 #00000244 -   324    26
PIRd           Rel    561 #00000231 -   317   322
PIRp           Rel    568 #00000238 -   319   323
POLHND         Rel     91 #0000005B -    40    20
SendNextBit    Rel    341 #00000155 -   169   185
SendOne        Rel    365 #0000016D -   178   171
SendStBit      Rel    421 #000001A5 -   225   166   167   168
SendZero       Rel    349 #0000015D -   173
=SlowPrint     Rel    403 #00000193 -   191   193
StBitHiCyc     Abs    135 #00000087 -   222   223
StBitHiLps     Abs     10 #0000000A -   221   222   231
StBitLoopHi    Rel    440 #000001B8 -   234   236
StBitLoopLo    Rel    463 #000001CF -   245   247
```



```
BitDone        Rel    381 #0000017D -   183   177   182
BitHiCyc       Abs    117 #00000075 -   268   269
BitHiLps       Abs      8 #00000008 -   267   268   274   278
BitLoopHi      Rel    491 #000001EB -   281   275   283
BitLoopLo      Rel    518 #00000206 -   294   296
BitOvrhdHi     Abs     13 #0000000D -   260   267   268
BitOvrhdLo     Abs     50 #00000032 -   263   269
BitOvrhdPr     Abs     42 #0000002A -   257   269
BitTotal       Abs    209 #000000D1 -   269
CLKS10         Rel    620 #0000026C -   424   426
CLKS20         Rel    639 #0000027F -   431   434   439
CLKSPD         Rel    597 #00000255 -   416   442
CountBits      Rel    525 #0000020D -   298   141   145   149   153
CountBits00    Rel    531 #00000213 -   301   308
CountBits10    Rel    536 #00000218 -   304   306
CountBits20    Rel    548 #00000224 -   309   303
Cyc/BitHi      Abs    122 #0000007A -   256   267
Cyc/Lp         Abs     13 #0000000D -   206   221   221   222   267   267   268
Cyc/Sec        Abs 617000 #00096A28 -   205   211   256
Cyc/StBitHi    Abs    141 #0000008D -   211   221
Cycles41       Rel    413 #0000019D -   198   175   180
FILEND         Rel    772 #00000304 -   474    10
GetAddr        Rel    200 #000000C8 -    91    81
HAM10          Rel    266 #0000010A -   144   142
HAM20          Rel    280 #00000118 -   148   146
HAM30          Rel    294 #00000126 -   152   150
HAM40          Rel    308 #00000134 -   156   154
HPart3         Rel    198 #000000C6 -    89    93
HalfBitOne     Rel    478 #000001DE -   276   176   179
HalfBitZero    Rel    470 #000001D6 -   272   174   181
Handler        Rel    209 #000000D1 -    94
HandlerDone    Rel    198 #000000C6 -    88   112
HandlerLoop    Rel    228 #000000E4 -   109   194
IDIV           Ext                -   469
IS-PRT         Ext                -    73   105   325   466
MASK1          Abs   2168 #00000878 -   122   140
```



```
StBitOvrhdHi   Abs      5 #00000005 -   215   221   222
StBitOvrhdLo   Abs     32 #00000020 -   218   223
StBitOvrhdPr   Abs     42 #0000002A -   212   223
StBitTotal     Abs    209 #000000D1 -   223
TIMER2         Ext                -   419
TxTbEn         Rel     88 #00000058 -    35
TxTbSt         Rel     79 #0000004F -    30    18
VER$en         Rel    139 #00000088 -    57    46    57
VER$st         Rel    125 #0000007D -    56    46    57
fLEX           Ext                -     6
hPRTIS         Rel    156 #0000009C -    73    64
hPRTIS2        Rel    183 #000000B7 -    80    78
hPRTIS3        Rel    187 #000000BB -    82    92
hVER$0         Rel     99 #00000063 -    43    41
hVER$1         Rel    143 #0000008F -    58    49
hVER$2         Rel    145 #00000091 -    62    42
idiv           Rel    765 #000002FD -   469   456   460
pPRTIS         Ext                -    62
uSec/BitHi     Abs    198 #000000C6 -   251   256
uSec/HalfBit   Abs    425 #000001A9 -   203
uSec/StBitHi   Abs    229 #000000E5 -   207   211
=xrom5C        Rel     70 #00000046 -    23
```