

MACRO ASSEMBLER A51 V6.10  
OBJECT MODULE PLACED IN .\BAL.OBJ  
ASSEMBLER INVOKED BY: C:\Keil\C51\BIN\A51.EXE .\BAL.A51 REGISTERBANK(0) SET(SMALL) DEBUG EP

```

LOC OBJ          LINE    SOURCE
                1      NAME    ButtonsAndLights
                2      ; Version 3.5
                3      ;
                4      ; Copyright (C) 2001, Intel Corporation
                5      ; All rights reserved.
                6      ; Permission is hereby granted to merge this program code with other program
                7      ; material to create a derivative work. This derivative work may be distributed
                8      ; in compiled object form only. Any other publication of this program, in any form,
                9      ; without the explicit permission of the copyright holder is prohibited.
               10      ;
               11      ; Send questions and comments to John.Hyde@intel.com
               12      ;
               13      ;
               14      ; This version works with dScope monSIO0.hex (uses Serial Port 0, loads at 1200H)
               15      ;
               16      ; Changes from Version 3.0
               17      ;     a) USBReset interrupt was not enabled! It now is (Thank you Greg Porter)
               18      ;     b) No reports are generated unless the device is Configured
               19      ;     c) Retrieve LENGTH missing near LoadEP0 (Thank you Greg Porter)
               20      ;     d) Windows given 200msec to unload driver during renumerate
               21      ;
               22      ; Changes from Version 2.0
               23      ;     a) Two misplaced labels corrected
               24      ;     b) CommandTable moved outside of the constrained page
               25      ;     c) SETUPDAT buffer copied to direct access memory, simplified coding
               26      ;     d) Optional Set_Idle now not supported (returns STALL, not ignorred)
               27      ;     e) R7 used in place of Temp, saved code space
               28      ;     f) Code reorganized to separate hardware dependant sections
               29      ;
               30      ;         Old          New
               31      ;         USBINT.A51    Decode.A51
               32      ;         Vectors.A51    EZInt.A51
               33      ;         Timer.A51      EZInt.A51
               34      ;         Main.A51      EZMain.A51
               35      ;     g) EP0Size made an equate to ease coding of other components
0040      35      EP0Size EQU    64      ; For EZ-USB
               36      ;     h) Code added for descriptors > EP0Size
               37      ;
               38      ; Changes from Version 1.0
               39      ;     a) Register saving removed from Vectors.A51
               40      ;         Main has no context which needs to be saved
               41      ;     b) There was a race condition in USB_INT::Set_Report: which
               42      ;         could cause OLD data to be read. Busy algorithm changed
               43      ;     c) The USB Version# was incorrectly declared in the Device Descriptor
               44      ;         It was 0101H and is now 0110H (data from USB IF)
               45      ;
               46      ;$include (Declare.A51)
               +1      47      ; This module declares the variables and constants used in the examples
               +1      48      ; It is common to all of the examples
               +1      49      ;
               +1      50      ; Declare Special Function Registers used
0088      +1      51      TimerControl    DATA    088H
0089      +1      52      TimerMode      DATA    089H
008C      +1      53      Timer0High     DATA    08CH
00A8      +1      54      EI              DATA    0A8H
00E8      +1      55      EIE           DATA    0E8H      ; EZ-USB specific
0091      +1      56      EXIF          DATA    091H      ; EZ-USB specific
00D8      +1      57      EICON         DATA    0D8H      ; EZ-USB specific
0092      +1      58      PageReg        DATA    092H      ; EZ-USB specific, used with MOVX @Ri

```

```

0086      +1  59  DPS          DATA    086H    ; EZ-USB specific, used with dual data pointers
          +1  60  ;
          +1  61  ; "External" memory locations used, EZ-USB specific
          +1  62  ; Note that most of these variables are in Page 7FH
7FE8      +1  63  SETUPDAT    EQU     07FE8H
7FD4      +1  64  SUDPTR     EQU     07FD4H
7FB4      +1  65  EP0Control  EQU     07FB4H
7F00      +1  66  EP0InBuffer EQU     07F00H
7EC0      +1  67  EP0OutBuffer EQU     07EC0H          ; Not in Page 7FH
7E80      +1  68  EP1InBuffer EQU     07E80H          ; Not in Page 7FH
7FB5      +1  69  IN0ByteCount EQU     07FB5H
7FC5      +1  70  Out0ByteCount EQU     07FC5H
7FB7      +1  71  IN1ByteCount EQU     07FB7H
7FAC      +1  72  IN07IEN    EQU     07FACH
7FA9      +1  73  IN07IRQ    EQU     07FA9H
7FAD      +1  74  OUT07IEN   EQU     07FADH
7FAA      +1  75  OUT07IRQ   EQU     07FAAH
7FAE      +1  76  USBIEN    EQU     07FAEH
7FAB      +1  77  USBIRQ    EQU     07FABH
7FD6      +1  78  USBControl EQU     07FD6H
7FA6      +1  79  I2CData    EQU     07FA6H
7FA5      +1  80  I2CControl EQU     07FA5H
7F93      +1  81  PortA_Config EQU     07F93H
7F94      +1  82  PortB_Config EQU     07F94H
7F95      +1  83  PortC_Config EQU     07F95H
7F96      +1  84  PortA_OUT   EQU     07F96H
7F97      +1  85  PortB_OUT   EQU     07F97H
7F98      +1  86  PortC_OUT   EQU     07F98H
7F99      +1  87  PortA_PINS  EQU     07F99H
7F9A      +1  88  PortB_PINS  EQU     07F9AH
7F9B      +1  89  PortC_PINS  EQU     07F9BH
7F9C      +1  90  PortA_OE    EQU     07F9CH
7F9D      +1  91  PortB_OE    EQU     07F9DH
7F9E      +1  92  PortC_OE    EQU     07F9EH
          +1  93  ;
          +1  94  ; Byte Variables
          +1  95
-----  +1  96          DSEG    AT 20H
0020      +1  97  FLAGS:      DS      1      ; This register is bit-addressable
          +1  98  ; Bit Variables
          +1  99  Configured  EQU     FLAGS.0 ; Is this device configured
          0001  +1 100  STALL      EQU     FLAGS.1 ; Need to STALL endpoint 0
          0002  +1 101  SendData   EQU     FLAGS.2 ; Need to send data to PC Host
          0003  +1 102  IsDescriptor EQU     FLAGS.3 ; Enable a shortcut reply
          0004  +1 103  SetAddress  EQU     FLAGS.4 ; Set the SIE address
          +1 104  ;
0021      +1 105  MonitorSpace: DS      1FH    ; Used by Dscope
0040      +1 106  DataSpace:
0040      +1 107  ReplyCount:  DS      1      ; Byte count for following buffer
0041      +1 108  ReplyBuffer:  DS      2      ; Buffer for immediate reply
0043      +1 109  CurrentConfiguration:
0043      +1 110          DS      1      ; Some examples support > 1 configurations
0044      +1 111  SaveDPH:      DS      1      ; Needed to save Descriptor Pointer ..
0045      +1 112  SaveDPL:      DS      1      ; .. for descriptors > EP0Size
0046      +1 113  SaveLength:   DS      1      ; Number of bytes still to send
0047      +1 114  SetupData:    ; Buffer in direct access memory
0047      +1 115  RequestType:  DS      1
0048      +1 116  Request:      DS      1
0049      +1 117  wValueLow:    DS      1
004A      +1 118  wValueHigh:   DS      1
004B      +1 119  wIndexLow:    DS      1
004C      +1 120  wIndexHigh:   DS      1
004D      +1 121  wLengthLow:   DS      1
004E      +1 122  wLengthHigh:  DS      1
          +1 123  ;
004F      +1 124  Old_Buttons:  DS      1      ; Used by BAL: stores current button position

```

```
0050      +1 125 LEDstrobe: DS 1 ; Used by BAL: strobe one LED on at a time
0051      +1 126 LEDvalue: DS 1 ; Used by BAL: stores current LED value
0052      +1 127 Msec_Counter: DS 1 ; Used by BAL: counts up to 4 msec
          +1 128 ;
          129
          130 ;$include (EZInt.A51)
          +1 131 ; This module contains all the EZUSB-specific hardware code
          +1 132 ; This module also contains all of the interrupt vector declarations and
          +1 133 ; the first level interrupt servicing (register save, call subroutine,
          +1 134 ; clear interrupt source, restore registers, return)
          +1 135 ; Suspend and Resume are handled totally in this module
          +1 136 ;
          +1 137 ; A Reset sends us to Program space location 0
----- +1 138 CSEG AT 0 ; Code space
          +1 139 USING 0 ; Reset forces Register Bank 0
0000 021308 +1 140 LJMP Reset
          +1 141 ;
          +1 142 ; The interrupt vector table is also located here
          +1 143 ; EZ-USB has two levels of USB interrupts:
          +1 144 ; 1-the main level is described in this table (at ORG 43H)
          +1 145 ; 2-there are 21 sources of USB interrupts and these are described in USB_ISR
          +1 146 ; This means that two levels of acknowledgement and clearing will be required
          +1 147 ; LJMP INT0_ISR ; Features not used are commented out
          +1 148 ; ORG 0BH
          +1 149 ; LJMP Timer0_ISR
          +1 150 ; ORG 13H
          +1 151 ; LJMP INT1_ISR
          +1 152 ; ORG 1BH
          +1 153 ; LJMP Timer1_ISR
          +1 154 ; ORG 23H
          +1 155 ; LJMP UART0_ISR
          +1 156 ; ORG 2BH
          +1 157 ; LJMP Timer2_ISR
0033      +1 158 ORG 33H
0033 021245 +1 159 LJMP WakeUp_ISR
          +1 160 ; ORG 3BH
          +1 161 ; LJMP UART1_ISR
0043      +1 162 ORG 43H
0043 021200 +1 163 LJMP USB_ISR ; Auto Vector will replace byte 45H
          +1 164 ; ORG 4BH
          +1 165 ; LJMP I2C_ISR
          +1 166 ; ORG 53H
          +1 167 ; LJMP INT4_ISR
          +1 168 ; ORG 5BH
          +1 169 ; LJMP INT5_ISR
          +1 170 ; ORG 63H
          +1 171 ; LJMP INT6_ISR
          +1 172
1200      +1 173 ORG 1200H ; Load above monSIO0.hex
1200 021283 +1 174 USB_ISR:LJMP SUDAV_ISR
1203 00      +1 175 DB 0 ; Pad entries to 4 bytes
1204 02126C +1 176 LJMP SOF_ISR
1207 00      +1 177 DB 0
1208 02121B +1 178 LJMP SUTOK_ISR
120B 00      +1 179 DB 0
120C 021230 +1 180 LJMP Suspend_ISR
120F 00      +1 181 DB 0
1210 021223 +1 182 LJMP USBReset_ISR
1213 00      +1 183 DB 0
1214 02121B +1 184 LJMP Reserved
1217 00      +1 185 DB 0
1218 02125B +1 186 LJMP EP0In_ISR
          +1 187 ; DB 0 ; Comment out features not used
          +1 188 ; LJMP EP0Out_ISR
          +1 189 ; DB 0
          +1 190 ; LJMP EP1In_ISR
```

```

+1 191 ; DB 0
+1 192 ; LJMP EP1Out_ISR
+1 193 ; DB 0
+1 194 ; LJMP EP2In_ISR
+1 195 ; DB 0
+1 196 ; LJMP EP2Out_ISR
+1 197 ; DB 0
+1 198 ; LJMP EP3In_ISR
+1 199 ; DB 0
+1 200 ; LJMP EP3Out_ISR
+1 201 ; DB 0
+1 202 ; LJMP EP4In_ISR
+1 203 ; DB 0
+1 204 ; LJMP EP4Out_ISR
+1 205 ; DB 0
+1 206 ; LJMP EP5In_ISR
+1 207 ; DB 0
+1 208 ; LJMP EP5Out_ISR
+1 209 ; DB 0
+1 210 ; LJMP EP6In_ISR
+1 211 ; DB 0
+1 212 ; LJMP EP6Out_ISR
+1 213 ; DB 0
+1 214 ; LJMP EP7In_ISR
+1 215 ; DB 0
+1 216 ; LJMP EP7Out_ISR
+1 217 ; End of Interrupt Vector tables
+1 218
+1 219 ; When a feature is used insert the required interrupt processing here
+1 220 ; The example use only used Endpoints 0 and 1 and also SOF for timing
121B +1 221 Reserved:
121B +1 222 INT0_ISR:
121B +1 223 Timer0_ISR:
121B +1 224 INT1_ISR:
121B +1 225 Timer1_ISR:
121B +1 226 UART0_ISR:
121B +1 227 Timer2_ISR:
121B +1 228 UART1_ISR:
121B +1 229 I2C_ISR:
121B +1 230 INT4_ISR:
121B +1 231 INT5_ISR:
121B +1 232 INT6_ISR:
121B +1 233 SUTOK_ISR:
121B +1 234 EP0Out_ISR:
121B +1 235 EP1In_ISR:
121B +1 236 EP1Out_ISR:
121B +1 237 EP2In_ISR:
121B +1 238 EP2Out_ISR:
121B +1 239 EP3In_ISR:
121B +1 240 EP3Out_ISR:
121B +1 241 EP4In_ISR:
121B +1 242 EP4Out_ISR:
121B +1 243 EP5In_ISR:
121B +1 244 EP5Out_ISR:
121B +1 245 EP6In_ISR:
121B +1 246 EP6Out_ISR:
121B +1 247 EP7In_ISR :
121B +1 248 EP7Out_ISR:
121B +1 249 Not_Used: ; Should not get any of these
121B 32 +1 250 RETI
+1 251
121C +1 252 ClearINT2: ; Tell the hardware that we're done
121C E591 +1 253 MOV A, EXIF
121E C2E4 +1 254 CLR ACC.4 ; Clear the Interrupt 2 bit
1220 F591 +1 255 MOV EXIF, A
1222 22 +1 256 RET

```

```

+1 257
1223 +1 258 USBReset_ISR: ; Bus has been Reset, move to DEFAULT state
1223 C200 +1 259 CLR Configured
1225 9010AB +1 260 MOV DPTR, #(1000H OR LOW(USBIRQ))
1228 +1 261 ExitISR: ; Common exit for all ISR's
+1 262 ; On entry DPH = Interrupt ID, DPL = LOW(Interrupt Register)
1228 511C +1 263 CALL ClearINT2
122A 747F +1 264 MOV A, #7FH ; EZ-USB I/O Register Page
122C C583 +1 265 XCH A, DPH
122E F0 +1 266 MOVX @DPTR, A ; Clear source of interrupt
122F 32 +1 267 RETI
+1 268
1230 +1 269 Suspend_ISR: ; SIE detected an Idle bus
+1 270 ; This routine assumes that MAIN has no context to save
1230 511C +1 271 CALL ClearINT2
1232 78AB +1 272 MOV R0, #LOW(USBIRQ) ; PageReg = 7FH = HIGH(USBIRQ)
1234 7408 +1 273 MOV A, #8
1236 F2 +1 274 MOVX @R0, A ; Clear the interrupt before suspending
1237 E4 +1 275 CLR A
1238 7897 +1 276 MOV R0, #LOW(PortB_Out) ; PageReg = 7FH = HIGH(PortB_Out)
123A F2 +1 277 MOVX @R0, A ; Turn off LEDs
123B E587 +1 278 MOV A, PCON
123D C2E0 +1 279 CLR ACC.0
123F F587 +1 280 MOV PCON, A ; Turn off the oscillator
1241 00 +1 281 NOP
1242 00 +1 282 IDLE: NOP ; Enter IDLE mode
1243 00 +1 283 NOP
1244 32 +1 284 RETI
+1 285
1245 +1 286 Wakeup_ISR: ; Wake up from IDLE mode
1245 C2DC +1 287 CLR EICON.4 ; Clear the interrupt signal
+1 288 ; This routine assumes that MAIN has no context to save
1247 78D6 +1 289 MOV R0, #LOW(USBControl)
1249 E2 +1 290 MOVX A, @R0 ; Get USB Control/Status register
124A 20E701 +1 291 JB ACC.7, IOdeviceRequest
124D +1 292 PChostRequest: ; The wakeup was caused by the PC Host
124D 32 +1 293 RETI
124E +1 294 IOdeviceRequest: ; The IO device wants to wakeup the PC Host
+1 295 ; NOTE that this capability is included in the Configuration Descriptor
124E 4581 +1 296 ORL A, 10000001B
1250 F2 +1 297 MOVX @R0, A ; Clear Wakeup source and signal a RESUME
1251 557E +1 298 ANL A, 01111110B
1253 FA +1 299 MOV R2, A ; Save ACC
1254 7F0F +1 300 MOV R7, #15
1256 7155 +1 301 CALL Wait1msec ; Signal a RESUME for 15ms
1258 EA +1 302 MOV A, R2
1259 F2 +1 303 MOVX @R0, A ; Remove RESUME signal
+1 304 ; We're done, return to the Suspend_ISR
125A 32 +1 305 RETI
+1 306
125B +1 307 EP0In_ISR: ; A prepared packet has been read by PC
host
125B E546 +1 308 MOV A, SaveLength ; Do I have any more data to send?
125D 6008 +1 309 JZ NoMoreToSend
125F 854483 +1 310 MOV DPH, SaveDPH ; Retrieve descriptor pointer
1262 854582 +1 311 MOV DPL, SaveDPL
1265 51D0 +1 312 CALL SendNextPieceOfDescriptor
1267 +1 313 NoMoreToSend:
1267 9001A9 +1 314 MOV DPTR, #(100H OR LOW(IN07IRQ))
126A 80BC +1 315 JMP ExitISR
+1 316
126C +1 317 SOF_ISR: ; A Start-Of-Frame packet has been received
+1 318 ; CALL ServiceTimerRoutine
+1 319 ; This routine services the real time interrupt
+1 320 ; It is also responsible for the "real world" buttons and lights
+1 321 ;
126C +1 322 ServiceTimerRoutine:

```

```
126C D5520F +1 323          DJNZ   Msec_counter, Done      ; Only need to check every 4msec
126F 755204 +1 324          MOV    Msec_counter, #4        ; Reinitialize
+1 325          ; LED task
1272 E551   +1 326          MOV    A, LEDValue
1274 907F97 +1 327          MOV    DPTR, #PortB_Out
1277 F0     +1 328          MOVX   @DPTR, A                ; Update the real world
+1 329          ;
+1 330          ; Create an Input Report from the Buttons value
+1 331          ; This will be continually overwritten while the PCHost is not polling for data
1278       +1 332          ReadButtons:
1278 907F99 +1 333          MOV    DPTR, #PortA_Pins
127B E0     +1 334          MOVX   A, @DPTR
127C 716C   +1 335          CALL  CreateInputReport
127E 9002AB +1 336          Done:  MOV    DPTR, #(200H OR LOW(USBRQ))
1281 80A5   +1 337          JMP    ExitISR
+1 338          ;
1283       +1 339          SUDAV_ISR:                    ; A Setup packet has been received
1283 754600 +1 340          MOV    SaveLength, #0        ; Clear any pending transactions (if any)
1286 907FE8 +1 341          MOV    DPTR, #SETUPDAT        ; Copy packet to direct access memory
1289 7847   +1 342          MOV    R0, #SetupData
128B 7F08   +1 343          MOV    R7, #8
128D E0     +1 344          CopySD: MOVX   A, @DPTR
128E F6     +1 345          MOV    @R0, A
128F A3     +1 346          INC    DPTR
1290 08     +1 347          INC    R0
1291 DFFA   +1 348          DJNZ   R7, CopySD
1293 717A   +1 349          CALL  ServiceSetupPacket      ; Handle the decode of the Setup packet
+1 350          ; if SetAddress { Update SIE address } // NOP on EZ-USB
+1 351          ; if STALL { Stall the endpoint }
+1 352          ; if SendData {
+1 353          ;     if IsDescriptor { send DPTR->descriptor, A = length }
+1 354          ;     else { send ReplyBuffer }
+1 355          ; }
1295 200126 +1 356          JB    STALL, SendSTALL
1298 300216 +1 357          JNB   SendData, HandShake
129B 200324 +1 358          JB    IsDescriptor, LoadEP0
+1 359          ; Send data in ReplyBuffer
129E 907F01 +1 360          MOV    DPTR, #EP0InBuffer+1
12A1 7842   +1 361          MOV    R0, #ReplyBuffer+1
12A3 7F02   +1 362          MOV    R7, #2                ; Copy the two byte buffer
12A5 E6     +1 363          CopyRB: MOV    A, @R0
12A6 F0     +1 364          MOVX   @DPTR, A
12A7 1582   +1 365          DEC    DPL
12A9 18     +1 366          DEC    R0
12AA DFF9   +1 367          DJNZ   R7, CopyRB
12AC E6     +1 368          MOV    A, @R0                ; Get BufferCount
12AD       +1 369          SendEP0InBuffer:
12AD 907FB5 +1 370          MOV    DPTR, #In0ByteCount
12B0       +1 371          StartXfer:
12B0 F0     +1 372          MOVX   @DPTR, A                ; This write initiates the transfer
12B1       +1 373          HandShake:
12B1 7F02   +1 374          MOV    R7, #00000010b        ; Set HSNACK to tell the SIE that we're done
12B3       +1 375          SetEP0Control:
12B3 907FB4 +1 376          MOV    DPTR, #EP0Control
12B6 E0     +1 377          MOVX   A, @DPTR
12B7 4F     +1 378          ORL   A, R7
12B8 F0     +1 379          MOVX   @DPTR, A                ; We're done
12B9 9001AB +1 380          MOV    DPTR, #(100H OR LOW(USBRQ))
12BC 4128   +1 381          JMP    ExitISR
12BE       +1 382          SendSTALL:                    ; Invalid Request was received
12BE 7F03   +1 383          MOV    R7, #00000011b        ; Set EPOSTALL and HSNACK
12C0 80F1   +1 384          JMP    SetEP0Control
12C2       +1 385          LoadEP0:                        ; Send the data pointed to by DPTR
12C2 FF     +1 386          MOV    R7, A                ; Save LENGTH
+1 387          ; Need to return the smaller of "Requested Length" and "Actual Length"
+1 388          ; If "Requested Length" > 255 then use "Actual Length"
```

```
+1 389 ; There are no descriptors > 255 in this example
12C3 E54E +1 390 MOV A, wLengthHigh
12C5 7008 +1 391 JNZ UseActual
12C7 EF +1 392 MOV A, R7 ; Retrieve LENGTH
12C8 C3 +1 393 CLR C
12C9 954D +1 394 SUBB A, wLengthLow
12CB E54D +1 395 MOV A, wLengthLow ; This does not affect Carry
12CD 5001 +1 396 JNC UsewLengthLow
12CF +1 397 UseActual:
12CF EF +1 398 MOV A, R7
12D0 +1 399 UsewLengthLow:
12D0 +1 400 SendNextPieceOfDescriptor: ; DPTR -> Descriptor to be sent
12D0 FF +1 401 MOV R7, A ; Save LENGTH again
12D1 754600 +1 402 MOV SaveLength, #0 ; Default case, overwrite if necessary
+1 403 ; Do I have more than a single packet to send?
12D4 C3 +1 404 CLR C
12D5 9440 +1 405 SUBB A, #EP0Size
12D7 4015 +1 406 JC SendPacket
+1 407 ; Need to send multiple packets.
+1 408 ; Calculate and save address of next packet, send next packet now
12D9 F546 +1 409 MOV SaveLength, A ; Send these next time
12DB 7F40 +1 410 MOV R7, #EP0Size
12DD C083 +1 411 PUSH DPH ; Save current pointer
12DF C082 +1 412 PUSH DPL
12E1 EF +1 413 MOV A, R7 ; Retrieve length
12E2 71BD +1 414 CALL BumpDPTR
12E4 858344 +1 415 MOV SaveDPH, DPH
12E7 858245 +1 416 MOV SaveDPL, DPL
12EA D082 +1 417 POP DPL
12EC D083 +1 418 POP DPH
12EE +1 419 SendPacket:
12EE EF +1 420 MOV A, R7 ; Retrieve length
12EF FE +1 421 MOV R6, A ; Save length in R6 for move
12F0 7800 +1 422 MOV R0, #LOW(EP0InBuffer) ; PageReg = 7FH = HIGH(EP0InBuffer)
12F2 E0 +1 423 CopySTD:MOVX A, @DPTR
12F3 F2 +1 424 MOVX @R0, A
12F4 A3 +1 425 INC DPTR
12F5 08 +1 426 INC R0
12F6 DEFA +1 427 DJNZ R6, CopySTD
12F8 EF +1 428 MOV A, R7 ; Retrieve LENGTH
12F9 80B2 +1 429 JMP SendEP0InBuffer
+1 430
12FB +1 431 GetOutputReport: ; Wait for this, it's next on USB
12FB 907FC5 +1 432 MOV DPTR, #Out0ByteCount ; Enable EP0OutBuffer to receive data
12FE F0 +1 433 MOVX @DPTR, A ; Any value will do
12FF 907FB4 +1 434 MOV DPTR, #EP0Control ; Wait for valid data in EP0OutBuffer
1302 E0 +1 435 Wait40:MOVX A, @DPTR
1303 5408 +1 436 ANL A, #00001000b ; Check OUTBSY
1305 70FB +1 437 JNZ Wait40
1307 22 +1 438 RET
+1 439
+1 440
+1 441
+1 442
+1 443 ;$include (EZMain.A51)
+1 444 ; This module initializes the microcontroller then executes MAIN forever
+1 445 ; It is hardware dependant
+1 446
1308 +1 447 Reset:
1308 7581DF +1 448 MOV SP, #0DFH ; Initialize the Stack
130B 75927F +1 449 MOV PageReg, #7FH ; Allows MOVX Ri to access EZ-USB memory
+1 450
130E 78D6 +1 451 MOV R0, #Low(USBControl) ; Simulate a disconnect
1310 E2 +1 452 MOVX A, @R0
1311 54F3 +1 453 ANL A, #11110011b ; Clear DISCON, DISCOE
1313 F2 +1 454 MOVX @R0, A
```



```
1364 22      +1 521          RET
              +1 522
1365          +1 523      ProcessOutputReport:          ; A Report has just been received
              +1 524      ; The report is only one byte long in this first example
              +1 525      ; It contains a new value for the LEDs
1365 907EC0   +1 526          MOV      DPTR, #EP0OutBuffer      ; Point to the Report
1368 E0       +1 527          MOVX     A, @DPTR              ; Get the Data
1369 F551     +1 528          MOV      LEDValue, A          ; Update the local variable
136B 22      +1 529          RET
              +1 530
136C          +1 531      CreateInputReport:          ; Called from TIMER which detected the need
              +1 532      ; The report is only one byte long in this first example
              +1 533      ; It contains a new value for the Buttons
136C 30000A   +1 534          JNB      Configured, NoReport      ; Must be Configured to create reports
136F 907E80   +1 535          MOV      DPTR, #EP1InBuffer      ; Point to the buffer
1372 F0       +1 536          MOVX     @DPTR, A              ; Update the Report
1373 907FB7   +1 537          MOV      DPTR, #IN1ByteCount
1376 7401     +1 538          MOV      A, #1
1378 F0       +1 539          MOVX     @DPTR, A              ; Endpoint 1 now 'armed', next IN will get
data
1379          +1 540      NoReport:
1379 22      +1 541          RET
              +1 542
              543
              544      ;$include (Decode.A51)
              +1 545      ; This module is common to all of the examples.
              +1 546      ; It decodes the USB Setup Packets and generates appropriate responses.
              +1 547      ; Interpretation of Reports is handled by MAIN
              +1 548      ;
----         +1 549          CSEG
137A          +1 550      ServiceSetupPacket:
137A E547     +1 551          MOV      A, RequestType
137C A2E7     +1 552          MOV      C, ACC.7          ; Bit 7 = 1 means IO device needs to send
data to P
              +1 553
              +1 554      C Host
137E 9202     +1 553          MOV      SendData, C
1380 545C     +1 554          ANL     A, #01011100b      ; IF RequestType[6.4.3.2] = 1 THEN goto
BadRequest
1382 7035     +1 555          JNZ     BadRequest
1384 E547     +1 556          MOV      A, RequestType      ; IF RequestType[1&0] = 1 THEN goto
BadRequest
1386 A2E0     +1 557          MOV      C, ACC.0
1388 82E1     +1 558          ANL     C, ACC.1
138A 402D     +1 559          JC      BadRequest
138C 30E502   +1 560          JNB     ACC.5, NotB5      ; IF RequestType[5] = 1 THEN
RequestType[1,0] = [1,
              1]
138F 7403     +1 561          MOV      A, #00000011b
1391 5403     +1 562      NotB5: ANL     A, #00000011b      ; Set CommandIndex[5,4] = RequestType[1,0]
1393 C4       +1 563          SWAP     A
1394 FF       +1 564          MOV      R7, A          ; Save HI nibble of CommandIndex
              +1 565          ; Set CommandIndex[3,0] = Request[3,0]
1395 E548     +1 566          MOV      A, Request
1397 54F0     +1 567          ANL     A, #11110000b      ; Check if Request > 15
1399 701E     +1 568          JNZ     BadRequest
139B E548     +1 569          MOV      A, Request
139D 540F     +1 570          ANL     A, #00001111b      ; Only 13 are defined today, handle in
table
139F 4F       +1 571          ORL     A, R7
              +1 572          ; CALL    CorrectSubroutine      ; goto CommandTable(CommandIndex)
13A0          +1 573      CorrectSubroutine:          ; Jump to the subroutine that DPTR is
pointing to
13A0 754001   +1 574          MOV      ReplyCount, #1      ; Set up a default reply
13A3 754100   +1 575          MOV      ReplyBuffer, #0
13A6 754200   +1 576          MOV      ReplyBuffer+1, #0
13A9 C204     +1 577          CLR     SetAddress      ; Clear all flags
13AB C201     +1 578          CLR     STALL
13AD C203     +1 579          CLR     IsDescriptor
13AF 9013C6   +1 580          MOV      DPTR, #CommandTable
13B2 71BD     +1 581          CALL    BumpDPTR          ; Point to entry
13B4 E0       +1 582          MOVX     A, @DPTR              ; Get the offset
13B5 901406   +1 583          MOV      DPTR, #Subroutines
13B8 73       +1 584          JMP     @A+DPTR          ; Go to the correct Subroutine
```

```

+1 585
13B9 +1 586 BadRequest: ; Decoded a Bad Request, STALL the Endpoint
13B9 D201 +1 587 SETB STALL
13BB 22 +1 588 RET
+1 589 ; Support routines
13BC +1 590 NextDPTR: ; Returns (DPTR + byte DPTR is pointing to)
13BC E0 +1 591 MOVX A, @DPTR
13BD +1 592 BumpDPTR: ; Returns (DPTR + ACC)
13BD 2582 +1 593 ADD A, DPL
13BF F582 +1 594 MOV DPL, A
13C1 5002 +1 595 JNC Skip
13C3 0583 +1 596 INC DPH ; Need 16 bit arithmetic here
13C5 22 +1 597 Skip: RET
+1 598
+1 599 ; Since the table only contains byte offsets, it is important that all these
routines are
+1 600 ; within one page (100H) of Subroutines
+1 601 ; V3.0 - CommandTable moved outside of this one page limited space
13C6 +1 602 CommandTable:
+1 603 ; First 16 commands are for the Device
13C6 18 +1 604 DB LOW(Device_Get_Status - Subroutines)
13C7 00 +1 605 DB LOW(Device_Clear_Feature - Subroutines)
13C8 00 +1 606 DB LOW(Invalid - Subroutines)
13C9 00 +1 607 DB LOW(Device_Set_Feature - Subroutines)
13CA 00 +1 608 DB LOW(Invalid - Subroutines)
13CB 03 +1 609 DB LOW(Set_Address - Subroutines)
13CC 32 +1 610 DB LOW(Get_Descriptor - Subroutines)
13CD 00 +1 611 DB LOW(Set_Descriptor - Subroutines)
13CE 14 +1 612 DB LOW(Get_Configuration - Subroutines)
13CF 20 +1 613 DB LOW(Set_Configuration - Subroutines)
13D0 00 +1 614 DB LOW(Invalid - Subroutines)
13D1 00 +1 615 DB LOW(Invalid - Subroutines)
13D2 00 +1 616 DB LOW(Invalid - Subroutines)
13D3 00 +1 617 DB LOW(Invalid - Subroutines)
13D4 00 +1 618 DB LOW(Invalid - Subroutines)
13D5 00 +1 619 DB LOW(Invalid - Subroutines)
+1 620 ; Next 16 commands are for the Interface
13D6 1C +1 621 DB LOW(Interface_Get_Status - Subroutines)
13D7 00 +1 622 DB LOW(Interface_Clear_Feature - Subroutines)
13D8 00 +1 623 DB LOW(Invalid - Subroutines)
13D9 00 +1 624 DB LOW(Interface_Set_Feature - Subroutines)
13DA 00 +1 625 DB LOW(Invalid - Subroutines)
13DB 00 +1 626 DB LOW(Invalid - Subroutines)
13DC 59 +1 627 DB LOW(Get_Class_Descriptor - Subroutines)
13DD 00 +1 628 DB LOW(Set_Class_Descriptor - Subroutines)
13DE 00 +1 629 DB LOW(Invalid - Subroutines)
13DF 00 +1 630 DB LOW(Invalid - Subroutines)
13E0 00 +1 631 DB LOW(Get_Interface - Subroutines)
13E1 00 +1 632 DB LOW(Set_Interface - Subroutines)
13E2 00 +1 633 DB LOW(Invalid - Subroutines)
13E3 00 +1 634 DB LOW(Invalid - Subroutines)
13E4 00 +1 635 DB LOW(Invalid - Subroutines)
13E5 00 +1 636 DB LOW(Invalid - Subroutines)
+1 637 ; Next 16 commands are for the Endpoint
13E6 1C +1 638 DB LOW(Endpoint_Get_Status - Subroutines)
13E7 00 +1 639 DB LOW(Endpoint_Clear_Feature - Subroutines)
13E8 00 +1 640 DB LOW(Invalid - Subroutines)
13E9 00 +1 641 DB LOW(Endpoint_Set_Feature - Subroutines)
13EA 00 +1 642 DB LOW(Invalid - Subroutines)
13EB 00 +1 643 DB LOW(Invalid - Subroutines)
13EC 00 +1 644 DB LOW(Invalid - Subroutines)
13ED 00 +1 645 DB LOW(Invalid - Subroutines)
13EE 00 +1 646 DB LOW(Invalid - Subroutines)
13EF 00 +1 647 DB LOW(Invalid - Subroutines)
13F0 00 +1 648 DB LOW(Invalid - Subroutines)
13F1 00 +1 649 DB LOW(Invalid - Subroutines)
13F2 00 +1 650 DB LOW(Endpoint_Sync_Frame - Subroutines)

```

```

13F3 00      +1  651      DB LOW(Invalid - Subroutines)
13F4 00      +1  652      DB LOW(Invalid - Subroutines)
13F5 00      +1  653      DB LOW(Invalid - Subroutines)
                +1  654      ; Next 16 commands are Class Requests
13F6 00      +1  655      DB LOW(Invalid - Subroutines)
13F7 0D      +1  656      DB LOW(Get_Report - Subroutines)
13F8 00      +1  657      DB LOW(Get_Idle - Subroutines)
13F9 00      +1  658      DB LOW(Get_Protocol - Subroutines)
13FA 00      +1  659      DB LOW(Invalid - Subroutines)
13FB 00      +1  660      DB LOW(Invalid - Subroutines)
13FC 00      +1  661      DB LOW(Invalid - Subroutines)
13FD 00      +1  662      DB LOW(Invalid - Subroutines)
13FE 00      +1  663      DB LOW(Invalid - Subroutines)
13FF 06      +1  664      DB LOW(Set_Report - Subroutines)
1400 00      +1  665      DB LOW(Set_Idle - Subroutines)
1401 00      +1  666      DB LOW(Set_Protocol - Subroutines)
1402 00      +1  667      DB LOW(Invalid - Subroutines)
1403 00      +1  668      DB LOW(Invalid - Subroutines)
1404 00      +1  669      DB LOW(Invalid - Subroutines)
1405 00      +1  670      DB LOW(Invalid - Subroutines)
                +1  671
1406         +1  672      Subroutines:
                +1  673      ;
                +1  674      ; Many requests are INVALID for this example
1406         +1  675      Get_Protocol:           ; We are not a Boot device
1406         +1  676      Set_Protocol:           ; We are not a Boot device
1406         +1  677      Set_Descriptor:         ; Our Descriptors are static
1406         +1  678      Set_Class_Descriptor:    ; Our Descriptors are static
1406         +1  679      Set_Interface:         ; We only have one Interface
1406         +1  680      Get_Interface:         ; We do not have an Alternate setting
1406         +1  681      Set_Idle:             ; V3.0 Optional command, not supported
1406         +1  682      Get_Idle:             ; V3.0 Optional command, not supported
1406         +1  683      Device_Set_Feature:    ; We have no features that can be set or cleared
1406         +1  684      Interface_Set_Feature: ; We have no features that can be set or cleared
1406         +1  685      Endpoint_Set_Feature:  ; We have no features that can be set or cleared
1406         +1  686      Endpoint_Clear_Feature: ; V3.0 We have no features that can be set or
cleared
1406         +1  687      Device_Clear_Feature:  ; We have no features that can be set or cleared
1406         +1  688      Interface_Clear_Feature: ; We have no features that can be set or cleared
1406         +1  689      Endpoint_Sync_Frame:   ; We are not an Isonchronous device
                +1  690
1406         +1  691      Invalid:               ; Invalid Request made, STALL the Endpoint
1406 D201     +1  692      SETB     STALL
1408 22       +1  693      Reply:  RET
                +1  694
1409         +1  695      Set_Address:           ; Set the address that the SIE will respond to
1409 D204     +1  696      SETB     SetAddress
140B 22       +1  697      RET
                +1  698
140C         +1  699      Set_Report:           ; Host wants to sent us a Report.
                +1  700      ; The ONLY case in this example where host sends data to us
140C 3000F7   +1  701      JNB     Configured, Invalid ; Need to be Configured to do this command
140F 51FB     +1  702      CALL    GetOutputReport ; Handled in EZUSB.A51
1411 6165     +1  703      JMP     ProcessOutputReport ; RETurn via this subroutine
1413         +1  704      Get_Report:         ; Host wants a Report
1413 3000F0   +1  705      JNB     Configured, Invalid ; Need to be Configured to do this command
1416 754142   +1  706      MOV     ReplyBuffer, #42H ; Reply with a recognizable (arbitrary)
value
1419 22       +1  707      RET
141A         +1  708      Get_Configuration:    ; Respond with CurrentConfiguration
141A 854341   +1  709      MOV     ReplyBuffer, CurrentConfiguration
141D 22       +1  710      RET
141E         +1  711      Device_Get_Status:    ; Only two bits of Device Status are
defined
141E 754101   +1  712      MOV     ReplyBuffer, #1 ; Bit 1=Remote Wakeup(=0), Bit 0=Self
Powered(=1)
1421 22       +1  713      RET
1422         +1  714      Interface_Get_Status:  ; Interface Status is currently defined as
0
1422         +1  715      Endpoint_Get_Status:
1422 754002   +1  716      MOV     ReplyCount, #2 ; Need a two byte 0 response

```

```

1425 22      +1 717      RET
1426      +1 718      Set_Configuration:                ; Valid values are 0 and 1
1426 E549    +1 719      MOV      A, wValueLow
1428 6009    +1 720      JZ      Deconfigured
142A 14      +1 721      DEC      A
142B 70D9    +1 722      JNZ     Invalid
142D D200    +1 723      SETB   Configured
142F 754301  +1 724      MOV     CurrentConfiguration, #1
1432 22      +1 725      RET
1433      +1 726      Deconfigured:
1433 C200    +1 727      CLR     Configured
1435 F543    +1 728      MOV     CurrentConfiguration, A
1437 22      +1 729      RET
1438      +1 730      Get_Descriptor:                ; Host wants to know who/what we are
1438 D203    +1 731      SETB   IsDescriptor
143A E54A    +1 732      MOV     A, wValueHigh
143C 14      +1 733      DEC     A                ; Valid Values are 1, 2 and 3
143D 901478  +1 734      MOV     DPTR, #DeviceDescriptor
1440 6031    +1 735      JZ     ReturnLength
1442 14      +1 736      DEC     A
1443 90148A  +1 737      MOV     DPTR, #ConfigurationDescriptor
1446 7003    +1 738      JNZ     TryString
1448 7422    +1 739      MOV     A, #ConfigLength
144A 22      +1 740      RET
144B      +1 741      TryString:
144B 14      +1 742      DEC     A
144C 70B8    +1 743      JNZ     Invalid
           +1 744      ; Request is for a String Descriptor
144E 9014C8  +1 745      MOV     DPTR, #String0        ; Point to String 0
1451 E549    +1 746      MOV     A, wValueLow        ; Get String Index
1453      +1 747      NextString:
1453 601E    +1 748      JZ     ReturnLength
1455 FF      +1 749      MOV     R7, A                ; Save String Index
1456 71BC    +1 750      CALL   NextDPTR
1458 E0      +1 751      MOVX   A, @DPTR            ; Get the String Length (= 0 means we're at
Backsto
           p)
1459 60AB    +1 752      JZ     Invalid                ; Asked for a string I don't have
145B EF      +1 753      MOV     A, R7
145C 14      +1 754      DEC     A
145D 80F4    +1 755      JMP     NextString          ; Check if we are there yet
145F      +1 756      Get_Class_Descriptor:      ; Valid values are 21H, 22H, 23H for Class
Request
145F D203    +1 757      SETB   IsDescriptor
1461 E54A    +1 758      MOV     A, wValueHigh
1463 C3      +1 759      CLR     C
1464 9421    +1 760      SUBB   A, #21H
1466 90149C  +1 761      MOV     DPTR, #HIDDescriptor
1469 6008    +1 762      JZ     ReturnLength
146B 14      +1 763      DEC     A
146C 9014AC  +1 764      MOV     DPTR, #ReportDescriptor
146F 6004    +1 765      JZ     ReturnRDlength
           +1 766      ; DEC     A                ; This example does not use Physical
Descriptors
           +1 767      ; JZ     Send_Physical_Descriptor
1471 8093    +1 768      JMP     Invalid
           +1 769      ;
1473      +1 770      ReturnLength:
1473 E0      +1 771      MOVX   A, @DPTR            ; Get Descriptor Length (first byte)
1474 22      +1 772      RET
1475      +1 773      ReturnRDlength:          ; Report Descriptor is different format
1475 741C    +1 774      MOV     A, #ReportLength
1477 22      +1 775      RET
           +1 776      ; Error check: this MUST be on within a page of Subroutines
           +1 777      WithinSamePage EQU $ - Subroutines
           +1 778      ;
           +1 779
           +1 780
           +1 781      ;$include (DTables.A51)

```

```

+1 782 ; This module declares the descriptors
+1 783 ;
+1 784 ; This example has one Device Descriptor with:
+1 785 ;     One Configuration - single IN port and single OUT port
+1 786 ;     One Interface - there is only one method of accessing the ports
+1 787 ;     One HID Descriptor - to make PC host software simpler
+1 788 ;     One Endpoint Descriptor - for HID Input Reports
+1 789 ;     One Report Descriptor - one byte IN and one byte OUT reports
+1 790 ;     Multiple Sting Descriptors - to aid the user
+1 791 ;
---- +1 792 CSEG
+1 793 DeviceDescriptor:
1478 1201 +1 794 DB 18, 1 ; Length, Type
147A 1001 +1 795 DB 10H, 1 ; USB Rev 1.1 (=0110H, low=10H, High=01H)
147C 000000 +1 796 DB 0, 0, 0 ; Class, Subclass and Protocol
147F 40 +1 797 DB EPOSize
1480 42420142 +1 798 DB 42H, 42H, 1, 42H, 0, 1; Vendor ID, Product ID and Version
1484 0001
1486 010200 +1 799 DB 1, 2, 0 ; Manufacturer, Product & Serial# Names
1489 01 +1 800 DB 1 ; #Configs
148A +1 801 ConfigurationDescriptor:
148A 0902 +1 802 DB 9, 2 ; Length, Type
148C 2200 +1 803 DB LOW(ConfigLength), HIGH(ConfigLength)
148E 010100 +1 804 DB 1, 1, 0 ; #Interfaces, Configuration#, Config. Name
1491 A0 +1 805 DB 10100000b ; Attributes = Bus Powered
1492 FA +1 806 DB 250 ; Max. Power is 250x2 = 500mA
1493 +1 807 InterfaceDescriptor:
1493 0904 +1 808 DB 9, 4 ; Length, Type
1495 000001 +1 809 DB 0, 0, 1 ; No alternate setting, HID uses EP1
1498 03 +1 810 DB 3 ; Class = Human Interface Device
1499 0000 +1 811 DB 0, 0 ; Subclass and Protocol
149B 00 +1 812 DB 0 ; Interface Name
149C +1 813 HIDDescriptor:
149C 0921 +1 814 DB 9, 21H ; Length, Type
149E 0001 +1 815 DB 0, 1 ; HID Class Specification compliance
14A0 00 +1 816 DB 0 ; Country localization (=none)
14A1 01 +1 817 DB 1 ; Number of descriptors to follow
14A2 22 +1 818 DB 22H ; And it's a Report descriptor
14A3 1C00 +1 819 DB LOW(ReportLength), HIGH(ReportLength)
14A5 +1 820 EndpointDescriptor:
14A5 0705 +1 821 DB 7, 5 ; Length, Type
14A7 81 +1 822 DB 10000001b ; Address = IN 1
14A8 03 +1 823 DB 00000011b ; Interrupt
14A9 4000 +1 824 DB EPOSize, 0 ; Maximum packet size (this example only uses 1)
14AB 64 +1 825 DB 100 ; Poll every 0.1 seconds
0022 +1 826 ConfigLength EQU $ - ConfigurationDescriptor
+1 827
14AC +1 828 ReportDescriptor: ; Generated with HID Tool, copied to here
14AC 0600FF +1 829 DB 6, 0, 0FFH ; Usage_Page (Vendor Defined)
14AF 0901 +1 830 DB 9, 1 ; Usage (I/O Device)
14B1 A101 +1 831 DB 0A1H, 1 ; Collection (Application)
14B3 1901 +1 832 DB 19H, 1 ; Usage_Minimum (Button 1)
14B5 2908 +1 833 DB 29H, 8 ; Usage_Maximum (Button 8)
14B7 1500 +1 834 DB 15H, 0 ; Logical_Minimum (0)
14B9 2501 +1 835 DB 25H, 1 ; Logical_Maximum (1)
14BB 7501 +1 836 DB 75H, 1 ; Report_Size (1)
14BD 9508 +1 837 DB 95H, 8 ; Report_Count (8)
14BF 8102 +1 838 DB 81H, 2 ; Input (Data,Var,Abs)
14C1 1901 +1 839 DB 19H, 1 ; Usage_Minimum (Led 1)
14C3 2908 +1 840 DB 29H, 8 ; Usage_Maximum (Led 8)
14C5 9102 +1 841 DB 91H, 2 ; Output (Data,Var,Abs)
14C7 C0 +1 842 DB 0C0H ; End_Collection
001C +1 843 ReportLength EQU $-ReportDescriptor
+1 844
14C8 +1 845 String0: ; Declare the UNICODE strings
14C8 04030904 +1 846 DB 4, 3, 9, 4 ; Only English language strings supported

```

```
14CC          +1  847      String1:          ; Manufacturer
14CC 2C03     +1  848          DB      (String2-String1),3 ; Length, Type
14CE 55005300 +1  849          DB      "U",0,"S",0,"B",0," ",0,"D",0,"e",0,"s",0,"i",0,"g",0,"n",0," ",0
14D2 42002000
14D6 44006500
14DA 73006900
14DE 67006E00
14E2 2000
14E4 42007900 +1  850          DB      "B",0,"y",0," ",0,"E",0,"x",0,"a",0,"m",0,"p",0,"l",0,"e",0
14E8 20004500
14EC 78006100
14F0 6D007000
14F4 6C006500
14F8          +1  851      String2:          ; Product Name
14F8 2203     +1  852          DB      (EndOfDescriptors-String2),3
14FA 42007500 +1  853          DB      "B",0,"u",0,"t",0,"t",0,"o",0,"n",0,"s",0," ",0
14FE 74007400
1502 6F006E00
1506 73002000
150A 26002000 +1  854          DB      "&",0," ",0,"L",0,"i",0,"g",0,"h",0,"t",0,"s",0
150E 4C006900
1512 67006800
1516 74007300
151A          +1  855      EndOfDescriptors:
151A 00       +1  856          DB      0          ; Backstop for String Descriptors
          +1  857
          +1  858
          +1  859
          860
          861
          862
          863      END
```