

MACRO ASSEMBLER A51 V6.10

OBJECT MODULE PLACED IN .\SPEAKER.OBJ

ASSEMBLER INVOKED BY: C:\Keil\C51\BIN\A51.EXE .\SPEAKER.A51 REGISTERBANK(0) SET(SMALL) DEBUG EP

```

LOC  OBJ          LINE      SOURCE
      1          NAME      Speaker
      2          ;
      3          ; Copyright (C) 2001, Intel Corporation
      4          ; All rights reserved.
      5          ; Permission is hereby granted to merge this program code with other program
      6          ; material to create a derivative work. This derivative work may be distributed
      7          ; in compiled object form only. Any other publication of this program, in any form,
      8          ; without the explicit permission of the copyright holder is prohibited.
      9          ;
     10          ; Send questions and comments to John.Hyde@intel.com
     11          ;
     12          ;
     13          ; Derived from BAL Version 3.5
     14          ;
     15          ; This is a simple 8-bit mono speaker example (suitable for speech and sound effects).
     16          ; It is an Audio Class device with a single (16KHz) sampling frequency
     17          ; The hardware includes a USBSIMM, latching DAC and an op amp
     18          ;
     19          ; This version works dScope monSIO0.hex (uses Serial Port 0, loads at 1200H)
     20          ;
0040          21      EP0Size EQU      64      ; For EZ-USB
     22          ;
     23          ;$include (Declare.A51)
+1          24      ; This module declares the variables and constants used in the examples
+1          25      ; It is common to all of the examples
+1          26      ;
+1          27      ; Declare Special Function Registers used
0088          +1      28      TimerControl    DATA    088H
0089          +1      29      TimerMode      DATA    089H
008A          +1      30      Timer0Low     DATA    08AH
008C          +1      31      Timer0High    DATA    08CH
00A8          +1      32      EI              DATA    0A8H
00E8          +1      33      EIE          DATA    0E8H      ; EZ-USB specific
0091          +1      34      EXIF        DATA    091H      ; EZ-USB specific
00D8          +1      35      IICON       DATA    0D8H      ; EZ-USB specific
0092          +1      36      PageReg     DATA    092H      ; EZ-USB specific, used with MOVX @Ri
0086          +1      37      DPS          DATA    086H      ; EZ-USB specific, used with dual data pointers
+1          38      ;
+1          39      ; "External" memory locations used, EZ-USB specific
+1          40      ; Note that most of these variables are in Page 7FH
7FE1          +1      41      IsoOutValid  EQU      07FE1H
7FF0          +1      42      EP8OutStartAddr EQU    07FF0H
7F71          +1      43      EP8OutBCL   EQU    07F71H
7F60          +1      44      EP8OutData   EQU    07F60H
7FE8          +1      45      SETUPDAT    EQU    07FE8H
7FD4          +1      46      SUDPTR      EQU    07FD4H
7FB4          +1      47      EP0Control   EQU    07FB4H
7F00          +1      48      EP0InBuffer  EQU    07F00H
7EC0          +1      49      EP0OutBuffer EQU    07EC0H      ; Not in Page 7FH
7E80          +1      50      EP1InBuffer  EQU    07E80H      ; Not in Page 7FH
7FB5          +1      51      IN0ByteCount EQU    07FB5H
7FC5          +1      52      Out0ByteCount EQU    07FC5H
7FB7          +1      53      IN1ByteCount EQU    07FB7H
7FAC          +1      54      IN07IEN     EQU    07FACH
7FA9          +1      55      IN07IRQ     EQU    07FA9H
7FAD          +1      56      OUT07IEN    EQU    07FADH
7FAA          +1      57      OUT07IRQ     EQU    07FAAH
7FAE          +1      58      USBIEN      EQU    07FAEH

```

```
7FAB      +1  59  USBIRQ      EQU  07FABH
7FD6      +1  60  USBControl  EQU  07FD6H
7FA6      +1  61  I2CData    EQU  07FA6H
7FA5      +1  62  I2CControl EQU  07FA5H
7F93      +1  63  PortA_Config EQU  07F93H
7F94      +1  64  PortB_Config EQU  07F94H
7F95      +1  65  PortC_Config EQU  07F95H
7F96      +1  66  PortA_OUT   EQU  07F96H
7F97      +1  67  PortB_OUT   EQU  07F97H
7F98      +1  68  PortC_OUT   EQU  07F98H
7F99      +1  69  PortA_PINS  EQU  07F99H
7F9A      +1  70  PortB_PINS  EQU  07F9AH
7F9B      +1  71  PortC_PINS  EQU  07F9BH
7F9C      +1  72  PortA_OE   EQU  07F9CH
7F9D      +1  73  PortB_OE   EQU  07F9DH
7F9E      +1  74  PortC_OE   EQU  07F9EH
          +1  75  ;
          +1  76  ; Byte Variables
          +1  77
----      +1  78
0020      +1  79  FLAGS:      DS  1      ; This register is bit-addressable
          +1  80  ; Bit Variables
          0000 +1  81  Configured  EQU  FLAGS.0 ; Is this device configured
          0001 +1  82  STALL      EQU  FLAGS.1 ; Need to STALL endpoint 0
          0002 +1  83  SendData   EQU  FLAGS.2 ; Need to send data to PC Host
          0003 +1  84  IsDescriptor EQU  FLAGS.3 ; Enable a shortcut reply
          0004 +1  85  SetAddress  EQU  FLAGS.4 ; Set the SIE address
          +1  86  ;
0021      +1  87  MonitorSpace: DS  1FH  ; Used by Dscope
0040      +1  88  DataSpace:
0040      +1  89  ReplyCount:  DS  1      ; Byte count for following buffer
0041      +1  90  ReplyBuffer: DS  2      ; Buffer for immediate reply
0043      +1  91  CurrentConfiguration:
0043      +1  92  DS  1      ; Some examples support > 1 configurations
0044      +1  93  AltSetting:  DS  1      ; Speaker has two alternate settings (0, 1)
0045      +1  94  SaveDPH:    DS  1      ; Needed to save Descriptor Pointer ..
0046      +1  95  SaveDPL:    DS  1      ; .. for descriptors > EP0Size
0047      +1  96  SaveLength: DS  1      ; Number of bytes still to send
0048      +1  97  SetupData:  ; Buffer in direct access memory
0048      +1  98  RequestType: DS  1
0049      +1  99  Request:    DS  1
004A      +1 100  wValueLow:  DS  1
004B      +1 101  wValueHigh: DS  1
004C      +1 102  wIndexLow:  DS  1
004D      +1 103  wIndexHigh: DS  1
004E      +1 104  wLengthLow:  DS  1
004F      +1 105  wLengthHigh: DS  1
          +1 106  ;
0050      +1 107  INT0counter: DS  1
0051      +1 108  SOFcounter:  DS  1
          +1 109
0060      +1 110  ORG 60H      ; Put on a xxx00000B boundary
0060      +1 111  IsoDataBuffer: DS  16
          112
          113  ;$include (EZInt.A51)
          +1 114  ; This module contains all the EZUSB-specific hardware code
          +1 115  ; This module also contains all of the interrupt vector declarations and
          +1 116  ; the first level interrupt servicing (register save, call subroutine,
          +1 117  ; clear interrupt source, restore registers, return)
          +1 118  ; Suspend and Resume are handled totally in this module
          +1 119  ;
          +1 120  ; A Reset sends us to Program space location 0
----      +1 121  CSEG AT 0      ; Code space
          +1 122  USING 0      ; Reset forces Register Bank 0
0000 0212E2 +1 123  LJMP  Reset
          +1 124  ;
```

```
+1 125 ; The interrupt vector table is also located here
+1 126 ; EZ-USB has two levels of USB interrupts:
+1 127 ; 1-the main level is described in this table (at ORG 43H)
+1 128 ; 2-there are 21 sources of USB interrupts and these are described in USB_ISR
+1 129 ; This means that two levels of acknowledgement and clearing will be required
000B +1 130 ; LJMP INT0_ISR ; Features not used are commented out
000B 02121C +1 131 ORG 0BH
+1 132 ; LJMP Timer0_ISR
+1 133 ; ORG 13H
+1 134 ; LJMP INT1_ISR
+1 135 ; ORG 1BH
+1 136 ; LJMP Timer1_ISR
+1 137 ; ORG 23H
+1 138 ; LJMP UART0_ISR
+1 139 ; ORG 2BH
+1 140 ; LJMP Timer2_ISR
+1 141 ; ORG 33H
+1 142 ; LJMP WakeUp_ISR
+1 143 ; ORG 3BH
0043 +1 144 ; LJMP UART1_ISR
0043 021200 +1 145 ORG 43H
+1 146 ; LJMP USB_ISR ; Auto Vector will replace byte 45H
+1 147 ; ORG 4BH
+1 148 ; LJMP I2C_ISR
+1 149
1200 +1 150 ORG 1200H ; Load above monSIO0.hex
1200 02126A +1 151 USB_ISR:LJMP SUDAV_ISR
1203 00 +1 152 DB 0 ; Pad entries to 4 bytes
1204 02124D +1 153 LJMP SOF_ISR
1207 00 +1 154 DB 0
1208 02121B +1 155 LJMP SUTOK_ISR
120B 00 +1 156 DB 0
120C 02121B +1 157 LJMP Suspend_ISR
120F 00 +1 158 DB 0
1210 02122F +1 159 LJMP USBReset_ISR
1213 00 +1 160 DB 0
1214 02121B +1 161 LJMP Reserved
1217 00 +1 162 DB 0
1218 02123C +1 163 LJMP EP0In_ISR
+1 164 ; DB 0 ; Comment out features not used
+1 165 ; LJMP EP0Out_ISR
+1 166 ; DB 0
+1 167 ; LJMP EP1In_ISR
+1 168 ; DB 0
+1 169 ; LJMP EP1Out_ISR
+1 170 ; DB 0
+1 171 ; End of Interrupt Vector tables
+1 172
+1 173 ; When a feature is used insert the required interrupt processing here
+1 174 ; The example use only used Endpoints 0 and 1 and also SOF for timing
121B +1 175 Reserved:
121B +1 176 INT0_ISR:
121B +1 177 INT1_ISR:
121B +1 178 Timer1_ISR:
121B +1 179 UART0_ISR:
121B +1 180 Timer2_ISR:
121B +1 181 UART1_ISR:
121B +1 182 I2C_ISR:
121B +1 183 SUTOK_ISR:
121B +1 184 EP0Out_ISR:
121B +1 185 EP1In_ISR:
121B +1 186 EP1Out_ISR:
121B +1 187 Suspend_ISR:
121B +1 188 WakeUp_ISR:
121B +1 189 Not_Used: ; Should not get any of these
121B 32 +1 190 RETI
```

```
+1 191
121C +1 192 Timer0_ISR: ; Timer0 has just overflowed.
+1 193 ; This routine must be as fast as possible. Note that it changes no flags
121C D2D3 +1 194 SETB RSO ; Use alternate register bank RB1
121E C6 +1 195 XCH A, @R0 ; Save A and GetNextValue.
121F F3 +1 196 MOVX @R1, A ; Send next sample to DAC
1220 C6 +1 197 XCH A, @R0 ; Restore A
1221 08 +1 198 INC R0 ; Point to next entry
1222 53080F +1 199 ANL 8, #00001111B ; Memory 8 = R0 in RB1. Maintain a circular buffer
1225 C2D3 +1 200 CLR RSO
1227 32 +1 201 RETI ; Interrupt flag is cleared by the hardware
+1 202
1228 +1 203 ClearINT2: ; Tell the hardware that we're done
1228 E591 +1 204 MOV A, EXIF
122A C2E4 +1 205 CLR ACC.4 ; Clear the Interrupt 2 bit
122C F591 +1 206 MOV EXIF, A
122E 22 +1 207 RET
+1 208
122F +1 209 USBReset_ISR: ; Bus has been Reset, move to DEFAULT state
122F C200 +1 210 CLR Configured
1231 9010AB +1 211 MOV DPTR, #(1000H OR LOW(USBIRQ))
1234 +1 212 ExitISR: ; Common exit for all ISR's
+1 213 ; On entry DPH = Interrupt ID, DPL = LOW(Interrupt Register)
1234 5128 +1 214 CALL ClearINT2
1236 747F +1 215 MOV A, #7FH ; EZ-USB I/O Register Page
1238 C583 +1 216 XCH A, DPH
123A F0 +1 217 MOVX @DPTR, A ; Clear source of interrupt
123B 32 +1 218 RETI
+1 219
123C +1 220 EP0In_ISR: ; A prepared packet has been read by PC host
123C E547 +1 221 MOV A, SaveLength ; Do I have any more data to send?
123E 6008 +1 222 JZ NoMoreToSend
1240 854583 +1 223 MOV DPH, SaveDPH ; Retrieve descriptor pointer
1243 854682 +1 224 MOV DPL, SaveDPL
1246 51B7 +1 225 CALL SendNextPieceOfDescriptor
1248 +1 226 NoMoreToSend:
1248 9001A9 +1 227 MOV DPTR, #(100H OR LOW(IN07IRQ))
124B 80E7 +1 228 JMP ExitISR
+1 229
124D +1 230 SOF_ISR: ; A Start-Of-Frame packet has been received
124D 7F10 +1 231 MOV R7, #16 ; We need 16 samples every frame
124F 7960 +1 232 MOV R1, #IsoDataBuffer
1251 7871 +1 233 MOV R0, #LOW(EP8outBCL) ; Point to (low) of bytes received
1253 E2 +1 234 MOVX A, @R0 ; Did we get a sample in the last frame?
1254 6009 +1 235 JZ RepeatLastSample
1256 +1 236 GetNewSamples:
1256 7860 +1 237 MOV R0, #LOW(EP8OutData)
1258 E2 +1 238 GNSLoop:MOVX A, @R0 ; Copy ISO data to direct access buffer
1259 F7 +1 239 MOV @R1, A
125A 09 +1 240 INC R1
125B DFFB +1 241 DJNZ R7, GNSLoop
125D 8006 +1 242 SJMP Done
125F +1 243 RepeatLastSample:
125F E56F +1 244 MOV A, IsoDataBuffer+15 ; Get last sample
1261 F7 +1 245 RLSLoop:MOV @R1, A
1262 09 +1 246 INC R1
1263 DFFC +1 247 DJNZ R7, RLSLoop
+1 248 ;
1265 9002AB +1 249 Done: MOV DPTR, #(200H OR LOW(USBIRQ))
1268 80CA +1 250 JMP ExitISR
+1 251
126A +1 252 SUDAV_ISR: ; A Setup packet has been received
126A 754700 +1 253 MOV SaveLength, #0 ; Clear any pending transactions (if any)
126D 907FE8 +1 254 MOV DPTR, #SETUPDAT ; Copy packet to direct access memory
1270 7848 +1 255 MOV R0, #SetupData
1272 7F08 +1 256 MOV R7, #8
```

```
1274 E0      +1  257      CopySD: MOVX    A, @DPTR
1275 F6      +1  258              MOV     @R0, A
1276 A3      +1  259              INC     DPTR
1277 08      +1  260              INC     R0
1278 DFFA     +1  261              DJNZ   R7, CopySD
127A 716A    +1  262              CALL   ServiceSetupPacket      ; Handle the decode of the Setup packet
+1  263      ; if SetAddress { Update SIE address } // NOP on EZ-USB
+1  264      ; if STALL { Stall the endpoint }
+1  265      ; if SendData {
+1  266      ;     if IsDescriptor { send DPTR->descriptor, A = length }
+1  267      ;     else { send ReplyBuffer }
+1  268      ;     }
127C 200126  +1  269              JB     STALL, SendSTALL
127F 300216  +1  270              JNB   SendData, HandShake
1282 200324  +1  271              JB     IsDescriptor, LoadEP0
+1  272      ; Send data in ReplyBuffer
1285 907F01  +1  273              MOV     DPTR, #EP0InBuffer+1
1288 7842    +1  274              MOV     R0, #ReplyBuffer+1
128A 7F02    +1  275              MOV     R7, #2                ; Copy the two byte buffer
128C E6      +1  276      CopyRB: MOV     A, @R0
128D F0      +1  277              MOVX   @DPTR, A
128E 1582    +1  278              DEC     DPL
1290 18      +1  279              DEC     R0
1291 DFF9    +1  280              DJNZ   R7, CopyRB
1293 E6      +1  281              MOV     A, @R0                ; Get BufferCount
1294         +1  282      SendEP0InBuffer:
1294 907FB5   +1  283              MOV     DPTR, #In0ByteCount
1297         +1  284      StartXfer:
1297 F0      +1  285              MOVX   @DPTR, A                ; This write initiates the transfer
1298         +1  286      HandShake:
1298 7F02    +1  287              MOV     R7, #00000010b       ; Handshake with host
129A         +1  288      SetEP0Control:
129A 907FB4   +1  289              MOV     DPTR, #EP0Control
129D E0      +1  290              MOVX   A, @DPTR
129E 4F      +1  291              ORL    A, R7
129F F0      +1  292              MOVX   @DPTR, A                ; We're done
12A0 9001AB  +1  293              MOV     DPTR, #(100H OR LOW(USBIRQ))
12A3 808F    +1  294              JMP     ExitISR
12A5         +1  295      SendSTALL:
12A5 7F03    +1  296              MOV     R7, #00000011b       ; Invalid Request was received
12A7 80F1    +1  297              JMP     SetEP0Control        ; Set EPOSTALL and HSNACK
12A9         +1  298      LoadEP0:
12A9 FF      +1  299              MOV     R7, A                ; Send the data pointed to by DPTR
+1  300      ; Need to return the smaller of "Requested Length" and "Actual Length"
+1  301      ; If "Requested Length" > 255 then use "Actual Length"
+1  302      ; There are no descriptors > 255 in this example
12AA E54F    +1  303              MOV     A, wLengthHigh
12AC 7008    +1  304              JNZ    UseActual
12AE EF      +1  305              MOV     A, R7                ; Retrieve LENGTH
12AF C3      +1  306              CLR     C
12B0 954E    +1  307              SUBB   A, wLengthLow
12B2 E54E    +1  308              MOV     A, wLengthLow        ; This does not affect Carry
12B4 5001    +1  309              JNC    UsewLengthLow
12B6         +1  310      UseActual:
12B6 EF      +1  311              MOV     A, R7
12B7         +1  312      UsewLengthLow:
12B7 FF      +1  313      SendNextPieceOfDescriptor:
12B7 FF      +1  314              MOV     R7, A                ; DPTR -> Descriptor to be sent
12B8 754700  +1  315              MOV     SaveLength, #0       ; Save LENGTH again
+1  316      ; Default case, overwrite if necessary
+1  317      ; Do I have more than a single packet to send?
12BB C3      +1  317              CLR     C
12BC 9440    +1  318              SUBB   A, #EP0Size
12BE 4015    +1  319              JC     SendPacket
+1  320      ; Need to send multiple packets.
+1  321      ; Calculate and save address of next packet, send next packet now
12C0 F547    +1  322              MOV     SaveLength, A        ; Send these next time
```

```
12C2 7F40      +1  323      MOV      R7, #EP0Size
12C4 C083      +1  324      PUSH     DPH                ; Save current pointer
12C6 C082      +1  325      PUSH     DPL
12C8 EF        +1  326      MOV      A, R7                ; Retrieve length
12C9 71AD      +1  327      CALL    BumpDPTR
12CB 858345    +1  328      MOV      SaveDPH, DPH
12CE 858246    +1  329      MOV      SaveDPL, DPL
12D1 D082      +1  330      POP      DPL
12D3 D083      +1  331      POP      DPH
12D5           +1  332      SendPacket:
12D5 EF        +1  333      MOV      A, R7                ; Retrieve length
12D6 FE        +1  334      MOV      R6, A                ; Save length in R6 for move
12D7 7800      +1  335      MOV      R0, #LOW(EP0InBuffer) ; PageReg = 7FH = HIGH(EP0InBuffer)
12D9 E0        +1  336      CopySTD:MOVX   A, @DPTR
12DA F2        +1  337      MOVX    @R0, A
12DB A3        +1  338      INC     DPTR
12DC 08        +1  339      INC     R0
12DD DEFA     +1  340      DJNZ   R6, CopySTD
12DF EF        +1  341      MOV      A, R7                ; Retrieve LENGTH
12E0 80B2     +1  342      JMP     SendEP0InBuffer
+1  343
+1  344
+1  345
+1  346
+1  347
+1  348      ;$include (EZMain.A51)
+1  349      ; This module initializes the microcontroller then executes MAIN forever
+1  350      ; It is hardware dependant
+1  351
12E2           +1  352      Reset:
12E2 7581DF    +1  353      MOV      SP, #0DFH            ; Initialize the Stack
12E5 75927F    +1  354      MOV      PageReg, #7FH        ; Allows MOVX Ri to access EZ-USB memory
+1  355
12E8 78D6     +1  356      MOV      R0, #Low(USBControl) ; Simulate a disconnect
12EA E2       +1  357      MOVX    A, @R0
12EB 54F3     +1  358      ANL     A, #11110011b        ; Clear DISCON, DISCOE
12ED F2       +1  359      MOVX    @R0, A
12EE 7158     +1  360      CALL    Wait200msec          ; Give the host time to react
12F0 E2       +1  361      MOVX    A, @R0                ; Reconnect with this new identity
12F1 4406     +1  362      ORL     A, #00000110b        ; Set DISCOE to enable pullup resistor
12F3 F2       +1  363      MOVX    @R0, A                ; Set RENUM so that 8051 handles USB requests
12F4 E4       +1  364      CLR     A
12F5 F520     +1  365      MOV      FLAGS, A            ; Start in Default state
+1  366
12F7           +1  367      InitializeIOSystem:          ; Setup for Simmbus A and B=output
+1  368      ; C=External RD#,WR#,TD0,TR0
12F7 7893     +1  369      MOV      R0, #LOW(PortA_Config) ; PageReg = 7F = HIGH(PortA_Config)
12F9 799C     +1  370      MOV      R1, #LOW(PortA_OE)
12FB 04       +1  371      INC     A                      ; = 00000001B
12FC F2       +1  372      MOVX    @R0, A                ; Alternate function on bit 0 = Timer0 Out
12FD F3       +1  373      MOVX    @R1, A                ; Enable PortA for Input[7:1], Output[0]
12FE 08       +1  374      INC     R0                      ; Point to PortB_Config
12FF 09       +1  375      INC     R1                      ; Point to PortB_OE
1300 14       +1  376      DEC     A                      ; = 0
1301 F2       +1  377      MOVX    @R0, A                ; No alternate functions
1302 F4       +1  378      CPL     A                      ; = 0FFH
1303 F3       +1  379      MOVX    @R1, A                ; Enable PortB for Output
1304 08       +1  380      INC     R0                      ; Point to PortC_Config
1305 09       +1  381      INC     R1                      ; Point to PortC_OE
1306 74C3     +1  382      MOV      A, #11000011b        ;
1308 F2       +1  383      MOVX    @R0, A                ; Alternate functions on [7,6,1,0]
1309 74C2     +1  384      MOV      A, #11000010b        ;
130B F3       +1  385      MOVX    @R1, A                ; Most alternate functions are outputs
+1  386      ; Use Timer 0 since dScope is using 1 (or 2)
130C E589     +1  387      MOV      A, TimerMode
130E 54F0     +1  388      ANL     A, #11110000B        ; Program Timer0 without changing Timer1
```

```
1310 4402      +1  389          ORL    A, #00000010B
1312 F589      +1  390          MOV    TimerMode, A          ; Timer0 = 8 bit auto reload
1314 758C80    +1  391          MOV    Timer0High, #128      ; 128 x 2MHz = 64usec
1317 758A9B    +1  392          MOV    Timer0Low, #(255-100) ; First interrupt will occur 50usec after
starting
131A          +1  393          InitializeEndpoints:
131A 78E1      +1  394          MOV    R0, #LOW(IsoOutValid) ; EZ-USB uses Endpoints 8-15 for iso traffic
131C 7401      +1  395          MOV    A, #00000001B       ; We'll use EP8, double buffered
131E F2        +1  396          MOVX   @R0, A               ; Enable EP8 OUT
131F 18        +1  397          DEC    R0                  ; Point to IsoInValid
1320 E4        +1  398          CLR    A
1321 F2        +1  399          MOVX   @R0, A               ; Disable EP8-15 In
1322 78F0      +1  400          MOV    R0, #LOW(EP8OutStartAddr)
1324 F2        +1  401          MOVX   @R0, A               ; Use the beginning of the ISO buffer
1325          +1  402          InitializeInterruptSystem: ; First initialize the USB level
1325 7401      +1  403          MOV    A, #00000001b
1327 78AC      +1  404          MOV    R0, #LOW(IN07IEN)
1329 F2        +1  405          MOVX   @R0, A               ; Enable interrupts from EP0IN only
132A 08        +1  406          INC    R0
132B E4        +1  407          CLR    A
132C F2        +1  408          MOVX   @R0, A               ; Disable interrupts from OUT Endpoints 0-7
132D 08        +1  409          INC    R0
132E 7413      +1  410          MOV    A, #00010011b
132E          +1  411          ; Enable USBReset, (Resume, Suspend,) SOF and SUDAV INTs
1330 F2        +1  412          MOVX   @R0, A
1331 08        +1  413          INC    R0
1332 7401      +1  414          MOV    A, #00000001b
1334 F2        +1  415          MOVX   @R0, A               ; Enable Auto Vectoring for USB interrupts
1335          +1  416          InitializeInterruptVariables:
1335 D2D3      +1  417          SETB   RS0                  ; Select RB1 (used to service INT0)
1337 7870      +1  418          MOV    R0, #IsoDataBuffer+16
1339 7997      +1  419          MOV    R1, #LOW(PortB_Out)
133B 7F10      +1  420          MOV    R7, #16
133D E4        +1  421          CLR    A
133E 18        +1  422          IIVLoop:DEC R0
133F F6        +1  423          MOV    @R0, A
1340 DFFC      +1  424          DJNZ   R7, IIVLoop
1342 C2D3      +1  425          CLR    RS0                  ; Go back to RB0 for MAIN
1342          +1  426          ; Synchronize start of Timer0 with SOF
1344 78AB      +1  427          MOV    R0, #LOW(USBIRQ)
1346 7402      +1  428          MOV    A, #00000010B
1348 F2        +1  429          MOVX   @R0, A               ; Clear pending SOF
1349          +1  430          WaitForNextSOF:
1349 E2        +1  431          MOVX   A, @R0
134A 30E1FC    +1  432          JNB   ACC.1, WaitForNextSOF
134D D28C      +1  433          SETB   TimerControl.4      ; Start Timer0
134D          +1  434          ; Now enable the main level
134F 75E801    +1  435          MOV    EIE, #00000001B     ; Enable INT2 = USB Interrupt (only)
1352 75A892    +1  436          MOV    EI, #10010010B     ; Enable interrupt subsystem
1352          +1  437          ; and Ser0 for dScope and Timer0
1352          +1  438
1352          +1  439          ; Initialization Complete.
1352          +1  440          ;
1355          +1  441          MAIN:
1355 00        +1  442          NOP
1356 80FD      +1  443          JMP    MAIN                  ; All actions are initiated by interrupts
1356          +1  444          ; We are a slave, we wait to be told what to do
1356          +1  445
1358          +1  446          Wait200msec:
1358 7FC8      +1  447          MOV    R7, #200
135A          +1  448          Wait1msec:
135A 758600    +1  449          MOV    DPS, #0              ; A delay loop
135D 90FB50    +1  450          MOV    DPTR, #-1200         ; Select primary DPTR
1360 A3        +1  451          More: INC DPTR              ; 3 cycles
1361 E582      +1  452          MOV    A, DPL               ; + 2
1363 4583      +1  453          ORL   A, DPH                ; + 2
1365 70F9      +1  454          JNZ   More                  ; + 3 = 10 cycles x 1200 = 1msec
```

```
1367 DFF1      +1 455          DJNZ   R7, Waitlmsec
1369 22        +1 456          RET
              +1 457
              +1 458
              459
              460          ;$include (Decode.A51)
              +1 461          ; This module is common to all of the examples.
              +1 462          ; It decodes the USB Setup Packets and generates appropriate responses.
              +1 463          ; Interpretation of Reports is handled by MAIN
              +1 464          ;
----         +1 465          CSEG
136A          +1 466          ServiceSetupPacket:
136A E548      +1 467          MOV    A, RequestType
136C A2E7      +1 468          MOV    C, ACC.7          ; Bit 7 = 1 means IO device needs to send data
to P
              C Host
136E 9202      +1 469          MOV    SendData, C
1370 545C      +1 470          ANL    A, #01011100b      ; IF RequestType[6.4.3.2] = 1 THEN goto
BadRequest
1372 7035      +1 471          JNZ    BadRequest
1374 E548      +1 472          MOV    A, RequestType      ; IF RequestType[1&0] = 1 THEN goto BadRequest
1376 A2E0      +1 473          MOV    C, ACC.0
1378 82E1      +1 474          ANL    C, ACC.1
137A 402D      +1 475          JC    BadRequest
137C 30E502    +1 476          JNB   ACC.5, NotB5      ; IF RequestType[5] = 1 THEN RequestType[1,0] =
[1,
              1]
137F 7403      +1 477          MOV    A, #00000011b
1381 5403      +1 478          NotB5: ANL    A, #00000011b      ; Set CommandIndex[5,4] = RequestType[1,0]
1383 C4        +1 479          SWAP   A
1384 FF        +1 480          MOV    R7, A          ; Save HI nibble of CommandIndex
              +1 481          ; Set CommandIndex[3,0] = Request[3,0]
1385 E549      +1 482          MOV    A, Request
1387 54F0      +1 483          ANL    A, #11110000b      ; Check if Request > 15
1389 701E      +1 484          JNZ    BadRequest
138B E549      +1 485          MOV    A, Request
138D 540F      +1 486          ANL    A, #00001111b      ; Only 13 are defined today, handle in table
138F 4F        +1 487          ORL    A, R7
              +1 488          ; CALL  CorrectSubroutine      ; goto CommandTable(CommandIndex)
1390          +1 489          CorrectSubroutine:      ; Jump to the subroutine that DPTR is pointing
to
1390 754001    +1 490          MOV    ReplyCount, #1      ; Set up a default reply
1393 754100    +1 491          MOV    ReplyBuffer, #0
1396 754200    +1 492          MOV    ReplyBuffer+1, #0
1399 C204      +1 493          CLR    SetAddress      ; Clear all flags
139B C201      +1 494          CLR    STALL
139D C203      +1 495          CLR    IsDescriptor
139F 9013B6    +1 496          MOV    DPTR, #CommandTable
13A2 71AD      +1 497          CALL  BumpDPTR          ; Point to entry
13A4 E0        +1 498          MOVX   A, @DPTR        ; Get the offset
13A5 9013F6    +1 499          MOV    DPTR, #Subroutines
13A8 73        +1 500          JMP    @A+DPTR          ; Go to the correct Subroutine
              +1 501
13A9          +1 502          BadRequest:          ; Decoded a Bad Request, STALL the Endpoint
13A9 D201      +1 503          SETB   STALL
13AB 22        +1 504          RET
              +1 505          ; Support routines
13AC          +1 506          NextDPTR:          ; Returns (DPTR + byte DPTR is pointing to)
13AC E0        +1 507          MOVX   A, @DPTR
13AD          +1 508          BumpDPTR:          ; Returns (DPTR + ACC)
13AD 2582      +1 509          ADD    A, DPL
13AF F582      +1 510          MOV    DPL, A
13B1 5002      +1 511          JNC    Skip
13B3 0583      +1 512          INC    DPH          ; Need 16 bit arithmetic here
13B5 22        +1 513          Skip:  RET
              +1 514
              +1 515          ; Since the table only contains byte offsets, it is important that all these routines
are
              +1 516          ; within one page (100H) of Subroutines
              +1 517          ; V3.0 - CommandTable moved outside of this one page limited space
13B6          +1 518          CommandTable:
```

```
+1 519 ; First 16 commands are for the Device
13B6 0A +1 520 DB LOW(Device_Get_Status - Subroutines)
13B7 00 +1 521 DB LOW(Device_Clear_Feature - Subroutines)
13B8 00 +1 522 DB LOW(Invalid - Subroutines)
13B9 00 +1 523 DB LOW(Device_Set_Feature - Subroutines)
13BA 00 +1 524 DB LOW(Invalid - Subroutines)
13BB 03 +1 525 DB LOW(Set_Address - Subroutines)
13BC 39 +1 526 DB LOW(Get_Descriptor - Subroutines)
13BD 00 +1 527 DB LOW(Set_Descriptor - Subroutines)
13BE 06 +1 528 DB LOW(Get_Configuration - Subroutines)
13BF 12 +1 529 DB LOW(Set_Configuration - Subroutines)
13C0 00 +1 530 DB LOW(Invalid - Subroutines)
13C1 00 +1 531 DB LOW(Invalid - Subroutines)
13C2 00 +1 532 DB LOW(Invalid - Subroutines)
13C3 00 +1 533 DB LOW(Invalid - Subroutines)
13C4 00 +1 534 DB LOW(Invalid - Subroutines)
13C5 00 +1 535 DB LOW(Invalid - Subroutines)
+1 536 ; Next 16 commands are for the Interface
13C6 0E +1 537 DB LOW(Interface_Get_Status - Subroutines)
13C7 00 +1 538 DB LOW(Interface_Clear_Feature - Subroutines)
13C8 00 +1 539 DB LOW(Invalid - Subroutines)
13C9 00 +1 540 DB LOW(Interface_Set_Feature - Subroutines)
13CA 00 +1 541 DB LOW(Invalid - Subroutines)
13CB 00 +1 542 DB LOW(Invalid - Subroutines)
13CC 00 +1 543 DB LOW(Get_Class_Descriptor - Subroutines)
13CD 00 +1 544 DB LOW(Set_Class_Descriptor - Subroutines)
13CE 00 +1 545 DB LOW(Invalid - Subroutines)
13CF 00 +1 546 DB LOW(Invalid - Subroutines)
13D0 30 +1 547 DB LOW(Get_Interface - Subroutines)
13D1 20 +1 548 DB LOW(Set_Interface - Subroutines)
13D2 00 +1 549 DB LOW(Invalid - Subroutines)
13D3 00 +1 550 DB LOW(Invalid - Subroutines)
13D4 00 +1 551 DB LOW(Invalid - Subroutines)
13D5 00 +1 552 DB LOW(Invalid - Subroutines)
+1 553 ; Next 16 commands are for the Endpoint
13D6 0E +1 554 DB LOW(Endpoint_Get_Status - Subroutines)
13D7 00 +1 555 DB LOW(Endpoint_Clear_Feature - Subroutines)
13D8 00 +1 556 DB LOW(Invalid - Subroutines)
13D9 00 +1 557 DB LOW(Endpoint_Set_Feature - Subroutines)
13DA 00 +1 558 DB LOW(Invalid - Subroutines)
13DB 00 +1 559 DB LOW(Invalid - Subroutines)
13DC 00 +1 560 DB LOW(Invalid - Subroutines)
13DD 00 +1 561 DB LOW(Invalid - Subroutines)
13DE 00 +1 562 DB LOW(Invalid - Subroutines)
13DF 00 +1 563 DB LOW(Invalid - Subroutines)
13E0 00 +1 564 DB LOW(Invalid - Subroutines)
13E1 00 +1 565 DB LOW(Invalid - Subroutines)
13E2 00 +1 566 DB LOW(Endpoint_Sync_Frame - Subroutines)
13E3 00 +1 567 DB LOW(Invalid - Subroutines)
13E4 00 +1 568 DB LOW(Invalid - Subroutines)
13E5 00 +1 569 DB LOW(Invalid - Subroutines)
+1 570 ; Next 16 commands are Class Requests
13E6 00 +1 571 DB LOW(Invalid - Subroutines)
13E7 00 +1 572 DB LOW(Invalid - Subroutines)
13E8 00 +1 573 DB LOW(Invalid - Subroutines)
13E9 00 +1 574 DB LOW(Invalid - Subroutines)
13EA 00 +1 575 DB LOW(Invalid - Subroutines)
13EB 00 +1 576 DB LOW(Invalid - Subroutines)
13EC 00 +1 577 DB LOW(Invalid - Subroutines)
13ED 00 +1 578 DB LOW(Invalid - Subroutines)
13EE 00 +1 579 DB LOW(Invalid - Subroutines)
13EF 00 +1 580 DB LOW(Invalid - Subroutines)
13F0 00 +1 581 DB LOW(Invalid - Subroutines)
13F1 00 +1 582 DB LOW(Invalid - Subroutines)
13F2 00 +1 583 DB LOW(Invalid - Subroutines)
13F3 00 +1 584 DB LOW(Invalid - Subroutines)
```

```
13F4 00      +1  585          DB LOW(Invalid - Subroutines)
13F5 00      +1  586          DB LOW(Invalid - Subroutines)
              +1  587
13F6         +1  588      Subroutines:
              +1  589      ;
              +1  590      ; Many requests are INVALID for this example
13F6         +1  591      Set_Descriptor:           ; Our Descriptors are static
13F6         +1  592      Set_Class_Descriptor:        ; Our Descriptors are static
13F6         +1  593      Get_Class_Descriptor:       ; We have no features that the PC host can read
13F6         +1  594      Set_Idle:                 ; V3.0 Optional command, not supported
13F6         +1  595      Get_Idle:                 ; V3.0 Optional command, not supported
13F6         +1  596      Device_Set_Feature:        ; We have no features that can be set or cleared
13F6         +1  597      Interface_Set_Feature:     ; We have no features that can be set or cleared
13F6         +1  598      Endpoint_Set_Feature:     ; We have no features that can be set or cleared
13F6         +1  599      Endpoint_Clear_Feature:   ; V3.0 We have no features that can be set or cleared
13F6         +1  600      Device_Clear_Feature:     ; We have no features that can be set or cleared
13F6         +1  601      Interface_Clear_Feature:   ; We have no features that can be set or cleared
13F6         +1  602      Endpoint_Sync_Frame:      ; We are not an Isonchronous device
              +1  603
13F6         +1  604      Invalid:                   ; Invalid Request made, STALL the Endpoint
13F6 D201    +1  605          SETB      STALL
13F8 22      +1  606      Reply:  RET
              +1  607
13F9         +1  608      Set_Address:           ; Set the address that the SIE will respond to
13F9 D204    +1  609          SETB      SetAddress
13FB 22      +1  610          RET
              +1  611
13FC         +1  612      Get_Configuration:         ; Respond with CurrentConfiguration
13FC 854341  +1  613          MOV      ReplyBuffer, CurrentConfiguration
13FF 22      +1  614          RET
1400         +1  615      Device_Get_Status:        ; Only two bits of Device Status are defined
1400 754101  +1  616          MOV      ReplyBuffer, #1      ; Bit 1=Remote Wakeup(=0), Bit 0=Self
Powered(=1)
1403 22      +1  617          RET
1404         +1  618      Interface_Get_Status:       ; Interface Status is currently defined as 0
1404         +1  619      Endpoint_Get_Status:
1404 754002  +1  620          MOV      ReplyCount, #2      ; Need a two byte 0 response
1407 22      +1  621          RET
1408         +1  622      Set_Configuration:        ; Valid values are 0 and 1
1408 E54A    +1  623          MOV      A, wValueLow
140A 6004    +1  624          JZ      Deconfigured
140C 14      +1  625          DEC      A
140D 70E7    +1  626          JNZ     Invalid
140F 04      +1  627          INC      A                ; Restore wValueLow
1410         +1  628      Deconfigured:
1410 F543    +1  629          MOV      CurrentConfiguration, A
1412 13      +1  630          RRC      A                ; Copy bit 0 into C
1413 9200    +1  631          MOV      Configured, C
1415 22      +1  632          RET
1416         +1  633      Set_Interface:           ; Set Interface 1 Alternate setting
1416 E54C    +1  634          MOV      A, wIndexLow
1418 14      +1  635          DEC      A                ; Check PC Host is using interface #1
1419 70DB    +1  636          JNZ     Invalid
141B E54A    +1  637          MOV      A, wValueLow       ; Valid values are 0 and 1
141D 6003    +1  638          JZ      OKtoSet
141F 14      +1  639          DEC      A
1420 70D4    +1  640          JNZ     Invalid
1422 854A44  +1  641      OKtoSet:MOV      AltSetting, wValueLow
1425 22      +1  642          RET
1426         +1  643      Get_Interface:         ; Return Interface 1 Alternate setting
1426 E54C    +1  644          MOV      A, wIndexLow
1428 14      +1  645          DEC      A                ; Check PC Host is using interface #1
1429 70CB    +1  646          JNZ     Invalid
142B 854441  +1  647          MOV      ReplyBuffer, AltSetting
142E 22      +1  648          RET
142F         +1  649      Get_Descriptor:        ; Host wants to know who/what we are
142F D203    +1  650          SETB      IsDescriptor
```

```
1431 E54B      +1 651          MOV     A, wValueHigh
1433 14        +1 652          DEC     A                ; Valid Values are 1, 2 and 3
1434 901458    +1 653          MOV     DPTR, #DeviceDescriptor
1437 601D      +1 654          JZ     ReturnLength
1439 14        +1 655          DEC     A
143A 90146A    +1 656          MOV     DPTR, #ConfigurationDescriptor
143D 7003      +1 657          JNZ    TryString
143F 7464      +1 658          MOV     A, #ConfigLength
1441 22        +1 659          RET
1442          +1 660          TryString:
1442 14        +1 661          DEC     A
1443 70B1      +1 662          JNZ    Invalid
+1 663          ; Request is for a String Descriptor
1445 9014CE    +1 664          MOV     DPTR, #String0    ; Point to String 0
1448 E54A      +1 665          MOV     A, wValueLow     ; Get String Index
144A          +1 666          NextString:
144A 600A      +1 667          JZ     ReturnLength
144C FF        +1 668          MOV     R7, A            ; Save String Index
144D 71AC      +1 669          CALL   NextDPTR
144F E0        +1 670          MOVX   A, @DPTR         ; Get the String Length (= 0 means we're at
Backsto
p)
1450 60A4      +1 671          JZ     Invalid          ; Asked for a string I don't have
1452 EF        +1 672          MOV     A, R7
1453 14        +1 673          DEC     A
1454 80F4      +1 674          JMP     NextString      ; Check if we are there yet
1456          +1 675          ReturnLength:
1456 E0        +1 676          MOVX   A, @DPTR         ; Get Descriptor Length (first byte)
1457 22        +1 677          RET
+1 678          ; Error check: this MUST be on within a page of Subroutines
0062          +1 679          WithinSamePage EQU $ - Subroutines
+1 680          ;
+1 681          ;
+1 682          ;
+1 683          ;$include (DTables.A51)
+1 684          ; This module declares the descriptors
+1 685          ;
+1 686          ; This example has one Device Descriptor with:
+1 687          ;     One Configuration - 8-bit mono PCM speaker
+1 688          ;     Two Interfaces:
+1 689          ;         AudioControl - with two units
+1 690          ;         AudioStreaming - with two ALT settings
+1 691          ;     Multiple Sting Descriptors - to aid the user
+1 692          ;
+1 693          ;     CSEG
1458          +1 694          DeviceDescriptor:
1458 1201      +1 695          DB 18, 1              ; Length, Type
145A 1001      +1 696          DB 10H, 1           ; USB Rev 1.1 (=0110H, low=10H, High=01H)
145C 000000    +1 697          DB 0, 0, 0          ; Class, Subclass and Protocol
145F 40        +1 698          DB EPOSize
1460 42420900 +1 699          DB 42H, 42H, 9H, 0, 0, 1 ; Vendor ID, Product ID and Version
1464 0001
1466 010200    +1 700          DB 1, 2, 0          ; Manufacturer, Product & Serial# Names
1469 01        +1 701          DB 1                ; #Configs
146A          +1 702          ConfigurationDescriptor:
146A 0902      +1 703          DB 9, 2            ; Length, Type
146C 6400      +1 704          DB LOW(ConfigLength), HIGH(ConfigLength)
146E 020100    +1 705          DB 2, 1, 0        ; #Interfaces, Configuration#, Config. Name
1471 80        +1 706          DB 10000000b       ; Attributes = Bus Powered
1472 FA        +1 707          DB 250             ; Max. Power is 250x2 = 500mA
1473          +1 708          AudioControlInterface:
1473 0904      +1 709          DB 9, 4            ; Length, Type
1475 000000    +1 710          DB 0, 0, 0        ; ID, No alternate setting, only uses EPO
1478 01        +1 711          DB 1                ; Class = Audio
1479 01        +1 712          DB 1                ; Sub-class = Audio Control
147A 00        +1 713          DB 0                ; Protocol
147B 03        +1 714          DB 3                ; Interface name
```

```
147C          +1 715 AudioControlClassInterface:
147C 0924      +1 716         DB 9, 24H           ; Length, Type = CS_INTERFACE
147E 01        +1 717         DB 1             ; Subtype = HEADER
147F 0001      +1 718         DB 0, 1           ; Audio Device Class spec release (ver 1.0)
1481 1E00      +1 719         DB LOW(ClassLength), HIGH(ClassLength)
1483 01        +1 720         DB 1             ; Number of AudioStreaming interfaces
1484 01        +1 721         DB 1             ; AudioStreaming interface ID
1485          +1 722 InputTerminalDescriptor:
1485 0C24      +1 723         DB 12, 24H          ; Length, Type = CS_INTERFACE
1487 02        +1 724         DB 2             ; Subtype = INPUT_TERMINAL
1488 01        +1 725         DB 1             ; Terminal ID
1489 0101      +1 726         DB 1,1          ; Terminal type = USB streaming
148B 00        +1 727         DB 0             ; Terminal association (none)
148C 01        +1 728         DB 1             ; Number of logical output channels
148D 0000      +1 729         DW 0             ; Channel configuration = mono
148F 00        +1 730         DB 0             ; Channel name
1490 00        +1 731         DB 0             ; Terminal name
1491          +1 732 OutputTerminalDescriptor:
1491 0924      +1 733         DB 9, 24H           ; Length, Type = CS_INTERFACE
1493 03        +1 734         DB 3             ; Subtype = OUTPUT_TERMINAL
1494 02        +1 735         DB 2             ; Terminal ID
1495 0103      +1 736         DB 1,3          ; Terminal type = Speaker
1497 00        +1 737         DB 0             ; Terminal association (none)
1498 01        +1 738         DB 1             ; ID of the source terminal
1499 00        +1 739         DB 0             ; Terminal name
    001E      +1 740 ClassLength EQU $-AudioControlClassInterface
149A          +1 741 AudioStreamingInterfaceAlt0:
149A 0904      +1 742         DB 9, 4             ; Length, Type
149C 010000    +1 743         DB 1, 0, 0          ; ID, Alternate setting, only uses EP0
149F 01        +1 744         DB 1             ; Class = Audio
14A0 02        +1 745         DB 2             ; Sub-class = Audio Streaming
14A1 00        +1 746         DB 0             ; Protocol
14A2 04        +1 747         DB 4             ; Interface name
14A3          +1 748 AudioStreamingInterfaceAlt1:
14A3 0904      +1 749         DB 9, 4             ; Length, Type
14A5 010101    +1 750         DB 1, 1, 1          ; ID, Alternate setting, Uses isochronous endpoint
14A8 01        +1 751         DB 1             ; Class = Audio
14A9 02        +1 752         DB 2             ; Sub-class = Audio Streaming
14AA 00        +1 753         DB 0             ; Protocol
14AB 04        +1 754         DB 4             ; Interface name
14AC          +1 755 AudioStreamingClassInterface:
14AC 0724      +1 756         DB 7, 24H          ; Length, Type
14AE 01        +1 757         DB 1             ; Subtype = AS_GENERAL
14AF 01        +1 758         DB 1             ; Terminal connected to
14B0 01        +1 759         DB 1             ; Interface delay
14B1 0200      +1 760         DB 2, 0          ; Audio Data Format = PCM8
14B3          +1 761 FormatTypeDescriptor:
14B3 0B24      +1 762         DB 11, 24H          ; Length, Type
14B5 02        +1 763         DB 2             ; Subtype = FORMAT_TYPE
14B6 01        +1 764         DB 1             ; Type 1
14B7 01        +1 765         DB 1             ; Number of physical channels
14B8 01        +1 766         DB 1             ; Number of bytes in an audio subframe
14B9 08        +1 767         DB 8             ; Number of bits per sample
14BA 01        +1 768         DB 1             ; Number of frequencies supported
14BB 803E00    +1 769         DB 80H, 3EH, 0        ; 16000Hz
14BE          +1 770 EndpointDescriptor:
14BE 0905      +1 771         DB 9, 5             ; Length, Type
14C0 08        +1 772         DB 00001000B          ; OUT Endpoint 8
14C1 09        +1 773         DB 00001001B          ; Transfer type = ADAPTIVE, ISOCHRONOUS
14C2 1000      +1 774         DB 16, 0          ; Maximum packet size
14C4 01        +1 775         DB 1             ; Polling interval
14C5 00        +1 776         DB 0             ; Refresh (unused)
14C6 00        +1 777         DB 0             ; Endpoint used for sysynchronization info (unused)
14C7          +1 778 ClassEndpointDescriptor:
14C7 0725      +1 779         DB 7, 25H          ; Length, Type
14C9 01        +1 780         DB 1             ; Subtype = EP_GENERAL
```

```

14CA 00      +1  781          DB 0                ; Attributes (none)
14CB 00      +1  782          DB 0                ; Lock delay units (unused)
14CC 0000    +1  783          DW 0                ; Lock delay (unused)
   0064      +1  784          Configlength EQU $-ConfigurationDescriptor
14CE        +1  785          String0:          ; Declare the UNICODE strings
14CE 04030904 +1  786          DB      4, 3, 9, 4      ; Only English language strings supported
14D2        +1  787          String1:          ; Manufacturer
14D2 2C03    +1  788          DB      (String2-String1),3 ; Length, Type
14D4 55005300 +1  789          DB      "U",0,"S",0,"B",0," ",0,"D",0,"e",0,"s",0,"i",0,"g",0,"n",0," ",0
14D8 42002000
14DC 44006500
14E0 73006900
14E4 67006E00
14E8 2000
14EA 42007900 +1  790          DB      "B",0,"y",0," ",0,"E",0,"x",0,"a",0,"m",0,"p",0,"l",0,"e",0
14EE 20004500
14F2 78006100
14F6 6D007000
14FA 6C006500
14FE        +1  791          String2:          ; Product Name
14FE 1E03    +1  792          DB      (String3-String2),3
1500 53006900 +1  793          DB      "S",0,"i",0,"m",0,"p",0,"l",0,"e",0," ",0
1504 6D007000
1508 6C006500
150C 2000
150E 53007000 +1  794          DB      "S",0,"p",0,"e",0,"a",0,"k",0,"e",0,"r",0
1512 65006100
1516 6B006500
151A 7200
151C 1A03    +1  795          String3:DB      (String4-String3),3
151E 41007500 +1  796          DB      "A",0,"u",0,"d",0,"i",0,"o",0
1522 64006900
1526 6F00
1528 43006F00 +1  797          DB      "C",0,"o",0,"n",0,"t",0,"r",0,"o",0,"l",0
152C 6E007400
1530 72006F00
1534 6C00
1536 1E03    +1  798          String4:DB      (EndOfDescriptors-String4),3
1538 41007500 +1  799          DB      "A",0,"u",0,"d",0,"i",0,"o",0
153C 64006900
1540 6F00
1542 53007400 +1  800          DB      "S",0,"t",0,"r",0,"e",0,"a",0,"m",0,"i",0,"n",0,"g",0
1546 72006500
154A 61006D00
154E 69006E00
1552 6700
1554        +1  801          EndOfDescriptors:
1554 00      +1  802          DB      0                ; Backstop for String Descriptors
   +1  803
   +1  804
   +1  805
   +1  806          END

```