

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

LOC	OBJ	LINE	SOURCE
		1	NAME Timer
		2	; Based on BAL Version 3.0
		3	; This program is used with MeasureLatency.cpp to measure USB interrupt latency
		4	;
		5	; Copyright (C) 2001, Intel Corporation
		6	; All rights reserved.
		7	; Permission is hereby granted to merge this program code with other program
		8	; material to create a derivative work. This derivative work may be distributed
		9	; in compiled object form only. Any other publication of this program, in any form,
		10	; without the explicit permission of the copyright holder is prohibited.
		11	;
		12	; Send questions and comments to John.Hyde@intel.com
		13	;
		14	; This version works with dScope monSIO0.hex (uses Serial Port 0, loads at 1200H)
		15	;
		16	; Changes from Version 2.0
		17	;
		18	a) Two misplaced labels corrected
		19	b) CommandTable moved outside of the constrained page
		20	c) SETUPDAT buffer copied to direct access memory, simplified coding
		21	d) Optional Set_Idle now not supported (returns STALL, not ignorred)
		22	e) R7 used in place of Temp, saved code space
		23	f) Code reorganized to separate hardware dependant sections
		24	Old New
		25	USBINT.A51 Decode.A51
		26	Vectors.A51 EZInt.A51
		27	Timer.A51 EZInt.A51
		28	Main.A51 EZMain.A51
		29	g) EP0Size made an equate to ease coding of other components
0040		29	EP0Size EQU 64 ; For EZ-USB
		30	h) Code added for descriptors > EP0Size
		31	;
		32	; Changes from Version 1.0
		33	;
		34	a) Register saving removed from Vectors.A51
		35	Main has no context which needs to be saved
		36	b) There was a race condition in USB_INT::Set_Report: which
		37	could cause OLD data to be read. Busy algorithm changed
		38	c) The USB Version# was incorrectly declared in the Device Descriptor
		39	It was 0101H and is now 0110H (data from USB IF)
		40	;
		40	;\$include (Declare.A51)
		+1	41 ; This module declares the variables and constants used in the examples
		+1	42 ; It is common to all of the examples
		+1	43 ;
		+1	44 ; Declare Special Function Registers used
0088		+1	45 TimerControl DATA 088H
0089		+1	46 TimerMode DATA 089H
008C		+1	47 Timer0High DATA 08CH
00A8		+1	48 EI DATA 0A8H
00E8		+1	49 EIE DATA 0E8H ; EZ-USB specific
0091		+1	50 EXIF DATA 091H ; EZ-USB specific
00D8		+1	51 EICON DATA 0D8H ; EZ-USB specific
0092		+1	52 PageReg DATA 092H ; EZ-USB specific, used with MOVX @Ri
0086		+1	53 DPS DATA 086H ; EZ-USB specific, used with dual data pointers
		+1	54 ;
		+1	55 ; "External" memory locations used, EZ-USB specific
		+1	56 ; Note that most of these variables are in Page 7FH
7FE8		+1	57 SETUPDAT EQU 07FE8H
7FD4		+1	58 SUDPTR EQU 07FD4H

```

7FB4      +1    59    EP0Control      EQU    07FB4H
7F00      +1    60    EP0InBuffer     EQU    07F00H
7EC0      +1    61    EP0OutBuffer    EQU    07EC0H      ; Not in Page 7FH
7E80      +1    62    EP1InBuffer     EQU    07E80H      ; Not in Page 7FH
7FB5      +1    63    IN0ByteCount    EQU    07FB5H
7FC5      +1    64    Out0ByteCount   EQU    07FC5H
7FB7      +1    65    IN1ByteCount    EQU    07FB7H
7FAC      +1    66    IN07IEN        EQU    07FACH
7FA9      +1    67    IN07IRQ        EQU    07FA9H
7FAD      +1    68    OUT07IEN       EQU    07FADH
7FAA      +1    69    OUT07IRQ       EQU    07FAAH
7FAE      +1    70    USBIEN         EQU    07FAEH
7FAB      +1    71    USBIRQ         EQU    07FABH
7FD6      +1    72    USBControl     EQU    07FD6H
7FA6      +1    73    I2CData        EQU    07FA6H
7FA5      +1    74    I2CControl     EQU    07FA5H
7F93      +1    75    PortA_Config   EQU    07F93H
7F94      +1    76    PortB_Config   EQU    07F94H
7F95      +1    77    PortC_Config   EQU    07F95H
7F96      +1    78    PortA_OUT      EQU    07F96H
7F97      +1    79    PortB_OUT      EQU    07F97H
7F98      +1    80    PortC_OUT      EQU    07F98H
7F99      +1    81    PortA_PINS     EQU    07F99H
7F9A      +1    82    PortB_PINS     EQU    07F9AH
7F9B      +1    83    PortC_PINS     EQU    07F9BH
7F9C      +1    84    PortA_OE       EQU    07F9CH
7F9D      +1    85    PortB_OE       EQU    07F9DH
7F9E      +1    86    PortC_OE       EQU    07F9EH
          +1    87    ;
          +1    88    ; Byte Variables
          +1    89
----
          +1    90
0020      +1    91    FLAGS:         DS      1      ; This register is bit-addressable
          +1    92    ; Bit Variables
          0000      +1    93    Configured     EQU    FLAGS.0 ; Is this device configured
          0001      +1    94    STALL          EQU    FLAGS.1 ; Need to STALL endpoint 0
          0002      +1    95    SendData       EQU    FLAGS.2 ; Need to send data to PC Host
          0003      +1    96    IsDescriptor   EQU    FLAGS.3 ; Enable a shortcut reply
          0004      +1    97    SetAddress     EQU    FLAGS.4 ; Set the SIE address
          +1    98    ;
0021      +1    99    MonitorSpace: DS      1FH    ; Used by Dscope
0040      +1   100    DataSpace:
0040      +1   101    ReplyCount:   DS      1      ; Byte count for following buffer
0041      +1   102    ReplyBuffer:  DS      2      ; Buffer for immediate reply
0043      +1   103    CurrentConfiguration:
0043      +1   104    DS      1      ; Some examples support > 1 configurations
0044      +1   105    SaveDPH:       DS      1      ; Needed to save Descriptor Pointer ..
0045      +1   106    SaveDPL:       DS      1      ; .. for descriptors > EP0Size
0046      +1   107    SaveLength:   DS      1      ; Number of bytes still to send
0047      +1   108    SetupData:     ; Buffer in direct access memory
0047      +1   109    RequestType:   DS      1
0048      +1   110    Request:       DS      1
0049      +1   111    wValueLow:    DS      1
004A      +1   112    wValueHigh:   DS      1
004B      +1   113    wIndexLow:    DS      1
004C      +1   114    wIndexHigh:   DS      1
004D      +1   115    wLengthLow:   DS      1
004E      +1   116    wLengthHigh:  DS      1
          +1   117    ;
004F      +1   118    CommandValue: DS      1      ; For OUT Reports from PC Host
0050      +1   119    Msec_Counter: DS      2      ; 16bit counter (LSB, MSB)
          +1   120    ;
          121
          122    ;$include (EZInt.A51)
          +1   123    ; This module contains all the EZUSB-specific hardware code
          +1   124    ; This module also contains all of the interrupt vector declarations and
    
```

```

+1 125 ; the first level interrupt servicing (register save, call subroutine,
+1 126 ; clear interrupt source, restore registers, return)
+1 127 ; Suspend and Resume are handled totally in this module
+1 128 ;
+1 129 ; A Reset sends us to Program space location 0
---- +1 130 CSEG AT 0 ; Code space
+1 131 USING 0 ; Reset forces Register Bank 0
0000 0212F1 +1 132 LJMP Reset
+1 133 ;
+1 134 ; The interrupt vector table is also located here
0043 +1 135 ORG 43H
0043 021200 +1 136 LJMP USB_ISR ; Auto Vector will replace byte 45H
+1 137
1200 +1 138 ORG 1200H ; Load above monSIO0.hex
1200 02126C +1 139 USB_ISR:LJMP SUDAV_ISR
1203 00 +1 140 DB 0 ; Pad entries to 4 bytes
1204 02125D +1 141 LJMP SOF_ISR
1207 00 +1 142 DB 0
1208 021223 +1 143 LJMP SUTOK_ISR
120B 00 +1 144 DB 0
120C 021230 +1 145 LJMP Suspend_ISR
120F 00 +1 146 DB 0
1210 02122B +1 147 LJMP USBReset_ISR
1213 00 +1 148 DB 0
1214 021223 +1 149 LJMP Reserved
1217 00 +1 150 DB 0
1218 02123F +1 151 LJMP EP0In_ISR
121B 00 +1 152 DB 0
121C 021223 +1 153 LJMP EP0Out_ISR ; Not enabled
121F 00 +1 154 DB 0
1220 021256 +1 155 LJMP EP1In_ISR
+1 156
+1 157 ; When a feature is used insert the required interrupt processing here
+1 158 ; The example use only used Endpoints 0 and 1 and also SOF for timing
1223 +1 159 Reserved:
1223 +1 160 SUTOK_ISR:
1223 +1 161 EP0Out_ISR:
1223 +1 162 Not_Used: ; Should not get any of these
1223 32 +1 163 RETI
+1 164
1224 +1 165 ClearINT2: ; Tell the hardware that we're done
1224 E591 +1 166 MOV A, EXIF
1226 C2E4 +1 167 CLR ACC.4 ; Clear the Interrupt 2 bit
1228 F591 +1 168 MOV EXIF, A
122A 22 +1 169 RET
+1 170
122B +1 171 USBReset_ISR: ; Bus has been Reset, move to DEFAULT state
122B C200 +1 172 CLR Configured
122D 5124 +1 173 CALL ClearINT2
+1 174 ; No need to clear source of interrupt
122F 32 +1 175 RETI
+1 176
1230 +1 177 Suspend_ISR: ; SIE detected an Idle bus
1230 E587 +1 178 MOV A, PCON
1232 4401 +1 179 ORL A, #1
1234 F587 +1 180 MOV PCON, A ; Go to sleep!
1236 00 +1 181 NOP
1237 00 +1 182 NOP ; Wake up here due to a USBResume
1238 00 +1 183 NOP
1239 5124 +1 184 CALL ClearINT2
123B 32 +1 185 RETI
+1 186
123C +1 187 WakeUp_ISR: ; Not using external WAKEUP in these
examples
+1 188 ; So this must be due to a USBResume
123C C2DC +1 189 CLR EICON.4 ; Clear the wakeup interrupt source
123E 32 +1 190 RETI

```

```

+1 191
123F E546 +1 192 EP0In_ISR: ; A prepared packet has been read by PC
host
123F E546 +1 193 MOV A, SaveLength ; Do I have any more data to send?
1241 6008 +1 194 JZ NoMoreToSend
1243 854483 +1 195 MOV DPH, SaveDPH ; Retrieve descriptor pointer
1246 854582 +1 196 MOV DPL, SaveDPL
1249 51B9 +1 197 CALL SendNextPieceOfDescriptor
124B +1 198 NoMoreToSend:
124B 9001A9 +1 199 MOV DPTR, #(100H OR LOW(IN07IRQ))
124E +1 200 ExitISR: ; Common exit for all ISR's
+1 201 ; On entry DPH = Interrupt ID, DPL = LOW(Interrupt Register)
124E 5124 +1 202 CALL ClearINT2
1250 747F +1 203 MOV A, #7FH ; EZ-USB I/O Register Page
1252 C583 +1 204 XCH A, DPH
1254 F0 +1 205 MOVX @DPTR, A ; Clear source of interrupt
1255 32 +1 206 RETI
+1 207
1256 +1 208 EP1In_ISR: ; PC Host has just read my Report
1256 7153 +1 209 CALL CreateInputReport ; Prepare another
1258 9002A9 +1 210 MOV DPTR, #(200H OR LOW(IN07IRQ))
125B 80F1 +1 211 JMP ExitISR
+1 212
125D +1 213 SOF_ISR: ; A Start-Of-Frame packet has been received
+1 214 ; Bump timer
125D 7850 +1 215 MOV R0, #Msec_Counter ; Need a 16bit increment
125F 7401 +1 216 MOV A, #1
1261 26 +1 217 ADD A, @R0 ; Carry may be set
1262 F6 +1 218 MOV @R0, A
1263 08 +1 219 INC R0
1264 E4 +1 220 CLR A
1265 36 +1 221 ADDC A, @R0 ; Add Carry into MSB
1266 F6 +1 222 MOV @R0, A
1267 9002AB +1 223 MOV DPTR, #(200H OR LOW(USBIRQ))
126A 80E2 +1 224 JMP ExitISR
+1 225
126C +1 226 SUDAV_ISR: ; A Setup packet has been received
126C 754600 +1 227 MOV SaveLength, #0 ; Clear any pending transactions (if any)
126F 907FE8 +1 228 MOV DPTR, #SETUPDAT ; Copy packet to direct access memory
1272 7847 +1 229 MOV R0, #SetupData
1274 7F08 +1 230 MOV R7, #8
1276 E0 +1 231 CopySD: MOVX A, @DPTR
1277 F6 +1 232 MOV @R0, A
1278 A3 +1 233 INC DPTR
1279 08 +1 234 INC R0
127A DFFA +1 235 DJNZ R7, CopySD
127C 7169 +1 236 CALL ServiceSetupPacket ; Handle the decode of the Setup packet
+1 237 ; if SetAddress { Update SIE address } // NOP on EZ-USB
+1 238 ; if STALL { Stall the endpoint }
+1 239 ; if SendData {
+1 240 ; if IsDescriptor { send DPTR->descriptor, A = length }
+1 241 ; else { send ReplyBuffer }
+1 242 ; }
127E 200126 +1 243 JB STALL, SendSTALL
1281 300216 +1 244 JNB SendData, HandShake
1284 200324 +1 245 JB IsDescriptor, LoadEP0
+1 246 ; Send data in ReplyBuffer
1287 907F01 +1 247 MOV DPTR, #EP0InBuffer+1
128A 7842 +1 248 MOV R0, #ReplyBuffer+1
128C 7F02 +1 249 MOV R7, #2 ; Copy the two byte buffer
128E E6 +1 250 CopyRB: MOV A, @R0
128F F0 +1 251 MOVX @DPTR, A
1290 1582 +1 252 DEC DPL
1292 18 +1 253 DEC R0
1293 DFF9 +1 254 DJNZ R7, CopyRB
1295 E6 +1 255 MOV A, @R0 ; Get BufferCount
1296 +1 256 SendEP0InBuffer:

```

```
1296 907FB5 +1 257          MOV    DPTR, #In0ByteCount
1299          +1 258      StartXfer:
1299 F0      +1 259          MOVX   @DPTR, A          ; This write initiates the transfer
129A          +1 260      HandShake:          ; Handshake with host
129A 7F02    +1 261          MOV    R7, #00000010b      ; Set HSNACK to tell the SIE that we're done
129C          +1 262      SetEP0Control:
129C 907FB4    +1 263          MOV    DPTR, #EP0Control
129F E0      +1 264          MOVX   A, @DPTR
12A0 4F      +1 265          ORL    A, R7
12A1 F0      +1 266          MOVX   @DPTR, A          ; We're done
12A2 9001AB  +1 267          MOV    DPTR, #(100H OR LOW(USBIRQ))
12A5 80A7    +1 268          JMP    ExitISR
12A7          +1 269      SendSTALL:          ; Invalid Request was received
12A7 7F03    +1 270          MOV    R7, #00000011b      ; Set EPOSTALL and HSNACK
12A9 80F1    +1 271          JMP    SetEP0Control
12AB          +1 272      LoadEP0:          ; Send the data pointed to by DPTR
12AB FF      +1 273          MOV    R7, A          ; Save LENGTH
+1 274          ; Need to return the smaller of "Requested Length" and "Actual Length"
+1 275          ; If "Requested Length" > 255 then use "Actual Length"
+1 276          ; There are no descriptors > 255 in this example
12AC E54E    +1 277          MOV    A, wLengthHigh
12AE 7008    +1 278          JNZ    UseActual
12B0 C3      +1 279          CLR    C
12B1 EF      +1 280          MOV    A, R7          ; Get LENGTH
12B2 954D    +1 281          SUBB   A, wLengthLow
12B4 E54D    +1 282          MOV    A, wLengthLow      ; This does not affect Carry
12B6 5001    +1 283          JNC    UsewLengthLow
12B8          +1 284      UseActual:
12B8 EF      +1 285          MOV    A, R7
12B9          +1 286      UsewLengthLow:
12B9          +1 287      SendNextPieceOfDescriptor: ; DPTR -> Descriptor to be sent
12B9 FF      +1 288          MOV    R7, A          ; Save LENGTH again
12BA 754600 +1 289          MOV    SaveLength, #0      ; Default case, overwrite if necessary
+1 290          ; Do I have more than a single packet to send?
12BD C3      +1 291          CLR    C
12BE 9440    +1 292          SUBB   A, #EP0Size
12C0 4015    +1 293          JC    SendPacket
+1 294          ; Need to send multiple packets.
+1 295          ; Calculate and save address of next packet, send next packet now
12C2 F546    +1 296          MOV    SaveLength, A      ; Send these next time
12C4 7F40    +1 297          MOV    R7, #EP0Size
12C6 C083    +1 298          PUSH   DPH          ; Save current pointer
12C8 C082    +1 299          PUSH   DPL
12CA EF      +1 300          MOV    A, R7          ; Retrieve length
12CB 71AC    +1 301          CALL   BumpDPTR
12CD 858344 +1 302          MOV    SaveDPH, DPH
12D0 858245 +1 303          MOV    SaveDPL, DPL
12D3 D082    +1 304          POP    DPL
12D5 D083    +1 305          POP    DPH
12D7          +1 306      SendPacket:
12D7 EF      +1 307          MOV    A, R7          ; Retrieve length
12D8 FE      +1 308          MOV    R6, A          ; Save length in R6 for move
12D9 7800    +1 309          MOV    R0, #LOW(EP0InBuffer) ; PageReg = 7FH = HIGH(EP0InBuffer)
12DB E0      +1 310      CopySTD:MOVX   A, @DPTR
12DC F2      +1 311          MOVX   @R0, A
12DD A3      +1 312          INC    DPTR
12DE 08      +1 313          INC    R0
12DF DEFA    +1 314          DJNZ   R6, CopySTD
12E1 EF      +1 315          MOV    A, R7          ; Retrieve LENGTH
12E2 80B2    +1 316          JMP    SendEP0InBuffer
+1 317
12E4          +1 318      GetOutputReport:          ; Wait for this, it's next on USB
12E4 907FC5 +1 319          MOV    DPTR, #Out0ByteCount ; Enable EP0OutBuffer to receive data
12E7 F0      +1 320          MOVX   @DPTR, A          ; Any value will do
12E8 907FB4 +1 321          MOV    DPTR, #EP0Control      ; Wait for valid data in EP0OutBuffer
12EB E0      +1 322      Wait40: MOVX   A, @DPTR
```

```
12EC 5408      +1  323          ANL    A, #00001000b      ; Check OUTBSY
12EE 70FB      +1  324          JNZ    Wait40
12F0 22        +1  325          RET
               +1  326
               +1  327
               +1  328
               329
               330          ;$include (EZMain.A51)
               +1  331          ; This module initializes the microcontroller then executes MAIN forever
               +1  332          ; It is hardware dependant
               +1  333
12F1          +1  334          Reset:
12F1 7581DF    +1  335          MOV    SP, #0DFH          ; Initialize the Stack
12F4 75927F    +1  336          MOV    PageReg, #7FH     ; Allows MOVX Ri to access EZ-USB memory
               +1  337
12F7 78D6      +1  338          MOV    R0, #Low(USBControl) ; Simulate a disconnect
12F9 E2        +1  339          MOVX   A, @R0
12FA 54F3      +1  340          ANL    A, #11110011b     ; Clear DISCON, DISCOE
12FC F2        +1  341          MOVX   @R0, A
12FD 713A      +1  342          CALL  Wait100msec       ; Give the host time to react
12FF E2        +1  343          MOVX   A, @R0          ; Reconnect with this new identity
1300 4406      +1  344          ORL    A, #00000110b    ; Set DISCOE to enable pullup resistor
1302 F2        +1  345          MOVX   @R0, A          ; Set RENUM so that 8051 handles USB
requests
1303 E4        +1  346          CLR    A
1304 F520      +1  347          MOV    FLAGS, A        ; Start in Default state
1306 7840      +1  348          MOV    R0, #DataSpace
1308 F6        +1  349          Clear: MOV    @R0, A        ; Clear all of available memory
1309 08        +1  350          INC    R0              ; For DEBUG
130A B8DFFB    +1  351          CJNE   R0, #0DFH, Clear
130D          +1  352          InitializeIOSystem:    ; Setup for Simmbus A=input, B=output
               +1  353          ; C=External RD#,WR#,TD0,TR0
130D 7893      +1  354          MOV    R0, #LOW(PortA_Config) ; PageReg = 7F = HIGH(PortA_Config)
130F 799C      +1  355          MOV    R1, #LOW(PortA_OE)
1311 E4        +1  356          CLR    A
1312 F2        +1  357          MOVX   @R0, A          ; No alternate functions
1313 F3        +1  358          MOVX   @R1, A          ; Enable PortA for Input
1314 08        +1  359          INC    R0              ; Point to PortB_Config
1315 09        +1  360          INC    R1              ; Point to PortB_OE
1316 F2        +1  361          MOVX   @R0, A          ; No alternate functions
1317 F4        +1  362          CPL    A              ; = 0FFH
1318 F3        +1  363          MOVX   @R1, A          ; Enable PortB for Output
1319 08        +1  364          INC    R0              ; Point to PortC_Config
131A 09        +1  365          INC    R1              ; Point to PortC_OE
131B 74C3      +1  366          MOV    A, #11000011b    ;
131D F2        +1  367          MOVX   @R0, A          ; Alternate functions on [7,6,1,0]
131E 74C2      +1  368          MOV    A, #11000010b    ;
1320 F3        +1  369          MOVX   @R1, A          ; Most alternate functions are outputs
1321          +1  370          InitializeInterruptSystem: ; First initialize the USB level
1321 7403      +1  371          MOV    A, #00000011b
1323 78AC      +1  372          MOV    R0, #LOW(IN07IEN)
1325 F2        +1  373          MOVX   @R0, A          ; Enable interrupts from EP0IN and EP1IN
1326 08        +1  374          INC    R0
1327 E4        +1  375          CLR    A
1328 F2        +1  376          MOVX   @R0, A          ; Disable interrupts from OUT Endpoints 0-7
1329 08        +1  377          INC    R0
132A 7403      +1  378          MOV    A, #00000011b
132C F2        +1  379          MOVX   @R0, A          ; Enable (Resume, Suspend,) SOF and SUDAV
INTs
132D 08        +1  380          INC    R0
132E 7401      +1  381          MOV    A, #00000001b
1330 F2        +1  382          MOVX   @R0, A          ; Enable Auto Vectoring for USB interrupts
               +1  383          ; Now enable the main level
1331 75E801    +1  384          MOV    EIE, #00000001b ; Enable INT2 = USB Interrupt (only)
1334 75A890    +1  385          MOV    EI, #10010000b  ; Enable interrupt subsystem (and Ser0 for
dScope)
               +1  386
               +1  387          ; Initialization Complete.
               +1  388          ;
```

```
1337      +1 389      MAIN:
1337 00      +1 390          NOP                                ; Not much of a main loop for this example
1338 80FD      +1 391          JMP      MAIN                    ; All actions are initiated by interrupts
              +1 392      ; We are a slave, we wait to be told what to do
              +1 393
133A      +1 394      Wait100msec:
133A 7F64      +1 395          MOV      R7, #100
133C      +1 396      Wait1msec:
133C 758600     +1 397          MOV      DPS, #0                ; A delay loop
133F 90FB50     +1 398          MOV      DPTR, #-1200           ; Select primary DPTR
1342 A3        +1 399      More:   INC      DPTR                ; 3 cycles
1343 E582      +1 400          MOV      A, DPL                ; + 2
1345 4583      +1 401          ORL      A, DPH                ; + 2
1347 70F9      +1 402          JNZ      More                  ; + 3 = 10 cycles x 1200 = 1msec
1349 DFF1      +1 403          DJNZ    R7, Wait1msec
134B 22        +1 404          RET
              +1 405
134C      +1 406      ProcessOutputReport:
              +1 407          ; A Report has just been received
              +1 408          ; The report is only one byte long in this first example
              +1 409          ; It contains a new value for the LEDs
134C 907EC0     +1 409          MOV      DPTR, #EP0OutBuffer      ; Point to the Report
134F E0        +1 410          MOVX    A, @DPTR                ; Get the Data
1350 F54F      +1 411          MOV      CommandValue, A        ; Update the local variable
1352 22        +1 412          RET
              +1 413
1353      +1 414      CreateInputReport:
              +1 415          ; Called from TIMER which detected the need
              +1 416          ; The report is the new value of the Msec_Counter
              +1 417          ; The Msec_Counter is also cleared by this routine
1353 907E80     +1 417          MOV      DPTR, #EP1InBuffer      ; Point to the buffer
1356 7850      +1 418          MOV      R0, #Msec_Counter
1358 E6        +1 419          MOV      A, @R0
1359 F0        +1 420          MOVX    @DPTR, A
135A E4        +1 421          CLR      A
135B F6        +1 422          MOV      @R0, A
135C 08        +1 423          INC      R0
135D A3        +1 424          INC      DPTR
135E E6        +1 425          MOV      A, @R0
135F F0        +1 426          MOVX    @DPTR, A        ; Update the Report
1360 E4        +1 427          CLR      A
1361 F6        +1 428          MOV      @R0, A
1362 907FB7     +1 429          MOV      DPTR, #IN1ByteCount
1365 7402      +1 430          MOV      A, #2
1367 F0        +1 431          MOVX    @DPTR, A        ; Endpoint 1 now 'armed', next IN will get
data
1368 22        +1 432          RET
              +1 433
              434
              435      ;$include (Decode.A51)
              +1 436          ; This module is common to all of the examples.
              +1 437          ; It decodes the USB Setup Packets and generates appropriate responses.
              +1 438          ; Interpretation of Reports is handled by MAIN
              +1 439          ;
              +1 440          CSEG
1369      +1 441      ServiceSetupPacket:
1369 E547      +1 442          MOV      A, RequestType
136B A2E7      +1 443          MOV      C, ACC.7                ; Bit 7 = 1 means IO device needs to send
data to P
              C Host
136D 9202      +1 444          MOV      SendData, C
136F 545C      +1 445          ANL      A, #01011100b           ; IF RequestType[6.4.3.2] = 1 THEN goto
BadRequest
1371 7035      +1 446          JNZ      BadRequest
1373 E547      +1 447          MOV      A, RequestType           ; IF RequestType[1&0] = 1 THEN goto
BadRequest
1375 A2E0      +1 448          MOV      C, ACC.0
1377 82E1      +1 449          ANL      C, ACC.1
1379 402D      +1 450          JC      BadRequest
137B 30E502     +1 451          JNB     ACC.5, NotB5        ; IF RequestType[5] = 1 THEN
RequestType[1,0] = [1,
1]
137E 7403      +1 452          MOV      A, #00000011b
```

```

1380 5403      +1  453      NotB5: ANL      A, #00000011b      ; Set CommandIndex[5,4] = RequestType[1,0]
1382 C4        +1  454          SWAP      A
1383 FF        +1  455          MOV       R7, A      ; Save HI nibble of CommandIndex
                                ; Set CommandIndex[3,0] = Request[3,0]
1384 E548      +1  457          MOV       A, Request
1386 54F0      +1  458          ANL      A, #11110000b      ; Check if Request > 15
1388 701E      +1  459          JNZ      BadRequest
138A E548      +1  460          MOV       A, Request
138C 540F      +1  461          ANL      A, #00001111b      ; Only 13 are defined today, handle in
table
138E 4F        +1  462          ORL      A, R7
                                ; goto CommandTable(CommandIndex)
138F          +1  463          ; CALL    CorrectSubroutine
CorrectSubroutine:                ; Jump to the subroutine that DPTR is
pointing to
138F 754001    +1  465          MOV       ReplyCount, #1      ; Set up a default reply
1392 754100    +1  466          MOV       ReplyBuffer, #0
1395 754200    +1  467          MOV       ReplyBuffer+1, #0
1398 C204      +1  468          CLR      SetAddress          ; Clear all flags
139A C201      +1  469          CLR      STALL
139C C203      +1  470          CLR      IsDescriptor
139E 9013B5    +1  471          MOV       DPTR, #CommandTable
13A1 71AC      +1  472          CALL    BumpDPTR            ; Point to entry
13A3 E0        +1  473          MOVX     A, @DPTR           ; Get the offset
13A4 9013F5    +1  474          MOV       DPTR, #Subroutines
13A7 73        +1  475          JMP      @A+DPTR           ; Go to the correct Subroutine
                                ; Decoded a Bad Request, STALL the Endpoint
13A8          +1  476          BadRequest:
13A8 D201      +1  478          SETB    STALL
13AA 22        +1  479          RET
                                ; Support routines
13AB          +1  480          ; Returns (DPTR + byte DPTR is pointing to)
13AB E0        +1  481          NextDPTR:
13AB          +1  482          MOVX     A, @DPTR
13AC          +1  483          BumpDPTR:
13AC 2582      +1  484          ADD     A, DPL
13AE F582      +1  485          MOV     DPL, A
13B0 5002      +1  486          JNC     Skip
13B2 0583      +1  487          INC     DPH                ; Need 16 bit arithmetic here
13B4 22        +1  488          Skip:  RET
                                ; Since the table only contains byte offsets, it is important that all these
routines are
13B5          +1  491          ; within one page (100H) of Subroutines
                                ; V3.0 - CommandTable moved outside of this one page limited space
13B5          +1  492          ; V3.0 - CommandTable moved outside of this one page limited space
CommandTable:
13B5          +1  493          ; First 16 commands are for the Device
13B5 18        +1  495          DB LOW(Device_Get_Status - Subroutines)
13B6 00        +1  496          DB LOW(Device_Clear_Feature - Subroutines)
13B7 00        +1  497          DB LOW(Invalid - Subroutines)
13B8 00        +1  498          DB LOW(Device_Set_Feature - Subroutines)
13B9 00        +1  499          DB LOW(Invalid - Subroutines)
13BA 03        +1  500          DB LOW(Set_Address - Subroutines)
13BB 33        +1  501          DB LOW(Get_Descriptor - Subroutines)
13BC 00        +1  502          DB LOW(Set_Descriptor - Subroutines)
13BD 14        +1  503          DB LOW(Get_Configuration - Subroutines)
13BE 20        +1  504          DB LOW(Set_Configuration - Subroutines)
13BF 00        +1  505          DB LOW(Invalid - Subroutines)
13C0 00        +1  506          DB LOW(Invalid - Subroutines)
13C1 00        +1  507          DB LOW(Invalid - Subroutines)
13C2 00        +1  508          DB LOW(Invalid - Subroutines)
13C3 00        +1  509          DB LOW(Invalid - Subroutines)
13C4 00        +1  510          DB LOW(Invalid - Subroutines)
13C5 1C        +1  511          ; Next 16 commands are for the Interface
13C5 1C        +1  512          DB LOW(Interface_Get_Status - Subroutines)
13C6 00        +1  513          DB LOW(Interface_Clear_Feature - Subroutines)
13C7 00        +1  514          DB LOW(Invalid - Subroutines)
13C8 00        +1  515          DB LOW(Interface_Set_Feature - Subroutines)
13C9 00        +1  516          DB LOW(Invalid - Subroutines)
13CA 00        +1  517          DB LOW(Invalid - Subroutines)
13CB 5A        +1  518          DB LOW(Get_Class_Descriptor - Subroutines)

```

```

13CC 00      +1  519      DB LOW(Set_Class_Descriptor - Subroutines)
13CD 00      +1  520      DB LOW(Invalid - Subroutines)
13CE 00      +1  521      DB LOW(Invalid - Subroutines)
13CF 00      +1  522      DB LOW(Get_Interface - Subroutines)
13D0 00      +1  523      DB LOW(Set_Interface - Subroutines)
13D1 00      +1  524      DB LOW(Invalid - Subroutines)
13D2 00      +1  525      DB LOW(Invalid - Subroutines)
13D3 00      +1  526      DB LOW(Invalid - Subroutines)
13D4 00      +1  527      DB LOW(Invalid - Subroutines)
              +1  528      ; Next 16 commands are for the Endpoint
13D5 1C      +1  529      DB LOW(Endpoint_Get_Status - Subroutines)
13D6 00      +1  530      DB LOW(Endpoint_Clear_Feature - Subroutines)
13D7 00      +1  531      DB LOW(Invalid - Subroutines)
13D8 00      +1  532      DB LOW(Endpoint_Set_Feature - Subroutines)
13D9 00      +1  533      DB LOW(Invalid - Subroutines)
13DA 00      +1  534      DB LOW(Invalid - Subroutines)
13DB 00      +1  535      DB LOW(Invalid - Subroutines)
13DC 00      +1  536      DB LOW(Invalid - Subroutines)
13DD 00      +1  537      DB LOW(Invalid - Subroutines)
13DE 00      +1  538      DB LOW(Invalid - Subroutines)
13DF 00      +1  539      DB LOW(Invalid - Subroutines)
13E0 00      +1  540      DB LOW(Invalid - Subroutines)
13E1 00      +1  541      DB LOW(Endpoint_Sync_Frame - Subroutines)
13E2 00      +1  542      DB LOW(Invalid - Subroutines)
13E3 00      +1  543      DB LOW(Invalid - Subroutines)
13E4 00      +1  544      DB LOW(Invalid - Subroutines)
              +1  545      ; Next 16 commands are Class Requests
13E5 00      +1  546      DB LOW(Invalid - Subroutines)
13E6 0D      +1  547      DB LOW(Get_Report - Subroutines)
13E7 00      +1  548      DB LOW(Get_Idle - Subroutines)
13E8 00      +1  549      DB LOW(Get_Protocol - Subroutines)
13E9 00      +1  550      DB LOW(Invalid - Subroutines)
13EA 00      +1  551      DB LOW(Invalid - Subroutines)
13EB 00      +1  552      DB LOW(Invalid - Subroutines)
13EC 00      +1  553      DB LOW(Invalid - Subroutines)
13ED 00      +1  554      DB LOW(Invalid - Subroutines)
13EE 06      +1  555      DB LOW(Set_Report - Subroutines)
13EF 00      +1  556      DB LOW(Set_Idle - Subroutines)
13F0 00      +1  557      DB LOW(Set_Protocol - Subroutines)
13F1 00      +1  558      DB LOW(Invalid - Subroutines)
13F2 00      +1  559      DB LOW(Invalid - Subroutines)
13F3 00      +1  560      DB LOW(Invalid - Subroutines)
13F4 00      +1  561      DB LOW(Invalid - Subroutines)
              +1  562
13F5         +1  563      Subroutines:
              +1  564      ;
              +1  565      ; Many requests are INVALID for this example
13F5         +1  566      Get_Protocol:           ; We are not a Boot device
13F5         +1  567      Set_Protocol:           ; We are not a Boot device
13F5         +1  568      Set_Descriptor:         ; Our Descriptors are static
13F5         +1  569      Set_Class_Descriptor:    ; Our Descriptors are static
13F5         +1  570      Set_Interface:         ; We only have one Interface
13F5         +1  571      Get_Interface:         ; We do not have an Alternate setting
13F5         +1  572      Set_Idle:             ; V3.0 Optional command, not supported
13F5         +1  573      Get_Idle:             ; V3.0 Optional command, not supported
13F5         +1  574      Device_Set_Feature:    ; We have no features that can be set or cleared
13F5         +1  575      Interface_Set_Feature: ; We have no features that can be set or cleared
13F5         +1  576      Endpoint_Set_Feature:  ; We have no features that can be set or cleared
13F5         +1  577      Endpoint_Clear_Feature: ; V3.0 We have no features that can be set or
cleared
13F5         +1  578      Device_Clear_Feature:  ; We have no features that can be set or cleared
13F5         +1  579      Interface_Clear_Feature: ; We have no features that can be set or cleared
13F5         +1  580      Endpoint_Sync_Frame:   ; We are not an Isonchronous device
              +1  581
13F5         +1  582      Invalid:                 ; Invalid Request made, STALL the Endpoint
13F5 D201    +1  583      SETB      STALL
13F7 22      +1  584      Reply:      RET

```

```

+1 585
13F8 +1 586 Set_Address: ; Set the address that the SIE will respond to
13F8 D204 +1 587 SETB SetAddress
13FA 22 +1 588 RET
+1 589
13FB +1 590 Set_Report: ; Host wants to sent us a Report.
+1 591 ; The ONLY case in this example where host sends data to us
13FB 3000F7 +1 592 JNB Configured, Invalid ; Need to be Configured to do this command
13FE 51E4 +1 593 CALL GetOutputReport ; Handled in EZUSB.A51
1400 614C +1 594 JMP ProcessOutputReport ; RETurn via this subroutine
1402 +1 595 Get_Report: ; Host wants a Report
1402 3000F0 +1 596 JNB Configured, Invalid ; Need to be Configured to do this command
1405 754142 +1 597 MOV ReplyBuffer, #42H ; Reply with a recognizable (arbitrary)
value
1408 22 +1 598 RET
1409 +1 599 Get_Configuration: ; Respond with CurrentConfiguration
1409 854341 +1 600 MOV ReplyBuffer, CurrentConfiguration
140C 22 +1 601 RET
140D +1 602 Device_Get_Status: ; Only two bits of Device Status are
defined
140D 754101 +1 603 MOV ReplyBuffer, #1 ; Bit 1=Remote Wakeup(=0), Bit 0=Self
Powered(=1)
1410 22 +1 604 RET
1411 +1 605 Interface_Get_Status: ; Interface Status is currently defined as
0
1411 +1 606 Endpoint_Get_Status:
1411 754002 +1 607 MOV ReplyCount, #2 ; Need a two byte 0 response
1414 22 +1 608 RET
1415 +1 609 Set_Configuration: ; Valid values are 0 and 1
1415 E549 +1 610 MOV A, wValueLow
1417 600A +1 611 JZ Deconfigured
1419 14 +1 612 DEC A
141A 70D9 +1 613 JNZ Invalid
141C D200 +1 614 SETB Configured
141E 754301 +1 615 MOV CurrentConfiguration, #1
1421 6153 +1 616 JMP CreateInputReport ; RETurn via CreateInputReport
1423 +1 617 Deconfigured:
1423 C200 +1 618 CLR Configured
1425 F543 +1 619 MOV CurrentConfiguration, A
1427 22 +1 620 RET
1428 +1 621 Get_Descriptor: ; Host wants to know who/what we are
1428 D203 +1 622 SETB IsDescriptor
142A E54A +1 623 MOV A, wValueHigh
142C 14 +1 624 DEC A ; Valid Values are 1, 2 and 3
142D 901468 +1 625 MOV DPTR, #DeviceDescriptor
1430 6031 +1 626 JZ ReturnLength
1432 14 +1 627 DEC A
1433 90147A +1 628 MOV DPTR, #ConfigurationDescriptor
1436 7003 +1 629 JNZ TryString
1438 7422 +1 630 MOV A, #ConfigLength
143A 22 +1 631 RET
143B +1 632 TryString:
143B 14 +1 633 DEC A
143C 70B7 +1 634 JNZ Invalid
+1 635 ; Request is for a String Descriptor
143E 9014BA +1 636 MOV DPTR, #String0 ; Point to String 0
1441 E549 +1 637 MOV A, wValueLow ; Get String Index
1443 +1 638 NextString:
1443 601E +1 639 JZ ReturnLength
1445 FF +1 640 MOV R7, A ; Save String Index
1446 71AB +1 641 CALL NextDPTR
1448 E0 +1 642 MOVX A, @DPTR ; Get the String Length (= 0 means we're at
Backsto
p)
1449 60AA +1 643 JZ Invalid ; Asked for a string I don't have
144B EF +1 644 MOV A, R7
144C 14 +1 645 DEC A
144D 80F4 +1 646 JMP NextString ; Check if we are there yet
144F +1 647 Get_Class_Descriptor: ; Valid values are 21H, 22H, 23H for Class
Request
144F D203 +1 648 SETB IsDescriptor
1451 E54A +1 649 MOV A, wValueHigh

```

```

1453 C3      +1 650          CLR      C
1454 9421    +1 651          SUBB    A, #21H
1456 90148C  +1 652          MOV     DPTR, #HIDDescriptor
1459 6008    +1 653          JZ      ReturnLength
145B 14      +1 654          DEC     A
145C 90149C  +1 655          MOV     DPTR, #ReportDescriptor
145F 6004    +1 656          JZ      ReturnRDlength
145F 6004    +1 657          ;      DEC     A          ; This example does not use Physical
Descriptors
+1 658          ;      JZ      Send_Physical_Descriptor
1461 8092    +1 659          JMP     Invalid
+1 660          ;
1463         +1 661          ReturnLength:
1463 E0      +1 662          MOVX   A, @DPTR          ; Get Descriptor Length (first byte)
1464 22      +1 663          RET
1465         +1 664          ReturnRDlength:      ; Report Descriptor is different format
1465 741E    +1 665          MOV     A, #ReportLength
1467 22      +1 666          RET
+1 667          ; Error check: this MUST be on within a page of Subroutines
0073       +1 668          WithinSamePage EQU $ - Subroutines
+1 669          ;
+1 670          ;
+1 671          ;
+1 672          ;$include (DTables.A51)
+1 673          ; This module declares the descriptors
+1 674          ;
+1 675          ; This example has one Device Descriptor with:
+1 676          ;      One Configuration - single IN port and single OUT port
+1 677          ;      One Interface - there is only one method of accessing the ports
+1 678          ;      One HID Descriptor - to make PC host software simpler
+1 679          ;      One Endpoint Descriptor - for HID Input Reports
+1 680          ;      One Report Descriptor - 2 byte IN and 1 byte OUT reports
+1 681          ;      Multiple Sting Descriptors - to aid the user
+1 682          ;
----       +1 683          CSEG
1468         +1 684          DeviceDescriptor:
1468 1201    +1 685          DB      18, 1          ; Length, Type
146A 1001    +1 686          DB      10H, 1         ; USB Rev 1.1 (=0110H, low=10H, High=01H)
146C 000000  +1 687          DB      0, 0, 0         ; Class, Subclass and Protocol
146F 40      +1 688          DB      EPOSize
1470 42420242 +1 689          DB      42H, 42H, 2, 42H, 0, 1; Vendor ID, Product ID and Version
1474 0001
1476 010200  +1 690          DB      1, 2, 0         ; Manufacturer, Product & Serial# Names
1479 01      +1 691          DB      1          ; #Configs
147A         +1 692          ConfigurationDescriptor:
147A 0902    +1 693          DB      9, 2          ; Length, Type
147C 2200    +1 694          DB      LOW(ConfigLength), HIGH(ConfigLength)
147E 010100  +1 695          DB      1, 1, 0         ; #Interfaces, Configuration#, Config. Name
1481 80      +1 696          DB      10000000b       ; Attributes = Bus Powered
1482 FA      +1 697          DB      250          ; Max. Power is 250x2 = 500mA
1483         +1 698          InterfaceDescriptor:
1483 0904    +1 699          DB      9, 4          ; Length, Type
1485 000001  +1 700          DB      0, 0, 1         ; No alternate setting, HID uses EP1
1488 03      +1 701          DB      3          ; Class = Human Interface Device
1489 0000    +1 702          DB      0, 0         ; Subclass and Protocol
148B 00      +1 703          DB      0          ; Interface Name
148C         +1 704          HIDDescriptor:
148C 0921    +1 705          DB      9, 21H         ; Length, Type
148E 0001    +1 706          DB      0, 1         ; HID Class Specification compliance
1490 00      +1 707          DB      0          ; Country localization (=none)
1491 01      +1 708          DB      1          ; Number of descriptors to follow
1492 22      +1 709          DB      22H         ; And it's a Report descriptor
1493 1E00    +1 710          DB      LOW(ReportLength), HIGH(ReportLength)
1495         +1 711          EndpointDescriptor:
1495 0705    +1 712          DB      7, 5          ; Length, Type
1497 81      +1 713          DB      10000001b      ; Address = IN 1
1498 03      +1 714          DB      00000011b     ; Interrupt

```

```

1499 4000      +1  715          DB      EPOSize, 0      ; Maximum packet size
149B 01        +1  716          DB      1              ; Poll every 1 msec (like isochronous)
      0022      +1  717      ConfigLength EQU $ - ConfigurationDescriptor
               +1  718
149C          +1  719      ReportDescriptor:      ; Generated with HID Tool, copied to here
149C 0600FF    +1  720          DB      6, 0, 0FFH      ; Usage_Page (Vendor Defined)
149F 0901      +1  721          DB      9, 1          ; Usage (I/O Device)
14A1 A101      +1  722          DB      0A1H, 1       ; Collection (Application)
14A3 1901      +1  723          DB      19H, 1       ; Usage_Minimum (Button 1)
14A5 2908      +1  724          DB      29H, 8       ; Usage_Maximum (Button 8)
14A7 1500      +1  725          DB      15H, 0       ; Logical_Minimum (0)
14A9 2501      +1  726          DB      25H, 1       ; Logical_Maximum (1)
14AB 7501      +1  727          DB      75H, 1       ; Report_Size (1)
14AD 9510      +1  728          DB      95H, 16      ; Report_Count (16)
14AF 8102      +1  729          DB      81H, 2       ; Input (Data,Var,Abs)
14B1 1901      +1  730          DB      19H, 1       ; Usage_Minimum (Led 1)
14B3 2908      +1  731          DB      29H, 8       ; Usage_Maximum (Led 8)
14B5 9508      +1  732          DB      95H, 8       ; Report_Count (8)
14B7 9102      +1  733          DB      91H, 2       ; Output (Data,Var,Abs)
14B9 C0        +1  734          DB      0C0H        ; End_Collection
      001E      +1  735      ReportLength EQU $-ReportDescriptor
               +1  736
14BA          +1  737      String0:      ; Declare the UNICODE strings
14BA 04030904  +1  738          DB      4, 3, 9, 4      ; Only English language strings supported
14BE          +1  739      String1:      ; Manufacturer
14BE 2C03      +1  740          DB      (String2-String1),3 ; Length, Type
14C0 55005300  +1  741          DB      "U",0,"S",0,"B",0," ",0,"D",0,"e",0,"s",0,"i",0,"g",0,"n",0," ",0
14C4 42002000
14C8 44006500
14CC 73006900
14D0 67006E00
14D4 2000
14D6 42007900  +1  742          DB      "B",0,"y",0," ",0,"E",0,"x",0,"a",0,"m",0,"p",0,"l",0,"e",0
14DA 20004500
14DE 78006100
14E2 6D007000
14E6 6C006500
14EA          +1  743      String2:      ; Product Name
14EA 0C03      +1  744          DB      (EndOfDescriptors-String2),3
14EC 54006900  +1  745          DB      "T",0,"i",0,"m",0,"e",0,"r",0
14F0 6D006500
14F4 7200
14F6          +1  746      EndOfDescriptors:
14F6 00        +1  747          DB      0              ; Backstop for String Descriptors
               +1  748
               +1  749
               +1  750
               751
               752
               753      END

```