

MACRO ASSEMBLER A51 V6.10

OBJECT MODULE PLACED IN .\RB.OBJ

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

```

LOC OBJ          LINE    SOURCE
                1      NAME    ReaderBoard
                2
                3      ;$include (Declare.A51)
+1             4      ; This module declares the variables and constants used in the examples
+1             5      ; It is common to all of the examples
+1             6      ;
+1             7      ; Declare Special Function Registers used
0088          +1     8      TimerControl    DATA    088H
0089          +1     9      TimerMode      DATA    089H
008C          +1    10     Timer0High     DATA    08CH
00A8          +1    11     EI              DATA    0A8H
00E8          +1    12     EIE            DATA    0E8H      ; EZ-USB specific
0091          +1    13     EXIF           DATA    091H      ; EZ-USB specific
00D8          +1    14     EICON          DATA    0D8H      ; EZ-USB specific
0092          +1    15     PageReg       DATA    092H      ; EZ-USB specific, used with MOVX @Ri
0086          +1    16     DPS            DATA    086H      ; EZ-USB specific, used with dual data pointers
+1            17     ;
+1            18     ; "External" memory locations used, EZ-USB specific
+1            19     ; Note that most of these variables are in Page 7FH
7FE8          +1    20     SETUPDAT     EQU      07FE8H
7FD4          +1    21     SUDPTR       EQU      07FD4H
7FB4          +1    22     EP0Control   EQU      07FB4H
7F00          +1    23     EP0InBuffer  EQU      07F00H
7EC0          +1    24     EP0OutBuffer EQU      07EC0H      ; Not in Page 7FH
7E80          +1    25     EP1InBuffer  EQU      07E80H      ; Not in Page 7FH
7FB5          +1    26     IN0ByteCount EQU      07FB5H
7FC5          +1    27     Out0ByteCount EQU      07FC5H
7FB7          +1    28     IN1ByteCount EQU      07FB7H
7FAC          +1    29     IN07IEN      EQU      07FACH
7FA9          +1    30     IN07IRQ      EQU      07FA9H
7FAD          +1    31     OUT07IEN     EQU      07FADH
7FAA          +1    32     OUT07IRQ      EQU      07FAAH
7FAE          +1    33     USBIEN       EQU      07FAEH
7FAB          +1    34     USBIRQ       EQU      07FABH
7FD6          +1    35     USBControl   EQU      07FD6H
7FA6          +1    36     I2CData      EQU      07FA6H
7FA5          +1    37     I2CControl   EQU      07FA5H
7F93          +1    38     PortA_Config  EQU      07F93H
7F94          +1    39     PortB_Config  EQU      07F94H
7F95          +1    40     PortC_Config  EQU      07F95H
7F96          +1    41     PortA_OUT     EQU      07F96H
7F97          +1    42     PortB_OUT     EQU      07F97H
7F98          +1    43     PortC_OUT     EQU      07F98H
7F99          +1    44     PortA_PINS    EQU      07F99H
7F9A          +1    45     PortB_PINS    EQU      07F9AH
7F9B          +1    46     PortC_PINS    EQU      07F9BH
7F9C          +1    47     PortA_OE      EQU      07F9CH
7F9D          +1    48     PortB_OE      EQU      07F9DH
7F9E          +1    49     PortC_OE      EQU      07F9EH
+1            50     ;
+1            51     ; Byte Variables
+1            52
----          +1    53     DSEG        AT 20H
0020          +1    54     FLAGS:      DS          1      ; This register is bit-addressable
+1            55     ; Bit Variables
0000          +1    56     Configured   EQU      FLAGS.0 ; Is this device configured
0001          +1    57     STALL        EQU      FLAGS.1 ; Need to STALL endpoint 0
0002          +1    58     SendData      EQU      FLAGS.2 ; Need to send data to PC Host

```

```

0003      +1  59  IsDescriptor  EQU    FLAGS.3 ; Enable a shortcut reply
          +1  60  ;
0021      +1  61  MonitorSpace: DS    1FH    ; Used by Dscope
0040      +1  62  Temp:         DS    1      ; A temporary working register
0041      +1  63  Idle_Time:   DS    1      ; The time the PC host wants us to wait
0042      +1  64  Expired_Time: DS    1      ; A downcounter for timed Reports
0043      +1  65  ReplyBuffer:  DS    3      ; First byte is Count
0046      +1  66  CurrentConfiguration:
0046      +1  67  DS            DS    1      ; Some examples support > 1 configurations
          +1  68  ;
          +1  69  ; Declare the specific variables used by each of the examples
0047      +1  70  Overlay      EQU    $
0047      +1  71  Old_Buttons: DS    1      ; Used by BAL: stores current button position
0048      +1  72  LEDstrobe:  DS    1      ; Used by BAL: strobe one LED on at a time
0049      +1  73  LEDvalue:   DS    1      ; Used by BAL: stores current LED value
004A      +1  74  Msec_Counter: DS    1      ; Used by BAL: counts up to 4 msec
          +1  75
0047      +1  76  ORG Overlay ; Overlay the variables (only one set in use at any
one tim
          e)
0047      +1  77  I2CDataByte: DS    1      ; Used by I2C: keep a local copy of data read from
I2C bus
          +1  78
0047      +1  79  ORG Overlay
0047      +1  80  LightValues: DS    6      ; Used by LP: local buffer for light brightness
004D      +1  81  WorkingValues: DS    6      ; Used by LP: counted down each half cycle
0053      +1  82  Mask:       DS    1      ; Used by LP: TurnON mask for Triacs
0004      +1  83  LastCycle   EQU    FLAGS.4 ; Used by LP: Tracks Positive & Negative Mains half
cycles
          +1  84
0047      +1  85  ORG Overlay
0047      +1  86  CurrentPosition: DS    1      ; Used by Stepper: motor has 16 stable positions
0048      +1  87  MotorControl:  DS    3      ; Used by Stepper: direction, Low(count) and
High(count)
          +1  88
0047      +1  89  ORG Overlay
0047      +1  90  LimitValues:  DS    12     ; Used by Temps: local buffer for limits
          +1  91
0047      +1  92  ORG Overlay
0047      +1  93  ButtonsValue: DS    1      ; Used by RB: buttons are read each full scan
0048      +1  94  DisplayPosition: DS    1      ; Used by RB: holds current display position
0049      +1  95  LEDBuffer:   DS    42     ; Used by RB: local buffer for reader board
          +1  96
          +1  97  ;
          +1  98
          +1  99  ;$include (Vectors.A51)
          +1 100  ; This module is common to all of the examples.
          +1 101  ; It contains all of the interrupt vector declarations and
          +1 102  ; the first level interrupt servicing (register save, call subroutine,
          +1 103  ; clear interrupt source, restore registers, return)
          +1 104  ; Suspend and Resume are handled totally in this module
          +1 105  ;
          +1 106  ; A Reset sends us to Program space location 0
----      +1 107  CSEG AT 0      ; Code space
          +1 108  USING 0      ; Reset forces Register Bank 0
0000 02035E +1 109  LJMP  Reset
          +1 110  ;
          +1 111  ; The interrupt vector table is also located here
          +1 112  ; EZ-USB has two levels of USB interrupts:
          +1 113  ; 1-the main level is described in this table (at ORG 43H)
          +1 114  ; 2-there are 21 sources of USB interrupts and these are described in USB_ISR
          +1 115  ; This means that two levels of acknowledgement and clearing will be required
          +1 116  ; LJMP INT0_ISR ; Features not used are commented out
          +1 117  ; ORG 0BH
          +1 118  ; LJMP Timer0_ISR
          +1 119  ; ORG 13H
          +1 120  ; LJMP INT1_ISR
          +1 121  ; ORG 1BH
          +1 122  ; LJMP Timer1_ISR
          +1 123  ; ORG 23H

```

```

+1 124 ; LJMP UART0_ISR
+1 125 ; ORG 2BH
+1 126 ; LJMP Timer2_ISR
+1 127 ; ORG 33H
+1 128 ; LJMP WakeUp_ISR
+1 129 ; ORG 3BH
+1 130 ; LJMP UART1_ISR
0043 +1 131 ORG 43H
0043 020100 +1 132 LJMP USB_ISR ; Auto Vector will replace byte 45H
+1 133 ; ORG 4BH
+1 134 ; LJMP I2C_ISR
+1 135 ; ORG 53H
+1 136 ; LJMP INT4_ISR
+1 137 ; ORG 5BH
+1 138 ; LJMP INT5_ISR
+1 139 ; ORG 63H
+1 140 ; LJMP INT6_ISR
+1 141
00E0 +1 142 ORG 0E0H ; Keep out of the way of dScope monitor
+1 143 ; If you are not using dScope then this memory hole
+1 144 ; may be used for useful routines.
0100 +1 145 ORG 100H
0100 02013C +1 146 USB_ISR:LJMP SUDAV_ISR
0103 00 +1 147 DB 0 ; Pad entries to 4 bytes
0104 020157 +1 148 LJMP SOF_ISR
0107 00 +1 149 DB 0
0108 020118 +1 150 LJMP SUTOK_ISR
010B 00 +1 151 DB 0
010C 020129 +1 152 LJMP Suspend_ISR
010F 00 +1 153 DB 0
0110 020120 +1 154 LJMP USBReset_ISR
0113 00 +1 155 DB 0
0114 020118 +1 156 LJMP Reserved
0117 00 +1 157 DB 0
+1 158 ; LJMP EP0In_ISR ; Endpoint Interrupts are not used in these
examples
+1 159 ; DB 0 ; Comment out features not used
+1 160 ; LJMP EP0Out_ISR
+1 161 ; DB 0
+1 162 ; LJMP EP1In_ISR
+1 163 ; DB 0
+1 164 ; LJMP EP1Out_ISR
+1 165 ; DB 0
+1 166 ; LJMP EP2In_ISR
+1 167 ; DB 0
+1 168 ; LJMP EP2Out_ISR
+1 169 ; DB 0
+1 170 ; LJMP EP3In_ISR
+1 171 ; DB 0
+1 172 ; LJMP EP3Out_ISR
+1 173 ; DB 0
+1 174 ; LJMP EP4In_ISR
+1 175 ; DB 0
+1 176 ; LJMP EP4Out_ISR
+1 177 ; DB 0
+1 178 ; LJMP EP5In_ISR
+1 179 ; DB 0
+1 180 ; LJMP EP5Out_ISR
+1 181 ; DB 0
+1 182 ; LJMP EP6In_ISR
+1 183 ; DB 0
+1 184 ; LJMP EP6Out_ISR
+1 185 ; DB 0
+1 186 ; LJMP EP7In_ISR
+1 187 ; DB 0
+1 188 ; LJMP EP7Out_ISR
+1 189 ; End of Interrupt Vector tables

```

```

+1 190
+1 191 ; When a feature is used insert the required interrupt processing here
+1 192 ; The example use only used Endpoints 0 and 1 and also SOF for timing
0118 +1 193 Reserved:
0118 +1 194 INT0_ISR:
0118 +1 195 Timer0_ISR:
0118 +1 196 INT1_ISR:
0118 +1 197 Timer1_ISR:
0118 +1 198 UART0_ISR:
0118 +1 199 Timer2_ISR:
0118 +1 200 UART1_ISR:
0118 +1 201 I2C_ISR:
0118 +1 202 INT4_ISR:
0118 +1 203 INT5_ISR:
0118 +1 204 INT6_ISR:
0118 +1 205 SUTOK_ISR:
0118 +1 206 EP0In_ISR:
0118 +1 207 EP0Out_ISR:
0118 +1 208 EP1In_ISR:
0118 +1 209 EP1Out_ISR:
0118 +1 210 EP2In_ISR:
0118 +1 211 EP2Out_ISR:
0118 +1 212 EP3In_ISR:
0118 +1 213 EP3Out_ISR:
0118 +1 214 EP4In_ISR:
0118 +1 215 EP4Out_ISR:
0118 +1 216 EP5In_ISR:
0118 +1 217 EP5Out_ISR:
0118 +1 218 EP6In_ISR:
0118 +1 219 EP6Out_ISR:
0118 +1 220 EP7In_ISR :
0118 +1 221 EP7Out_ISR:
0118 +1 222 Not_Used: ; Should not get any of these
0118 32 +1 223 RETI
+1 224
0119 +1 225 ClearINT2: ; Tell the hardware that we're done
0119 E591 +1 226 MOV A, EXIF
011B C2E4 +1 227 CLR ACC.4 ; Clear the Interrupt 2 bit
011D F591 +1 228 MOV EXIF, A
011F 22 +1 229 RET
+1 230
0120 +1 231 USBReset_ISR: ; Bus has been Reset, move to DEFAULT state
0120 C0E0 +1 232 PUSH ACC
0122 C200 +1 233 CLR Configured
0124 3119 +1 234 CALL ClearINT2
+1 235 ; No need to clear source of interrupt
0126 D0E0 +1 236 POP ACC
0128 32 +1 237 RETI
+1 238
0129 +1 239 Suspend_ISR: ; SIE detected an Idle bus
0129 C0E0 +1 240 PUSH ACC
012B E587 +1 241 MOV A, PCON
012D 4401 +1 242 ORL A, #1
012F F587 +1 243 MOV PCON, A ; Go to sleep!
0131 00 +1 244 NOP
0132 00 +1 245 NOP ; Wake up here due to a USBResume
0133 00 +1 246 NOP
0134 3119 +1 247 CALL ClearINT2
0136 D0E0 +1 248 POP ACC
0138 32 +1 249 RETI
+1 250
0139 +1 251 WakeUp_ISR: ; Not using external WAKEUP in these examples
+1 252 ; So this must be due to a USBResume
0139 C2DC +1 253 CLR EICON.4 ; Clear the wakeup interrupt source
013B 32 +1 254 RETI
+1 255

```

```

013C      +1 256      SUDAV_ISR:                ; A Setup packet has been received
013C COD0  +1 257      PUSH      PSW                ; Save Registers before the service routine
013E C0E0  +1 258      PUSH      ACC
0140 C082  +1 259      PUSH      DPL
0142 C083  +1 260      PUSH      DPH
0144 3167  +1 261      CALL      ServiceSetupPacket
0146 3119  +1 262      CALL      ClearINT2
                                ; Clear the source of the interrupt
0148 7401  +1 264      MOV        A, #00000001b
014A 907FAB +1 265      ExitISR:MOV     DPTR, #USBIRQ
014D F0     +1 266      MOVX     @DPTR, A
014E D083  +1 267      POP       DPH                ; Restore Registers
0150 D082  +1 268      POP       DPL
0152 D0E0  +1 269      POP       ACC
0154 D0D0  +1 270      POP       PSW
0156 32     +1 271      RETI
                                +1 272
0157      +1 273      SOF_ISR:                ; A Start-Of-Frame packet has been received
0157 COD0  +1 274      PUSH      PSW                ; Save Registers before the service routine
0159 C0E0  +1 275      PUSH      ACC
015B C082  +1 276      PUSH      DPL
015D C083  +1 277      PUSH      DPH
015F B1F0  +1 278      CALL      ServiceTimerRoutine
0161 3119  +1 279      CALL      ClearINT2
                                ; Clear the source of the interrupt
0163 7402  +1 281      MOV        A, #00000010b
0165 80E3  +1 282      JMP       ExitISR
                                +1 283
                                +1 284
                                +1 285
                                +1 286      ;$include (USB_INT.A51)
                                +1 287      ; This module is common to all of the examples.
                                +1 288      ; It services USB Requests from the SIE.
                                +1 289      ; Interpretation of the Output Reports is handled by MAIN
                                +1 290      ;
                                +1 291      CSEG
0167      +1 292      ServiceSetupPacket:
0167 907FE8 +1 293      MOV        DPTR, #SETUPDAT        ; Point to Setup Packet data
016A E0     +1 294      MOVX     A, @DPTR                ; Get the RequestType
016B A2E7  +1 295      MOV        C, ACC.7            ; Bit 7 = 1 means IO device needs to send
data to P
                                C Host
016D 9202  +1 296      MOV        SendData, C
016F 545C  +1 297      ANL        A, #01011100b        ; IF RequestType[6.4.3.2] = 1 THEN goto
BadRequest
0171 7050  +1 298      JNZ        BadRequest
0173 E0     +1 299      MOVX     A, @DPTR                ; IF RequestType[1&0] = 1 THEN goto
BadRequest
0174 A2E0  +1 300      MOV        C, ACC.0
0176 82E1  +1 301      ANL        C, ACC.1
0178 4049  +1 302      JC         BadRequest
017A 30E502 +1 303      JNB        ACC.5, NotB5        ; IF RequestType[5] = 1 THEN
RequestType[1,0] = [1,
1]
017D 7403  +1 304      MOV        A, #00000011b
017F 5403  +1 305      NotB5: ANL        A, #00000011b        ; Set CommandIndex[5,4] = RequestType[1,0]
0181 C4     +1 306      SWAP     A
0182 F540  +1 307      MOV        Temp, A                ; Save HI nibble of CommandIndex
                                +1 308      ; Set CommandIndex[3,0] = Request[3,0]
0184 A3     +1 309      INC        DPTR                ; Point to Request
0185 E0     +1 310      MOVX     A, @DPTR
0186 540F  +1 311      ANL        A, #00001111b        ; Only 13 are defined today, handle in
table
0188 4540  +1 312      ORL        A, Temp
018A 31D2  +1 313      CALL     CorrectSubroutine        ; goto CommandTable(CommandIndex)
                                +1 314      ; Returns STALL=1 if a stall is required
018C 200134 +1 315      JB         STALL, BadRequest
018F 300218 +1 316      JNB        SendData, HandShake
0192 200320 +1 317      JB         IsDescriptor, LoadSUDPTR; EZ-USB has a short cut for descriptors
                                +1 318      ; Send data in ReplyBuffer
0195 907F02 +1 319      MOV        DPTR, #EP0InBuffer+2

```

```

0198 7846      +1 320          MOV     R0, #ReplyBuffer+3
019A 754003    +1 321          MOV     Temp, #3                ; Copy maximum byte count
019D E6        +1 322      CopyRB: MOV     A, @R0
019E F0        +1 323          MOVX    @DPTR, A
019F 1582      +1 324          DEC     DPL
01A1 18        +1 325          DEC     R0
01A2 D540F8    +1 326          DJNZ   Temp, CopyRB
01A5 E6        +1 327          MOV     A, @R0                ; Get real byte count
01A6          +1 328      SendEP0InBuffer:
01A6 907FB5    +1 329          MOV     DPTR, #In0ByteCount
01A9          +1 330      StartXfer:
01A9 F0        +1 331          MOVX    @DPTR, A                ; This write initiates the transfer
01AA          +1 332      HandShake:                ; Handshake with host
01AA 754002    +1 333          MOV     Temp, #00000010b      ; Set HSNACK to tell the SIE that we're done
01AD          +1 334      SetEP0Control:
01AD 907FB4    +1 335          MOV     DPTR, #EP0Control
01B0 E0        +1 336          MOVX    A, @DPTR
01B1 4540      +1 337          ORL     A, Temp
01B3 F0        +1 338          MOVX    @DPTR, A
01B4 22        +1 339          RET
01B5          +1 340      LoadSUDPTR:                ; Send the data pointed to by DPTR
01B5 858240    +1 341          MOV     Temp, DPL
01B8 E583      +1 342          MOV     A, DPH
01BA 907FD4    +1 343          MOV     DPTR, #SUDPTR
01BD F0        +1 344          MOVX    @DPTR, A
01BE E540      +1 345          MOV     A, Temp
01C0 A3        +1 346          INC     DPTR
01C1 80E6      +1 347          JMP     StartXfer
01C3          +1 348      BadRequest:                ; Invalid Request was received
01C3 754003    +1 349          MOV     Temp, #00000011b      ; Set EPOSTALL and HSNACK
01C6 80E5      +1 350          JMP     SetEP0Control
01C8          +1 351
01C8 E0        +1 352      NextDPTR:                ; Returns (DPTR + byte DPTR is pointing to)
01C8          +1 353          MOVX    A, @DPTR
01C9          +1 354      BumpDPTR:                ; Returns (DPTR + ACC)
01C9 2582      +1 355          ADD     A, DPL
01CB F582      +1 356          MOV     DPL, A
01CD 5002      +1 357          JNC     Skip
01CF 0583      +1 358          INC     DPH                ; Need 16 bit arithmetic here
01D1 22        +1 359      Skip:  RET
01D2          +1 360
01D2          +1 361      CorrectSubroutine:        ; Jump to the subroutine that DPTR is
pointing to
01D2 9001F7    +1 362          MOV     DPTR, #CommandTable
01D5 31C9      +1 363          CALL   BumpDPTR                ; Point to entry
01D7 E0        +1 364          MOVX    A, @DPTR                ; Get the offset
01D8 9001F7    +1 365          MOV     DPTR, #CommandTable
01DB 31C9      +1 366          CALL   BumpDPTR                ; Get the routine address
01DD C082      +1 367          PUSH   DPL                ; Create a RETURN address on stack
01DF C083      +1 368          PUSH   DPH                ; Note: JMP @A+DPTR not used since A, DPTR
needed
01E1 7845      +1 369          MOV     R0, #ReplyBuffer+2
01E3 E4        +1 370          CLR     A
01E4 F6        +1 371          MOV     @R0, A                ; Clear ReplyBuffer
01E5 18        +1 372          DEC     R0
01E6 F6        +1 373          MOV     @R0, A
01E7 18        +1 374          DEC     R0
01E8 7601      +1 375          MOV     @R0, #1                ; Default non-descriptor reply
01EA 907FEA    +1 376          MOV     DPTR, #SETUPDAT+2      ; Point to LOW(wValue)
01ED E0        +1 377          MOVX    A, @DPTR                ; Many of the routines need these
01EE F5F0      +1 378          MOV     B, A                ; LOW(wValue) in B
01F0 A3        +1 379          INC     DPTR
01F1 E0        +1 380          MOVX    A, @DPTR                ; HIGH(wValue) in A
01F2 C201      +1 381          CLR     STALL
01F4 C203      +1 382          CLR     IsDescriptor
01F6 22        +1 383          RET                ; Go to service routine
01F6          +1 384
01F6          +1 385      ; Since the table only contains byte offsets, it is important that all these
routines are

```

```
+1 386 ; within one page (100H) of CommandTable
+1 387 ;
01F7 +1 388 CommandTable:
+1 389 ; First 16 commands are for the Device
01F7 6C +1 390 DB Device_Get_Status - CommandTable
01F8 40 +1 391 DB Device_Clear_Feature - CommandTable
01F9 40 +1 392 DB Invalid - CommandTable
01FA 40 +1 393 DB Device_Set_Feature - CommandTable
01FB 40 +1 394 DB Invalid - CommandTable
01FC 40 +1 395 DB Invalid - CommandTable ; SIE implements Device_Set_Address
01FD 80 +1 396 DB Get_Descriptor - CommandTable
01FE 40 +1 397 DB Set_Descriptor - CommandTable
01FF 69 +1 398 DB Get_Configuration - CommandTable
0200 73 +1 399 DB Set_Configuration - CommandTable
0201 40 +1 400 DB Invalid - CommandTable
0202 40 +1 401 DB Invalid - CommandTable
0203 40 +1 402 DB Invalid - CommandTable
0204 40 +1 403 DB Invalid - CommandTable
0205 40 +1 404 DB Invalid - CommandTable
0206 40 +1 405 DB Invalid - CommandTable
+1 406 ; Next 16 commands are for the Interface
0207 70 +1 407 DB Interface_Get_Status - CommandTable
0208 40 +1 408 DB Interface_Clear_Feature - CommandTable
0209 40 +1 409 DB Invalid - CommandTable
020A 40 +1 410 DB Interface_Set_Feature - CommandTable
020B 40 +1 411 DB Invalid - CommandTable
020C 40 +1 412 DB Invalid - CommandTable
020D A4 +1 413 DB Get_Class_Descriptor - CommandTable
020E 40 +1 414 DB Set_Class_Descriptor - CommandTable
020F 40 +1 415 DB Invalid - CommandTable
0210 40 +1 416 DB Invalid - CommandTable
0211 40 +1 417 DB Get_Interface - CommandTable
0212 40 +1 418 DB Set_Interface - CommandTable
0213 40 +1 419 DB Invalid - CommandTable
0214 40 +1 420 DB Invalid - CommandTable
0215 40 +1 421 DB Invalid - CommandTable
0216 40 +1 422 DB Invalid - CommandTable
+1 423 ; Next 16 commands are for the Endpoint
0217 70 +1 424 DB Endpoint_Get_Status - CommandTable
0218 42 +1 425 DB Endpoint_Clear_Feature - CommandTable
0219 40 +1 426 DB Invalid - CommandTable
021A 40 +1 427 DB Endpoint_Set_Feature - CommandTable
021B 40 +1 428 DB Invalid - CommandTable
021C 40 +1 429 DB Invalid - CommandTable
021D 40 +1 430 DB Invalid - CommandTable
021E 40 +1 431 DB Invalid - CommandTable
021F 40 +1 432 DB Invalid - CommandTable
0220 40 +1 433 DB Invalid - CommandTable
0221 40 +1 434 DB Invalid - CommandTable
0222 40 +1 435 DB Invalid - CommandTable
0223 40 +1 436 DB Endpoint_Sync_Frame - CommandTable
0224 40 +1 437 DB Invalid - CommandTable
0225 40 +1 438 DB Invalid - CommandTable
0226 40 +1 439 DB Invalid - CommandTable
+1 440 ; Next 16 commands are Class Requests
0227 40 +1 441 DB Invalid - CommandTable
0228 55 +1 442 DB Get_Report - CommandTable
0229 62 +1 443 DB Get_Idle - CommandTable
022A 40 +1 444 DB Get_Protocol - CommandTable
022B 40 +1 445 DB Invalid - CommandTable
022C 40 +1 446 DB Invalid - CommandTable
022D 40 +1 447 DB Invalid - CommandTable
022E 40 +1 448 DB Invalid - CommandTable
022F 40 +1 449 DB Invalid - CommandTable
0230 43 +1 450 DB Set_Report - CommandTable
0231 5C +1 451 DB Set_Idle - CommandTable
```

```

0232 40      +1 452          DB Set_Protocol - CommandTable
0233 40      +1 453          DB Invalid - CommandTable
0234 40      +1 454          DB Invalid - CommandTable
0235 40      +1 455          DB Invalid - CommandTable
0236 40      +1 456          DB Invalid - CommandTable
          +1 457          ;
          +1 458          ; Many requests are INVALID for this example
0237          +1 459          Get_Protocol:          ; We are not a Boot device
0237          +1 460          Set_Protocol:          ; We are not a Boot device
0237          +1 461          Set_Descriptor:        ; Our Descriptors are static
0237          +1 462          Set_Class_Descriptor:  ; Our Descriptors are static
0237          +1 463          Set_Interface:         ; We only have one Interface
0237          +1 464          Get_Interface:         ; We do not have an Alternate setting
0237          +1 465          Device_Set_Feature:    ; We have no features that can be set or cleared
0237          +1 466          Interface_Set_Feature: ; We have no features that can be set or cleared
0237          +1 467          Endpoint_Set_Feature:  ; We have no features that can be set or cleared
0237          +1 468          Device_Clear_Feature:  ; We have no features that can be set or cleared
0237          +1 469          Interface_Clear_Feature; We have no features that can be set or cleared
0237          +1 470          Endpoint_Sync_Frame:   ; We are not an Isonchronous device
          +1 471
0237          +1 472          Invalid:                ; Invalid Request made, STALL the Endpoint
0237 D201    +1 473          SETB     STALL
          +1 474          ;
0239          +1 475          Endpoint_Clear_Feature: ; We have no features that can be set or cleared
          +1 476          ;
0239 22      +1 477          Reply:  RET
          +1 478
023A          +1 479          Set_Report:            ; Host wants to sent us a Report.
          +1 480          ; The ONLY case in this example where host sends data to us
023A 3000FA  +1 481          JNB     Configured, Invalid ; Need to be Configured to do this command
023D 907FC5  +1 482          MOV     DPTR, #Out0ByteCount ; Enable EP0OutBuffer to receive data
0240 F0      +1 483          MOVX    @DPTR, A          ; Any value will do
0241 907FAA  +1 484          MOV     DPTR, #OUT07IRQ    ; Wait for valid data in EP0OutBuffer
0244 E0      +1 485          Wait4D: MOVX    A, @DPTR
0245 5401    +1 486          ANL     A, #00000001b
0247 60FB    +1 487          JZ      Wait4D
0249 F0      +1 488          MOVX    @DPTR, A          ; Clear the interrupt
024A 61C5    +1 489          JMP     ProcessOutputReport ; RETurn via this subroutine
024C          +1 490          Get_Report:            ; Host wants a Report
024C 3000E8  +1 491          JNB     Configured, Invalid ; Need to be Configured to do this command
024F 08      +1 492          INC     R0                ; Point to ReplyBuffer(1)
0250 7618    +1 493          MOV     @R0, #18H         ; Reply with a recognizable (arbitrary)
value
0252 22      +1 494          RET
0253          +1 495          Set_Idle:              ; Host wants to tell us how often we should
talk
0253 3000E1  +1 496          JNB     Configured, Invalid ; Need to be Configured to do this command
0256 F541    +1 497          MOV     Idle_Time, A
0258 22      +1 498          RET
0259          +1 499          Get_Idle:              ; Handshake with host
to do
0259 3000DB  +1 500          JNB     Configured, Invalid ; Host must have forgotten what he told us
025C 08      +1 501          INC     R0                ; Need to be Configured to do this command
025D A641    +1 502          MOV     @R0, Idle_Time      ; Point to ReplyBuffer(1)
025F 22      +1 503          RET
0260          +1 504          Get_Configuration:      ; Need to return 0 or 1
0260 300004  +1 505          JNB     Configured, Configuration0
0263          +1 506          Configuration1:        ; Same bit pattern as Device_Get_Status
0263          +1 507          Device_Get_Status:      ; Only two bits of Device Status are
defined
0263 08      +1 508          INC     R0                ; Point to ReplyBuffer(1)
0264 7601    +1 509          MOV     @R0, #1          ; Bit 1=Remote Wakeup(=0), Bit 0=Self
Powered(=1)
0266 22      +1 510          RET
0267          +1 511          Configuration0:        ; Same bit pattern as Interface_Get_Status
0267          +1 512          Interface_Get_Status:    ; Interface Status is currently defined as
0
0267          +1 513          Endpoint_Get_Status:    ;
0267 7602    +1 514          MOV     @R0, #2
0269 22      +1 515          RET
026A          +1 516          Set_Configuration:      ; Valid values are 0 and 1
026A E5F0    +1 517          MOV     A, B                ; Get LOW(wValue)

```

```
026C 6006      +1  518          JZ      Deconfigured
026E 14        +1  519          DEC     A
026F 70C6      +1  520          JNZ     Invalid
0271 D200      +1  521          SETB   Configured
0273 22        +1  522          RET
0274          +1  523      Deconfigured:
0274 C200      +1  524          CLR     Configured
0276 22        +1  525          RET
0277          +1  526      Get_Descriptor:                ; Host wants to know who/what we are
0277 D203      +1  527          SETB   IsDescriptor
0279 14        +1  528          DEC     A                ; Valid Values are 1, 2 and 3
027A 9002C1    +1  529          MOV     DPTR, #DeviceDescriptor
027D 60BA      +1  530          JZ      Reply
027F 14        +1  531          DEC     A
0280 9002D3    +1  532          MOV     DPTR, #ConfigurationDescriptor
0283 60B4      +1  533          JZ      Reply
0285 14        +1  534          DEC     A
0286 70AF      +1  535          JNZ     Invalid
              +1  536      ; Request is for a String Descriptor
0288 900312    +1  537          MOV     DPTR, #String0        ; Point to String 0
028B E5F0      +1  538          MOV     A, B                ; Get String Index
028D          +1  539      NextString:
028D 601E      +1  540          JZ      FixUpthenReply
028F F540      +1  541          MOV     Temp, A            ; Save String Index
0291 31C8      +1  542          CALL   NextDPTR
0293 E0        +1  543          MOVX   A, @DPTR            ; Get the String Length (= 0 means we're at
Backsto

0294 60A1      +1  544          JZ      Invalid                ; Asked for a string I don't have
0296 E540      +1  545          MOV     A, Temp
0298 14        +1  546          DEC     A
0299 80F2      +1  547          JMP     NextString          ; Check if we are there yet
029B          +1  548      Get_Class_Descriptor:        ; Valid values are 21H, 22H, 23H for Class
Request
029B D203      +1  549          SETB   IsDescriptor
029D C3        +1  550          CLR     C
029E 9421      +1  551          SUBB   A, #21H
02A0 9002E5    +1  552          MOV     DPTR, #HIDDescriptor
02A3 6094      +1  553          JZ      Reply
02A5 14        +1  554          DEC     A
02A6 9002F5    +1  555          MOV     DPTR, #ReportDescriptor
02A9 608E      +1  556          JZ      Reply
              +1  557      ; DEC     A                ; This example does not use Physical
Descriptors
02AB 808A      +1  558          ; JZ      Send_Physical_Descriptor
              +1  559          JMP     Invalid
              +1  560          ;
              +1  561      ; Error check: this MUST be on within a page of CommandTable
00B6          +1  562      WithinSamePage EQU $ - CommandTable
              +1  563          ;
02AD          +1  564      FixUpthenReply:                ; EZ-USB Rev D has a String Descriptor bug
              +1  565          ; Need to fill the IN0BUF (@ 7F00H) myself
02AD E0        +1  566          MOVX   A, @DPTR            ; Get the string length
02AE FF        +1  567          MOV     R7, A                ; Save counter
02AF F5F0      +1  568          MOV     B, A
02B1 7800      +1  569          MOV     R0, #LOW(EP0InBuffer) ; PageReg = 7FH = HIGH(EP0InBuffer)
02B3 F2        +1  570      CopySD: MOVX   @R0, A
02B4 08        +1  571          INC     R0
02B5 A3        +1  572          INC     DPTR
02B6 E0        +1  573          MOVX   A, @DPTR
02B7 DFFA      +1  574          DJNZ   R7, CopySD
              +1  575      ; Fixup complete, get back to the program flow
02B9 D0E0      +1  576          POP     ACC                ; Get rid of the return address
02BB D0E0      +1  577          POP     ACC
02BD E5F0      +1  578          MOV     A, B                ; Retrieve byte count
02BF 21A6      +1  579          JMP     SendEP0InBuffer
              580
              581      ;$include (DTables.A51)
              +1  582      ; This module declares the descriptors
```

```

+1 583 ;
+1 584 ; This example has one Device Descriptor with:
+1 585 ;     One Configuration - single IN port and single OUT port
+1 586 ;     One Interface - there is only one method of accessing the ports
+1 587 ;     One HID Descriptor - to make PC host software simpler
+1 588 ;     One Endpoint Descriptor - for HID Input Reports
+1 589 ;     One Report Descriptor - 40 byte IN and 40 byte OUT reports
+1 590 ;     Multiple Sting Descriptors - to aid the user
+1 591 ;
----
+1 592         CSEG
02C1 +1 593 DeviceDescriptor:
+1 594         DB      18, 1           ; Length, Type
02C3 0101 +1 595         DW      101H           ; USB Rev 1.1
02C5 000000 +1 596         DB      0, 0, 0       ; Class, Subclass and Protocol
02C8 40 +1 597         DB      64           ; EP0 size
02C9 4242 +1 598         DW      4242H, 1, 1   ; Vendor ID, Product ID and Version
02CB 0001
02CD 0001
02CF 010200 +1 599         DB      1, 2, 0       ; Manufacturer, Product & Serial# Names
02D2 01 +1 600         DB      1           ; #Configs
02D3 +1 601 ConfigurationDescriptor:
02D3 0902 +1 602         DB      9, 2           ; Length, Type
02D5 2200 +1 603         DB      LOW(ConfigLength), HIGH(ConfigLength)
02D7 010100 +1 604         DB      1, 1, 0       ; #Interfaces, Configuration#, Config. Name
02DA 80 +1 605         DB      10000000b     ; Attributes = Bus Powered
02DB 32 +1 606         DB      50           ; Max. Power is 50x2 = 100mA
02DC +1 607 InterfaceDescriptor:
02DC 0904 +1 608         DB      9, 4           ; Length, Type
02DE 000001 +1 609         DB      0, 0, 1       ; No alternate setting, HID uses EP1
02E1 03 +1 610         DB      3           ; Class = Human Interface Device
02E2 0000 +1 611         DB      0, 0       ; Subclass and Protocol
02E4 00 +1 612         DB      0           ; Interface Name
02E5 +1 613 HIDDescriptor:
02E5 0921 +1 614         DB      9, 21H        ; Length, Type
02E7 0001 +1 615         DB      0, 1           ; HID Class Specification compliance
02E9 00 +1 616         DB      0           ; Country localization (=none)
02EA 01 +1 617         DB      1           ; Number of descriptors to follow
02EB 22 +1 618         DB      22H          ; And it's a Report descriptor
02EC 1D00 +1 619         DB      LOW(ReportLength), HIGH(ReportLength)
02EE +1 620 EndpointDescriptor:
02EE 0705 +1 621         DB      7, 5           ; Length, Type
02F0 81 +1 622         DB      10000001b     ; Address = IN 1
02F1 03 +1 623         DB      00000011b    ; Interrupt
02F2 4000 +1 624         DB      64, 0        ; Maximum packet size (this example only uses 1)
02F4 64 +1 625         DB      100         ; Poll every 0.1 seconds
    0022 +1 626 ConfigLength EQU $ - ConfigurationDescriptor
+1 627
02F5 +1 628 ReportDescriptor:
02F5 0600FF +1 629         DB      6, 0, 0FFH        ; Generated with HID Tool, copied to here
02F8 0901 +1 630         DB      9, 1           ; Usage_Page (Vendor Defined)
02FA A101 +1 631         DB      0A1H, 1       ; Usage (I/O Device)
02FC 1901 +1 632         DB      19H, 1       ; Collection (Application)
02FE 2902 +1 633         DB      29H, 2       ; Usage_Minimum
0300 1500 +1 634         DB      15H, 0       ; Usage_Maximum
0302 26FF00 +1 635         DB      26H, 255, 0   ; Logical_Minimum (0)
0305 7508 +1 636         DB      75H, 8       ; Logical_Maximum (255)
0307 9528 +1 637         DB      95H, 40    ; Report_Size (8)
0309 8102 +1 638         DB      81H, 2       ; Report_Count (40)
030B 1901 +1 639         DB      19H, 1       ; Input (Data,Var,Abs) = DOTs
030D 2902 +1 640         DB      29H, 2       ; Usage_Minimum
030F 9102 +1 641         DB      91H, 2       ; Usage_Maximum
0311 C0 +1 642         DB      0C0H          ; Output (Data,Var,Abs) = Text or DOTs
    001D +1 643 ReportLength EQU $-ReportDescriptor
+1 644
0312 +1 645 String0:
0312 04030904 +1 646         DB      4, 3, 9, 4       ; Declare the UNICODE strings
; Only English language strings supported

```

```

0316      +1  647      String1:                ; Manufacturer
0316 2C03      +1  648      DB          (String2-String1),3 ; Length, Type
0318 55005300 +1  649      DB          "U",0,"S",0,"B",0," ",0,"D",0,"e",0,"s",0,"i",0,"g",0,"n",0," ",0
031C 42002000
0320 44006500
0324 73006900
0328 67006E00
032C 2000
032E 42007900 +1  650      DB          "B",0,"y",0," ",0,"E",0,"x",0,"a",0,"m",0,"p",0,"l",0,"e",0
0332 20004500
0336 78006100
033A 6D007000
033E 6C006500
0342      +1  651      String2:                ; Product Name
0342 1A03      +1  652      DB          (EndOfDescriptors-String2),3
0344 52006500 +1  653      DB          "R",0,"e",0,"a",0,"d",0,"e",0,"r",0
0348 61006400
034C 65007200
0350 20004200 +1  654      DB          " ",0,"B",0,"o",0,"a",0,"r",0,"d",0
0354 6F006100
0358 72006400
035C      +1  655      EndOfDescriptors:
035C 0000      +1  656      DW          0                ; Backstop for String Descriptors
+1  657
+1  658
+1  659
+1  660
+1  661      ;$include (Main.A51)
+1  662      ; This module initializes the microcontroller then executes MAIN forever
+1  663      ;
+1  664
035E      +1  665      Reset:
035E 7581EB +1  666      MOV          SP, #235                ; Initialize the Stack at top of internal
memory
0361 75927F +1  667      MOV          PageReg, #7FH           ; Needed to use MOVX @Ri
+1  668
0364 78D6   +1  669      MOV          R0, #LOW(USBControl)    ; Simulate a disconnect
0366 E2     +1  670      MOVX         A, @R0
0367 54F3   +1  671      ANL          A, #11110011b         ; Clear DISCON, DISCOE
0369 F2     +1  672      MOVX         @R0, A
036A 71B4   +1  673      CALL         Wait100msec           ; Give the host time to react
036C E2     +1  674      MOVX         A, @R0                ; Reconnect with this new identity
036D 4406   +1  675      ORL          A, #00000110b         ; Set DISCOE to enable pullup resistor
036F F2     +1  676      MOVX         @R0, A                ; Set RENUM so that 8051 handles USB
requests
0370 E4     +1  677      CLR          A
0371 F520   +1  678      MOV          FLAGS, A              ; Start in Default state
0373 F546   +1  679      MOV          CurrentConfiguration, A ; This example supports two configurations
0375 7848   +1  680      MOV          R0, #DisplayPosition  ; Note LEDBuffer follows DisplayPosition
0377 7F2B   +1  681      MOV          R7, #43
0379 7408   +1  682      MOV          A, #8                 ; Display a horizontal line
037B      +1  683      ClearDisplay:
037B F6     +1  684      MOV          @R0, A
037C 08     +1  685      INC          R0
037D DFFC   +1  686      DJNZ         R7, ClearDisplay
+1  687
037F      +1  688      InitializeIOSystem:                ; This example uses PortA an IN/OUT and
+1  689      ; the lower 4 bits of PortC as OUT
+1  690      ; Assume a pre-existing configuration (ie Dscope)
037F 7893   +1  691      MOV          R0, #LOW(PortA_Config) ; PageReg = 7F = HIGH(PortA_Config)
0381 E4     +1  692      CLR          A
0382 F2     +1  693      MOVX         @R0, A                ; No alternate functions
0383 799C   +1  694      MOV          R1, #LOW(PortA_OE)
0385 F4     +1  695      CPL          A                      ; = 0FFH
0386 F3     +1  696      MOVX         @R1, A                ; Enable PortA for Output
0387 7895   +1  697      MOV          R0, #LOW(PortC_Config) ; PageReg = 7F = HIGH(PortC_Config)
0389 799E   +1  698      MOV          R1, #LOW(PortC_OE)
038B E2     +1  699      MOVX         A, @R0                ; Get current configuration

```

```

038C 54F0      +1 700      ANL      A, #0F0H
038E F2        +1 701      MOVX     @R0, A                ; No alternate functions on lower nibble
038F E3        +1 702      MOVX     A, @R1              ; Get current configuration
0390 440F      +1 703      ORL      A, #0FH
0392 F3        +1 704      MOVX     @R1, A              ; Enable PortC_Bits[3:0] for Output
0393 7898      +1 705      MOV      R0, #LOW(PortC_Out)
0395 E2        +1 706      MOVX     A, @R0
0396 54F0      +1 707      ANL      A, #0F0H
0398 F2        +1 708      MOVX     @R0, A                ; Set Bits [3:0] low
                                +1 709
0399          +1 710      InitializeInterruptSystem:    ; First initialize the USB level
0399 78AC      +1 711      MOV      R0, #LOW(IN07IEN)
039B F2        +1 712      MOVX     @R0, A                ; Disable interrupts from IN Endpoints 0-7
039C 08        +1 713      INC      R0
039D F2        +1 714      MOVX     @R0, A                ; Disable interrupts from OUT Endpoints 0-7
039E 08        +1 715      INC      R0
039F 7403      +1 716      MOV      A, #00000011b
03A1 F2        +1 717      MOVX     @R0, A                ; Enable (Resume, Suspend,) SOF and SUDAV
INTs
03A2 08        +1 718      INC      R0
03A3 7401      +1 719      MOV      A, #00000001b
03A5 F2        +1 720      MOVX     @R0, A                ; Enable Auto Vectoring for USB interrupts
03A6 78AA      +1 721      MOV      R0, #LOW(OUT07IRQ)
03A8 74FF      +1 722      MOV      A, #0FFH
03AA F2        +1 723      MOVX     @R0, A                ; Clear out any pending interrupts
                                +1 724      ; Now enable the main level
03AB 75E801    +1 725      MOV      EIE, #00000001b      ; Enable INT2 = USB Interrupt (only)
03AE 75A8C0    +1 726      MOV      EI, #11000000b      ; Enable interrupt subsystem (and Ser1 for
Dscope)
                                +1 727
                                +1 728      ; Initialization Complete.
                                +1 729      ;
03B1          +1 730      MAIN:
03B1 00        +1 731      NOP                                ; Not much of a main loop for this example
03B2 80FD      +1 732      JMP      MAIN                    ; All actions are initiated by interrupts
                                +1 733      ; We are a slave, we wait to be told what to do
                                +1 734
03B4          +1 735      Wait100msec:
03B4 754064    +1 736      MOV      Temp, #100
03B7          +1 737      Wait1msec:                        ; A delay loop
03B7 90FB50    +1 738      MOV      DPTR, #-1200
03BA A3        +1 739      More:  INC      DPTR                ; 3 cycles
03BB E582      +1 740      MOV      A, DPL                  ; + 2
03BD 4583      +1 741      ORL      A, DPH                  ; + 2
03BF 70F9      +1 742      JNZ      More                    ; + 3 = 10 cycles x 1200 = 1msec
03C1 D540F3    +1 743      DJNZ     Temp, Wait1msec
03C4 22        +1 744      RET
                                +1 745
03C5          +1 746      ProcessOutputReport:            ; A Report has just been received
                                +1 747      ; The report is 7 characters long and must be translated into a dot pattern
03C5 758601    +1 748      MOV      DPS, #1                ; Select the other Data Pointer
03C8 907EC0    +1 749      MOV      DPTR, #EP0OutBuffer    ; Point to the Report
03CB 7849      +1 750      MOV      R0, #LEDBuffer
03CD 7F07      +1 751      MOV      R7, #7                  ; Have room for 7 characters
03CF E0        +1 752      ORLoop: MOVX     A, @DPTR
03D0 30E702    +1 753      JNB     ACC.7, Skip1            ; Valid Characters are 20H to 7FH
03D3 743F      +1 754      MOV      A, #3FH                ; Replace characters > 7FH with ?
03D5 C3        +1 755      Skip1: CLR      C
03D6 9420      +1 756      SUBB    A, #20H
03D8 5002      +1 757      JNC     Skip2
03DA 743C      +1 758      MOV      A, #3CH                ; Replace characters < 20H with []
03DC 75F005    +1 759      Skip2: MOV      B, #5
03DF A4        +1 760      MUL     AB                      ; B = HI((Char-20H)*5), A = LOW
03E0 0586      +1 761      INC     DPS                      ; Swap to DPTR0
03E2 900415    +1 762      MOV     DPTR, #LookupTable
03E5 2582      +1 763      ADD     A, DPL
03E7 F582      +1 764      MOV     DPL, A
03E9 E5F0      +1 765      MOV     A, B

```

```

03EB 3583      +1  766          ADDC    A, DPH
03ED F583      +1  767          MOV     DPH, A           ; DPTR now pointing into Lookup Table
03EF 7E05      +1  768          MOV     R6, #5
03F1 E0        +1  769      LTCopy: MOVX   A, @DPTR
03F2 F6        +1  770          MOV     @R0, A
03F3 A3        +1  771          INC     DPTR
03F4 08        +1  772          INC     R0
03F5 DEFA      +1  773          DJNZ   R6, LTCopy
03F7 7600      +1  774          MOV     @R0, #0         ; Put a space between characters
03F9 08        +1  775          INC     R0
03FA 0586      +1  776          INC     DPS             ; Switch to DPTR1
03FC A3        +1  777          INC     DPTR           ; Point to next character in Output Report
03FD DFD0      +1  778          DJNZ   R7, ORLoop
03FF 0586      +1  779          INC     DPS             ; Return DPTR to DPTR0
                   +1  780          ; Fall into CreateInputReport
0401           +1  781      CreateInputReport:
                   +1  782      ; The report is 40 bytes long in this example
                   +1  783      ; It contains the LED Buffer
0401 907E80    +1  784          MOV     DPTR, #EP1InBuffer ; Point to the buffer
0404 7849      +1  785          MOV     R0, #LEDBuffer
0406 7F28      +1  786          MOV     R7, #40
0408 E6        +1  787      IRLoop: MOV    A, @R0
0409 F0        +1  788          MOVX   @DPTR, A
040A A3        +1  789          INC     DPTR
040B 08        +1  790          INC     R0
040C DFFA      +1  791          DJNZ   R7, IRLoop
040E 907FB7    +1  792          MOV     DPTR, #IN1ByteCount
0411 7428      +1  793          MOV     A, #40
0413 F0        +1  794          MOVX   @DPTR, A         ; Endpoint 1 now 'armed', next IN will get
data
0414 22        +1  795          RET
                   +1  796
0415           +1  797      LookupTable:           ; Contains the 5 columns of the 7x5 display
0415 00000000  +1  798          DB     00H,00H,00H,00H,00H ; SPACE
0419 00
041A 00007D00  +1  799          DB     00H,00H,7DH,00H,00H ; !
041E 00
041F 00700070  +1  800          DB     00H,70H,00H,70H,00H ; "
0423 00
0424 147F147F  +1  801          DB     14H,7FH,14H,7FH,14H ; #
0428 14
0429 122A7F2A  +1  802          DB     12H,2AH,7FH,2AH,24H ; $
042D 24
042E 62640813  +1  803          DB     62H,64H,08H,13H,23H ; %
0432 23
0433 36495522  +1  804          DB     36H,49H,55H,22H,05H ; &
0437 05
0438 00506000  +1  805          DB     00H,50H,60H,00H,00H ; '
043C 00
043D 001C2241  +1  806          DB     00H,1CH,22H,41H,00H ; (
0441 00
0442 0041221C  +1  807          DB     00H,41H,22H,1CH,00H ; )
0446 00
0447 14083E08  +1  808          DB     14H,08H,3EH,08H,14H ; *
044B 14
044C 08083E08  +1  809          DB     08H,08H,3EH,08H,08H ; +
0450 08
0451 00050600  +1  810          DB     00H,05H,06H,00H,00H ; ,
0455 00
0456 08080808  +1  811          DB     08H,08H,08H,08H,08H ; -
045A 08
045B 00030300  +1  812          DB     00H,03H,03H,00H,00H ; .
045F 00
0460 02040810  +1  813          DB     02H,04H,08H,10H,20H ; /
0464 20
0465 3E454951  +1  814          DB     3EH,45H,49H,51H,3EH ; 0
0469 3E

```

046A	00217F01	+1	815	DB	00H,21H,7FH,01H,00H	; 1
046E	00					
046F	21434549	+1	816	DB	21H,43H,45H,49H,31H	; 2
0473	31					
0474	42415169	+1	817	DB	42H,41H,51H,69H,46H	; 3
0478	46					
0479	0C14247F	+1	818	DB	0CH,14H,24H,7FH,04H	; 4
047D	04					
047E	72515151	+1	819	DB	72H,51H,51H,51H,4EH	; 5
0482	4E					
0483	1E294949	+1	820	DB	1EH,29H,49H,49H,06H	; 6
0487	06					
0488	40474850	+1	821	DB	40H,47H,48H,50H,60H	; 7
048C	60					
048D	36494949	+1	822	DB	36H,49H,49H,49H,36H	; 8
0491	36					
0492	3049494A	+1	823	DB	30H,49H,49H,4AH,3CH	; 9
0496	3C					
0497	00333300	+1	824	DB	00H,33H,33H,00H,00H	; :
049B	00					
049C	00656600	+1	825	DB	00H,65H,66H,00H,00H	; ;
04A0	00					
04A1	00081422	+1	826	DB	00H,08H,14H,22H,41H	; <
04A5	41					
04A6	14141414	+1	827	DB	14H,14H,14H,14H,14H	; =
04AA	14					
04AB	41221408	+1	828	DB	41H,22H,14H,08H,00H	; >
04AF	00					
04B0	20404548	+1	829	DB	20H,40H,45H,48H,30H	; ?
04B4	30					
04B5	26494F41	+1	830	DB	26H,49H,4FH,41H,3EH	; @
04B9	3E					
04BA	3F484848	+1	831	DB	3FH,48H,48H,48H,3FH	; A
04BE	3F					
04BF	7F494949	+1	832	DB	7FH,49H,49H,49H,36H	; B
04C3	36					
04C4	3E414141	+1	833	DB	3EH,41H,41H,41H,22H	; C
04C8	22					
04C9	7F414122	+1	834	DB	7FH,41H,41H,22H,1CH	; D
04CD	1C					
04CE	7F494949	+1	835	DB	7FH,49H,49H,49H,41H	; E
04D2	41					
04D3	7F484848	+1	836	DB	7FH,48H,48H,48H,40H	; F
04D7	40					
04D8	3E414949	+1	837	DB	3EH,41H,49H,49H,2FH	; G
04DC	2F					
04DD	7F080808	+1	838	DB	7FH,08H,08H,08H,7FH	; H
04E1	7F					
04E2	00417F41	+1	839	DB	00H,41H,7FH,41H,00H	; I
04E6	00					
04E7	0201417E	+1	840	DB	02H,01H,41H,7EH,40H	; J
04EB	40					
04EC	7F081422	+1	841	DB	7FH,08H,14H,22H,41H	; K
04F0	41					
04F1	7F010101	+1	842	DB	7FH,01H,01H,01H,01H	; L
04F5	01					
04F6	7F201820	+1	843	DB	7FH,20H,18H,20H,7FH	; M
04FA	7F					
04FB	7F100804	+1	844	DB	7FH,10H,08H,04H,7FH	; N
04FF	7F					
0500	3E414141	+1	845	DB	3EH,41H,41H,41H,3EH	; O
0504	3E					
0505	7F484848	+1	846	DB	7FH,48H,48H,48H,30H	; P
0509	30					
050A	3E414542	+1	847	DB	3EH,41H,45H,42H,3DH	; Q
050E	3D					

050F	7F484C4A	+1	848	DB	7FH,48H,4CH,4AH,31H	; R			
0513	31								
0514	31494949	+1	849	DB	31H,49H,49H,49H,46H	; S			
0518	46								
0519	40407F40	+1	850	DB	40H,40H,7FH,40H,40H	; T			
051D	40								
051E	7E010101	+1	851	DB	7EH,01H,01H,01H,7EH	; U			
0522	7E								
0523	7C020102	+1	852	DB	7CH,02H,01H,02H,7CH	; V			
0527	7C								
0528	7E010E01	+1	853	DB	7EH,01H,0EH,01H,7EH	; W			
052C	7E								
052D	63140814	+1	854	DB	63H,14H,08H,14H,63H	; X			
0531	63								
0532	70080708	+1	855	DB	70H,08H,07H,08H,70H	; Y			
0536	70								
0537	43454951	+1	856	DB	43H,45H,49H,51H,61H	; Z			
053B	61								
053C	007F4141	+1	857	DB	00H,7FH,41H,41H,00H	; [
0540	00								
0541	7F414141	+1	858	DB	7FH,41H,41H,41H,7FH	; []			
0545	7F								
0546	0041417F	+1	859	DB	00H,41H,41H,7FH,00H	;]			
054A	00								
054B	10204020	+1	860	DB	10H,20H,40H,20H,10H	; ^			
054F	10								
0550	01010101	+1	861	DB	01H,01H,01H,01H,01H	; _			
0554	01								
0555	00402010	+1	862	DB	00H,40H,20H,10H,00H	; a	DB	02H,15H,15H,15H,0FH	
0559	00								
		+1	863						
055A	7F091111	+1	864	DB	7FH,09H,11H,11H,0EH	; b			
055E	0E								
055F	0E111111	+1	865	DB	0EH,11H,11H,11H,02H	; c			
0563	02								
0564	0E111109	+1	866	DB	0EH,11H,11H,09H,7FH	; d			
0568	7F								
0569	0E151515	+1	867	DB	0EH,15H,15H,15H,0CH	; e			
056D	0C								
056E	083F4840	+1	868	DB	08H,3FH,48H,40H,20H	; f			
0572	20								
0573	18252525	+1	869	DB	18H,25H,25H,25H,3EH	; g			
0577	3E								
0578	7F081010	+1	870	DB	7FH,08H,10H,10H,0FH	; h			
057C	0F								
057D	00115F01	+1	871	DB	00H,11H,5FH,01H,00H	; i			
0581	00								
0582	0201115E	+1	872	DB	02H,01H,11H,5EH,00H	; j			
0586	00								
0587	7F040A11	+1	873	DB	7FH,04H,0AH,11H,00H	; k			
058B	00								
058C	00417F01	+1	874	DB	00H,41H,7FH,01H,00H	; l			
0590	00								
0591	1F100C10	+1	875	DB	1FH,10H,0CH,10H,0FH	; m			
0595	0F								
0596	1F081010	+1	876	DB	1FH,08H,10H,10H,0FH	; n			
059A	0F								
059B	0E111111	+1	877	DB	0EH,11H,11H,11H,0EH	; o			
059F	0E								
05A0	1F141414	+1	878	DB	1FH,14H,14H,14H,08H	; p			
05A4	08								
05A5	08141414	+1	879	DB	08H,14H,14H,14H,1FH	; q			
05A9	1F								
05AA	1F081010	+1	880	DB	1FH,08H,10H,10H,08H	; r			
05AE	08								
05AF	09151515	+1	881	DB	09H,15H,15H,15H,02H	; s			

```

05B3 02
05B4 107E1101 +1 882 DB 10H,7EH,11H,01H,02h ; t
05B8 02
05B9 1E010101 +1 883 DB 1EH,01H,01H,01H,1FH ; u
05BD 1F
05BE 1C020102 +1 884 DB 1CH,02H,01H,02H,1CH ; v
05C2 1C
05C3 1E010601 +1 885 DB 1EH,01H,06H,01H,1EH ; w
05C7 1E
05C8 110A040A +1 886 DB 11H,0AH,04H,0AH,11H ; x
05CC 11
05CD 18050505 +1 887 DB 18H,05H,05H,05H,1EH ; y
05D1 1E
05D2 11131519 +1 888 DB 11H,13H,15H,19H,11H ; z
05D6 11
05D7 00083641 +1 889 DB 00H,08H,36H,41H,00H ; {
05DB 00
05DC 00007F00 +1 890 DB 00H,00H,7FH,00H,00H ; |
05E0 00
05E1 00413608 +1 891 DB 00H,41H,36H,08H,00H ; }
05E5 00
05E6 08082A1C +1 892 DB 08H,08H,2AH,1CH,08H ; ->
05EA 08
05EB 081C2A08 +1 893 DB 08H,1CH,2AH,08H,08H ; <-
05EF 08
+1 894
+1 895
+1 896 ;$include (Timer.A51)
+1 897 ; This module services the real time interrupt
+1 898 ;
+1 899 ; Get a Real Time interrupt every One millisecond (using SOF interrupt)
+1 900 ;
+1 901 ; Strobe the Reader Board columns every 25msec so that no flicker is evident
+1 902 ; The circuitry is shown in Figure 6-22 except that Port C is used not Port B
+1 903 ; Pin assignment of Port C is:
+1 904 ; Bit 0 = Clock
+1 905 ; Bit 1 = Data
+1 906 ; Bit 2 = Reset
+1 907 ; Bit 3 = DIR/OE (0 = Write LEDs, 1 = Read Buttons)
05F0 +1 908 ServiceTimerRoutine:
05F0 D54A37 +1 909 DJNZ Msec_counter, Done ; Only need to react every 25msec
05F3 754A19 +1 910 MOV Msec_counter, #25 ; Reinitialize
+1 911
+1 912 SetupPointers:
05F6 7898 +1 913 MOV R0, #Low(PortC_OUT) ; Will strobe the hardware here
05F8 7996 +1 914 MOV R1, #Low(PortA_OUT) ; Will output data to here
05FA 7A01 +1 915 MOV R2, #0001 ; PortC, Code for Clock Hi
+1 916
05FC E548 +1 917 MOV A, DisplayPosition ; Which column are we currently displaying
05FE B4271A +1 918 CJNE A, #39, NextColumn
0601 +1 919 ReadButtons: ; Before each full scan
0601 907F9C +1 920 MOV DPTR, #PortA_OE ; Need to flip Port A
0604 E4 +1 921 CLR A
0605 F0 +1 922 MOVX @DPTR, A ; Setup Port A for INPUT
0606 740C +1 923 MOV A, #01100b ; Set DIR and Reset = 1
0608 F2 +1 924 MOVX @R0, A ; Enable hardware to input
0609 7999 +1 925 MOV R1, #Low(PortA_Pins)
060B E3 +1 926 MOVX A, @R1 ; Get Port A data and save for a later
Report
060C F547 +1 927 MOV ButtonsValue, A ;
060E 7996 +1 928 MOV R1, #Low(PortA_OUT)
0610 7410 +1 929 MOV A, #0010H ; Set DIR and Reset = 0 and Data = 1
0612 F2 +1 930 MOVX @R0, A
0613 74FF +1 931 MOV A, #0FFH
0615 F0 +1 932 MOVX @DPTR, A ; Setup Port A for OUTPUT
0616 7A0B +1 933 MOV R2, #0011 ; PortC, Code for Data and Clock = 1
0618 7548FF +1 934 MOV DisplayPosition, #-1 ; Allow for later INC

```

```
061B      +1  935      NextColumn:
061B E4    +1  936          CLR      A          ; First clear display to prevent ghosting
061C F3    +1  937          MOVX     @R1, A       ; Write to Port A
061D EA    +1  938          MOV      A, R2       ; = 00x1, x = Data = 1 on Column 0, else 0
061E F2    +1  939          MOVX     @R0, A       ; Set clock HI to select next column
061F E4    +1  940          CLR      A          ; Set Data and Clock Low
0620 F2    +1  941          MOVX     @R0, A       ; Complete strobe of hardware
0621 0548  +1  942          INC      DisplayPosition ; Point to next row of dots
0623 E548  +1  943          MOV      A, DisplayPosition
0625 2449  +1  944          ADD      A, #LEDBuffer    ; Index into LED Buffer
0627 F8    +1  945          MOV      R0, A
0628 E6    +1  946          MOV      A, @R0       ; Get the next row of dots
0629 F3    +1  947          MOVX     @R1, A       ; Update the display
062A 22    +1  948      Done:   Ret
          +1  949
          950
          951
          952      END
```