USBC PROTOCOL

fileName: USBCurrentProjectDocE619.doc based on firmware BLE vE6.44
//***
//* ALERT: data should be encrypted before sending out from HOST to BLE DEVICE
//***
//* vE6.01 2017.04.19 1. basic, named E601 based on BLE
//* vE6.02 2017.04.21 1. reDefined HOST command
//* vE6.03 2017.05.02 1. added Queue dataStructure
//* vE6.04 2017.05.04 1. changed TX MODE as 0x80, 0x81 and 0x82
//* vE6.05 2017.05.08 1. fineTune RX and TX modes
//* vE6.06 2017.05.09 1. added RequestSingleQueueData
//* vE6.07 2017.05.12 1. added readConfiguration, eraseQueue and testMode
//* vE6.08 2017.05.17 1. added breakCurrentIndex
//* vE6.09 2017.05.31 1. added one more record for CONFIGURATION parameters
//* vE6.10 2017.06.01 1. added command 0x0E and echo 0x4E for configuration 02
//* vE6.13 2017.06.13 1. (E6.28) added echo 0x4F and echo 0x5F for all of parameters
//* vE6.14 2017.07.12 1. (E6.30) fixed some typoError
//* vE6.15 2017.07.20 1. (E6.32 notYet) added command for START TO SEND
//* vE6.16 2017.07.27 1. (E6.32) implement on command 0x0C, low current limit mode
//* vE6.17 2017.07.31 1. (E6.33) OK! Test on REAL case
//* vE6.18 2017.08.01 1. (E6.34) added OFFLINE ENABLE feature on command 0x11
//* vE6.19 2017.08.25 1. (E6.44) set default low current limit 200 mA, 20 min, enable
//***

//***

(1) COMMAND: HOST => DEVICE

//***
RX MODE (COMMAND MODE) SUMMARY:
RX mode = 0x01 // DRIVE command, 1 byte (0 : OFF, 1 : ON)
RX mode = 0x02 // SET CUTOFF TIMER VALUE command, 4 bytes (ms)
RX mode = 0x03 // SET ENABLE TIMER command, 1 byte (0 : DISABLE, 1 : ENABLE)
RX mode = 0x04 // READ QUEUE LOGGED DATA command := HOST read QueueLogged 120 records
RX mode = 0x05 // RUN FACTORY RESET command := HOST format FRAM
RX mode = 0x06 // SET SAMPLE TIME INTERVAL TO WRITE QUEUE command, 1 byte (1..12, min)
RX mode = 0x07 // READ CONFIGURATION RECORD #1 command
RX mode = 0x08 // ERASE QUEUE command
RX mode = 0x09 // RUN TEST command
RX mode = 0x0A // (autoNotify, do NOT send this command)
RX mode = 0x0B // SET HIGH CURRENT LIMIT INDEX command, 1 byte (1..50, unit: 0.1 Amp)
RX mode = 0x0C // SET LOW CURRENT LIMIT INDEX, OFFSET TIME and ENABLE command, 1 bytes
RX mode = 0x0D // (autoNotify, do NOT send this command)
RX mode = 0x0E // READ CONFIGURATION RECORD #2 (TIMERENABLE 1 Byte, TIMERVALUE 4 Bytes)
RX mode = 0x0F // (autoNotify, do NOT send this command)
RX mode = 0x11 // SET OFFLINE ENABLE for ADVERTISING PARAMETER DYNAMICALLY(echo 0x51)
//***

```
//*****************************************************************************
//* define protocol dataStructure for HOST => BLE RX:
//*****************************************************************************
//* MODE 0x01 := set DRIVE ON OFF command
//* value: 0 := ON
//*        1 := OFF
//*       AA := doNothing
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  d  r  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  1  d AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
//*****************************************************************************

//*****************************************************************************
//* MODE 0x02 := set TIME VALUE and wantedCapacity command
//* (t0..t3) value from 0x00000000 to 0xFFFFFFFF, unit: ms
//* (c) value: 0..100, unit: %
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  r tc tc tc tc  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  2 AA t3 t2 t1 t0 AA AA AA AA AA AA AA AA AA AA cs  )
Notes: (tc) in littleEndian format
buf[4..7] := tc := set CUT OFF TIME (unit: ms)
<t3 t2 t1 t0> := in LITTLE ENDIAN format (t3 := LSB, t0 := MSB)
for example := buf[4 5 6 7] := <78 56 34 12> := value 0x12345678
//*****************************************************************************

//*****************************************************************************
//* MODE 0x03 := set ENABLE or DISABLE TIMER command
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  e  r  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  3  e AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
Notes: set ENABLE or DISABLE TIMER
buf[3] := e := (1: ENABLE, 0: DISBALE)
//*****************************************************************************

//*****************************************************************************
//* MODE 0x04 := REQUEST QUEUE LOGGED DATA command
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  a  r  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  4  a AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
Notes: read QUEUE data by address value
buf[3] := a := 0xAA, read ALL of QUEUE data, total 120 records
           := 1..120, request only ONE record by this index
//*****************************************************************************

//*****************************************************************************
//* MODE 0x05 := FACTORY RESET command
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  r  r  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  5 AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
//*****************************************************************************
```

```
//*****************************************************************************
//* MODE 0x06 := SET SAMPLE TIME interval to write to QUEUE command
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m ts  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  6 ts AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
Notes: set SAMPLE TIME interval to write to QUEUE periodically
buf[3] := ts := 0xAA, using DEFAULT time interval (5 min) to write to QUEUE
             := 1..255 (without 0xAA), (unit: min) to write QUEUE
//*****************************************************************************

//*****************************************************************************
//* MODE 0x07 := READ CONFIGURATION RECORD #1
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  r  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  7 AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
//*****************************************************************************

//*****************************************************************************
//* MODE 0x08 := ERASE QUEUE ONLY (KEEP CONFIDURATION RECORD)
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  e  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  8 AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
//*****************************************************************************

//*****************************************************************************
//* MODE 0x09 := RUN TEST MODE
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  e  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  9 AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
//*****************************************************************************

//*****************************************************************************
//* MODE 0x0A := // (autoNotify, do NOT send this command)
//*****************************************************************************

//*****************************************************************************
//* MODE 0x0B := SET HIGH CURRENT INDEX
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m Hi  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  B Hi AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
Notes: set HIGH CURRENT LIMIT INDEX to protect load device
buf[3] := Hi := 01..50 (means 0.1 .. 5.0 Amp based on unit 0.1 Amp)
//*****************************************************************************

//*****************************************************************************
//* MODE 0x0C := SET LOW CURRENT LIMIT, OFF TIME and ENABLE
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m Li Lt Le  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  C Li Lt Le AA AA AA AA AA AA AA AA AA AA AA cs  )
Notes: for time interval to write to QUEUE periodically
buf[3] := Li := 0..255, (unit: 2 mA),  set LOW CURRENT LIMIT VALUE
buf[4] := Lt := 0..255, (unit: min),   set LOW OFF TIME when lower threshold current
buf[5] := Le := 0..1,   (unit: none),  set ENABLE or DISABLE LOW CURRENT LIMIT
//*****************************************************************************
```

```
//*****************************************************************************
//* MODE 0x0D := // (autoNotify, do NOT send this command)
//*****************************************************************************

//*****************************************************************************
//* MODE 0x0E := READ CONFIGURATION RECORD #2 (TIMERVALUE 4Bytes, TIMERENABLE 1Byte)
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  r  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f  E AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
//*****************************************************************************

//*****************************************************************************
//* MODE 0x0F := // (autoNotify, do NOT send this command)
//*****************************************************************************

//*****************************************************************************
//* MODE 0x11 := SET OFFLINE ENABLE
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m OE  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f 11 OE AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
Notes:
buf[3] := OE := 0..1,   (unit: none),  set ENABLE or DISABLE ON OFFLINE ADVERTISING
//*                      0: DISABLE, 1: ENABLE
//*****************************************************************************
```

//***

(2) ECHO: DEVICE => HOST

//***

TX MODE SUMMARY (ECHO MODE):
TX mode = 0x41 // echo driverEcho := DEVICE echo drive command
TX mode = 0x42 // echo setTimerValueEcho := DEVICE echo timeValue
TX mode = 0x43 // echo enableTimerEcho := DEVICE echo DRIVERENABLE
TX mode = 0x44 // echo readQueueLoggedData := DEVICE echo QueueLoggedData
TX mode = 0x45 // echo factoryRestEcho := DEVICE return to FACTORY RESET
TX mode = 0x46 // echo setSampleTimeWriteQueueEcho
TX mode = 0x47 // echo readConfiguration #1
TX mode = 0x48 // echo eraseQueue
TX mode = 0x49 // echo testMode
TX mode = 0x4A // echo periodicEcho := DEVICE send message to HOST with periodic 1 sec
TX mode = 0x4B // echo highCurrentIndex
TX mode = 0x4C // echo lowCurrentIndex
TX mode = 0x4D // echo WarnMessage(includedHighOrLowCurrentLimitWarn)
TX mode = 0x4E // echo readConfiguration #2
TX mode = 0x4F // echo DEVICE configuration #1 based on advertising mode per 2 seconds
TX mode = 0x51 // echo setOfflineEnable(echo on command 0x11)
TX mode = 0x5F // echo DEVICE configuration #2 based on advertising mode per 2 seconds

//***
define BLE TX dataStructure:
//***
//* MODE 0x41 := echo DRIVE ON OFF
byteIndex: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
byteName: h f m s v v i i c c c c t t t t R R cs t
byteValue: (f 41 s v1 v0 i1 i0 c3 c2 c1 c0 t3 t2 t1 t0 rs1 rs0 cs)
Notes: (v i c t r) in littleEndian format
buf[0] := h := header
buf[1] := f := flow
buf[2] := m := mode
buf[3] := s := status (0 := OFF, 1 := ON)
buf[4..5] := v := voltage (unit: mV), in LITTLE ENDIAN format (v1 := LSB, v0 := MSB)
for example := buf[4 5] := <34 12> := value 0x1234
buf[6..7] := i := current (unit: mA), in LITTLE ENDIAN format (i1 := LSB, i0 := MSB)
for example := buf[6 7] := <34 12> := value 0x1234
buf[8..11] := c := capacitance (unit: uAH), in LITTLE ENDIAN (c3 := LSB, c0 := MSB)
for example := buf[12 13 14 15] := <78 56 34 12> := value 0x12345678
buf[12..14] := t := timer (uint: sec), in LITTLE ENDIAN format (t3 := LSB, t0 := MSB)
buf[15..16] := R := resistance (uint: ohm), in LITTLE ENDIAN (rs1 := LSB, rs0 := MSB)
buf[18] := cs := check sum
buf[19] := T := tail
//***

```
//*****************************************************************************
//* MODE 0x42 := echo SET TIMER VALUE
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  r tn tn tn tn  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f 42 AA t3 t2 t1 t0 AA AA AA AA AA AA AA AA AA cs  )
Notes: (tn) in littleEndian format
<t3 t2 t1 t0> := in LITTLE ENDIAN format (t3 := LSB, t0 := MSB)
for example := buf[4 5 6 7] := <78 56 34 12> := value 0x12345678
//*****************************************************************************

//*****************************************************************************
//* MODE 0x43 := echo ENABLE or DISABLE TIMER
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  e  t  t  t  t  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f 43  e t3 t2 t1 t0 AA AA AA AA AA AA AA AA AA cs  )
Notes: (t) in littleEndian format
//*****************************************************************************

//*****************************************************************************
//* MODE 0x44 := echo REQUEST QUEUE LOGGED DATA
Periodical Report: (default per 100 ms)
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  a ts tn tn tn tn  v  v in in  s  d  p  r  r cs  T
byteValue:     (  f 44  a ts t3 t2 t1 t0 v1 v0 i1 i0  s  d  p  r  r cs  )
Notes: (tn v in) in littleEndian format
byteIndex [ 3] := a := the physical memory location
byteIndex [ 4] := ts := SAMPLE TIME interval for QUEUE writing
byteIndex [13] := s := 0x02 means powerOn state for APP easy to check
byteIndex [14] := d := device status
byteIndex [15] := p := pointer to the next location (JB_pwmKeep) for QUEUE writing
//*****************************************************************************

//*****************************************************************************
//* MODE 0x45 := echo FACTORY RESET
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  r  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f 45 AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
//*****************************************************************************

//*****************************************************************************
//* MODE 0x46 := echo SAMPLE TIME interval ON QUEUE WRITING
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m ts  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f 46 ts AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
Notes: echo SAMPLE TIME interval to write to QUEUE periodically
buf[3] := ts := 0xAA, using DEFAULT time interval (5 min) to write to QUEUE
buf[3] := ts := 1..255 (without 0xAA), (unit: min) to write QUEUE
//*****************************************************************************
```

```
//*****************************************************************************
//* MODE 0x47 := echo READ CONFIGURATION record #1
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  a ts tc tc tc tc  w  w  r Hi  r pQ  p  r  r cs  T
byteValue:     (  f 47  a ts t3 t2 t1 t0 w1 w0  r Hi  r pQ  p AA AA cs  )
recordIndex:              0  1  2  3  4  5  6  7  8  9 10 11
Notes:
buf[3] := a := 0x01..0x78, (1..120), point to the CURRENT location for QUEUE writing
buf[4] := ts := 1..12 (unit: min) set SAMPLE TIME interval to write to QUEUE
buf[5..8] := tc := CUT OFF TIMER value (unit: ms)
buf[9..10] := w := 0000..9999 := passWord
buf[11] := r := reserved
buf[12] := Hi := 0x01..0x32, (1..50 means 0.1..5.0 Amp,unit 0.1 Amp), High Current
buf[13] := r := reserved
buf[14] := pQ := 0xAA, just FORMATED, point to Queue
             := 0x01..0x78 (1..120), point to the NEXT location for QUEUE writing
buf[15] := p := 0x01..0x78 (1..120), point to the NEXT location for QUEUE writing
//*****************************************************************************

//*****************************************************************************
//* MODE 0x48 := echo ERASE QUEUE
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  r  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f 48 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
//*****************************************************************************

//*****************************************************************************
//* MODE 0x49 := echo TEST MODE
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  r  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f 49 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
//*****************************************************************************

//*****************************************************************************
//* MODE 0x4A := echo device status per 1 sec periodically
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  s  v  v in in  c  c  c  c tc tc tc tc  r  r cs  T
byteName:      (  f 4A  s v1 v0 i1 i0 c3 c2 c1 c0 t3 t2 t1 t0 r1 r0 cs  )
Notes:
buf[3] := s := status (0: OFF, 1: ON)
buf[4..5] := v := voltage (unit: mV)
buf[6..7] := in := current (unit: mA)
buf[8..11] := c := capacity (unit: uAH)
buf[12..15] := tc := cutOffTime (unit: ms)
buf[16..17] := r := resistance (unit: Ohm)
//*****************************************************************************

//*****************************************************************************
//* MODE 0x4B := echo BREAKCURRENT index (unit: 0.1 A)
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  b  r  r  r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f 4B  b AA AA AA AA AA AA AA AA AA AA AA AA AA AA cs  )
Notes:
buf[3] := b := 0x01..0x32 (index 01..50 means value 0.1..5.0 Amp based on unit 0.1 Amp)
//*****************************************************************************
```

```
//*****************************************************************************
//* MODE 0x4C := ECHO LOW CURRENT LIMIT INDEX, OFFSET TIME and ENABLE
byteIndex:     0  1  2  3  4  5   6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m Li Lt Le Ltc  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f 4C Li Lt Le Ltc AA AA AA AA AA AA AA AA AA AA cs  )
Notes:
Notes:
buf[3] := Li := 0..255, (unit: 2 mA),   set LOW CURRENT LIMIT VALUE
buf[4] := Lt := 0..255, (unit: min),    set LOW CURRENT LIMIT OFF TIME
buf[5] := Le := 0..1,   (unit: none),   set LOW CURRENT LIMIT ENABLE or DISABLE
buf[6] := Ltc := 0..255,   (unit: min), set LOW CURRENT LIMIT OFF TIME COUNT
//*****************************************************************************

//*****************************************************************************
//* MODE 0x4D := reserved for warning message
//*****************************************************************************

//*****************************************************************************
//* MODE 0x4D := ECHO WARNING MESSGE
byteIndex:     0  1  2  3  4  5  6  7  8  9 10  11 12 13 14 15 16 17 18 19
byteName:      h  f  m  s  v  v  i  i Hw Lw Ltc Le  r  r  r  r  r  r cs  t
byteValue:     (  f 4D  s v1 v0 i1 i0 Hw Lw Ltc Le AA AA AA AA AA AA cs  )
Notes:
buf[3] := s := status (0: OFF, 1: ON)
buf[4..5] := v := voltage (unit: mV)
buf[6..7] := i := current (unit: mA)
buf[8] := Hw := 0..1, (0: NORMAL, 1: HIGH CURRENT WARN), (unit: none)
buf[9] := Lw := 0..4,
                0: NORMAL,
                1: LOW CURRENT WARN, COUNTER DECREMENT ONE PER 1 MIN
                2: LOW CURRENT WARN, COUNTER = 0
                3: LOW CURRENT WARN, PHONE TURN OFF BY ITSELF, I = 0 Amp
                4: LOW CURRENT WARN, USER UNPLUG PHONE FROM CHARGER, I = 0 Amp
buf[10] := Ltc := 0..255, (unit: min), echo LOW CURRENT COUNT DOWN TIMER
buf[11] := Le := 0..1, (unit: none), 1 enable, 0 disable
//*****************************************************************************

//*****************************************************************************
//* MODE 0x4E := echo READ CONFIGURATION record #2
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12  13 14 15 16 17 18 19
byteName:      h  f  m  A TE TC TC TC TC Li Lt Le Ltc OE  r  r  r  r cs  T
byteValue:     (  f 4E  a te t3 t2 t1 t0 Li Lt Le Ltc OE AA AA AA AA cs  )
recordIndex:            0  1  2  3  4  5  6  7  8  9  10 11
Notes: (TC) in littleEndian format
buf[3] := A := the physical memory location
buf[4] := TE := 0x00 TIMER DISABLE, 0x01 TIMER ENABLE
buf[5..8] := TC := CUT OFF TIMER INITIAL SETTING value (unit: ms)
buf[9]  := Li  := 0..255, (unit: 2 mA), set LOW CURRENT LIMIT INDEX
buf[10] := Lt  := 0..255, (uint: min), set LOW CURRENT OFFSET TIME INDEX
buf[11] := Le  := 0..1, (0 := disable; 1 := enable), set LOW CURRENT ENABLE INDEX
buf[12] := Ltc := 0..255, (uint: min), set LOW CURRENT OFFSET TIME DOWN COUNTER INDEX
buf[13] := OE  := 0..1 (0 := disable; 1 := enable), set OFFLINE ENABLE(advertisingMode)
//*****************************************************************************
```

```
//*****************************************************************************
//* ALERT: on {advertising} echo message to HOST in two cascaded modes of 0x4F and 0x5F
//*        the followings two mode have all of parameters of device
//*        the message will be received in INTERLACE method per 2 seconds
//*        one second for mode 0x4F and one second for mode 0x5F
//*****************************************************************************
//* MODE 0x4F := echo device status per 2 sec periodically
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  s  v  v in in  c  c  c  c tc tc tc tc  r  r cs TL
byteName:      (  f 4F  s v1 v0 i1 i0 c3 c2 c1 c0 t3 t2 t1 t0 r1 r0 cs  )
Notes: (v in c tc) in littleEndian format
buf[3] := s := status (0: OFF, 1: ON)
buf[4..5] := v := voltage (unit: mV)
buf[6..7] := in := current (unit: mA)
buf[8..11] := c := capacity (unit: uAH)
buf[12..15] := tc := cutOffTime (unit: ms)
buf[16..17] := r := resistance (unit: Ohm)
//*****************************************************************************

//*****************************************************************************
//* MODE 0x51 := ECHO OFFLINE ENABLE
byteIndex:     0  1  2  3  4  5   6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m OE  r  r   r  r  r  r  r  r  r  r  r  r  r cs  t
byteValue:     (  f 51 OE AA AA  AA AA AA AA AA AA AA AA AA AA AA cs  )
Notes:
buf[3] := OE := 0..1,   (unit: none),   echo OFFLINE ENABLE on command 0x11
                0: DISABLE, 1: ENABLE
//*****************************************************************************

//*****************************************************************************
//* MODE 0x5F := echo device status per 2 sec periodically
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  e tc tc tc tc Tc Tc Tc Tc Hi ts  r  r  r  r cs TL
byteName:      (  f 5F  e t3 t2 t1 t0 T3 T2 T1 T0 Hi ts AA AA AA AA cs  )
Notes: (tc Tc) in littleEndian format
buf[3] := e := timer enable status (0: DISABLE, 1: ENABLE)
buf[4..7] := tc := cutOffTime (unit: ms)
buf[8..11] := TC := CUT OFF TIMER INITIAL SETTING value (unit: ms)
buf[12] := Hi := 01..50, highCurrentLimit (means 0.1 .. 5.0 Amp based on unit 0.1 Amp)
buf[13] := ts := 0xAA, using DEFAULT sample time interval (5 min) to write to QUEUE
             := 1..255 (without 0xAA), (unit: min) to write QUEUE
//*****************************************************************************
```

//***

(3) DATA STRUCTURE

//***

//***

(PART I#1) CONFIGURATION RECORD #1 (see command: 0x07)

(I#1) MERGED CONFIGURATION RECORD into BLE TX PACKED 20 bytes

```
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  a ts tc tc tc tc  w  w  r Hi  r pQ  p  r  r cs  T
byteValue:     (  f 47  a ts t3 t2 t1 t0 w1 w0 AA Hi AA pQ  p AA AA cs  )
recordIndex:              0  1  2  3  4  5  6  7  8  9 10 11 (on record location 0x00)
```

Notes: **default Hi = 0x32(5.0 A)**

buf[0] := h := header
buf[1] := f := flow
buf[2] := m := mode
buf[3] := a := 0x01..0x78, (1..120), point to the CURRENT location for QUEUE writing
buf[4] := ts := 1..12 (unit: min) set SAMPLE TIME interval to write to QUEUE
buf[5..8] := tc := countDown cutOffTime value (unit: ms)
buf[9..10] := w := 0000..9999 := passWord
buf[12] := Hi := 0x01..0x32, (1..50 means 0.1..5.0 Amp based on unit 0.1 Amp),
default 0x32 (5.0 Amp)
buf[13] := r := reserved
buf[14] := pQ := 0xAA, just FORMATED, point to QUEUE record location
:= 0x01..0x78 (1..120), point to the NEXT location for QUEUE writing
buf[15] := p := 0x01..0x78 (1..120), point to the NEXT location for QUEUE writing
buf[16..17] := r := reserved
buf[18] := cs := check sum
buf[19] := T := tail

//***

//***

(PART I#2) CONFIGURATION RECORD #2 (see command: 0x0E)

(I#2) MERGED CONFIGURATION RECORD #2 into BLE TX PACKED 20 bytes

```
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12  13 14 15 16 17 18 19
byteName:      h  f  m  A TE TC TC TC TC Li Lt Le Ltc OE  r  r  r  r cs  T
byteValue:     (  f 4E  a te t3 t2 t1 t0 Li Lt Le Ltc OE AA AA AA AA cs  )
recordIndex:              0  1  2  3  4  5  6  7  8  9 10 11 (on record location 0x79)
```

Notes: **default Li = 0x64(200 mA), Lt = 0x14(20 min), Le = 1(enable)**

buf[0] := h := header
buf[1] := f := flow
buf[2] := m := mode
buf[3] := A := the physical memory location
buf[4] := TE := 0x00 TIMER DISABLE, 0x01 TIMER ENABLE
buf[5..8] := TC := CUT OFF TIMER INITIAL SETTING value (unit: ms)
buf[9] := Li := 0..255, (unit: 2 mA), set LOW CURRENT LIMIT INDEX
buf[10] := Lt := 0..255, (uint: min), set LOW CURRENT OFFSET TIME INDEX
buf[11] := Le := 0..1, (0 := disable; 1 := enable), set LOW CURRENT ENABLE INDEX
buf[12] := Ltc := 0..255, (uint: min), set LOW CURRENT OFFSET TIME DOWN COUNTER INDEX
buf[13] := OE := 0..1 (0 := disable; 1 := enable), set OFFLINE ENABLE(advertisingMode)
buf[14..17] := r := reserved
buf[18] := cs := check sum
buf[19] := T := tail

//***

```
//*****************************************************************************
```

(PART II) ECHO MESSAGE per 1 Sec Automatically

```
(II) echo message
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15  16  17 18 19
byteName:      h  f  m  s  v  v in in  c  c  c  c tc tc tc tc   R   R cs  T
byteName:      (  f 4A  s v1 v0 i1 i0 c3 c2 c1 c0 t3 t2 t1 t0 rs1 rs0 cs  )
Notes:
buf[0] := h := header
buf[1] := f := flow
buf[2] := m := mode
buf[3] := s := status (0: OFF, 1: ON)
buf[4..5] := v := voltage (unit: mV)
buf[6..7] := in := current (unit: mA)
buf[8..11] := c := capacity (unit: uAH)
buf[12..15] := tc := cutOffTime (unit: ms)
buf[16..17] := R := resistance (unit: Ohm)
buf[18] := cs := check sum
buf[19] := T := tail
//*****************************************************************************

/*****************************************************************************
```

(PART III) QUEUE RECORDS (Total := 120 x 12 BYTES)

```
(II.1) QUEUE RECORD:
byteIndex:     0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
byteName:      h  f  m  a ts tn tn tn tn  v  v  i  i  s pQ  p  r  r cs  T
byteValue:     (  f  m  a ts t3 t2 t1 t0 v1 v0 i1 i0  s pQ  p  r  r cs  )
recordIndex:            0  1  2  3  4  5  6  7  8  9 10 11
buf[0] := h := header
buf[1] := f := flow
buf[2] := m := mode
buf[3] := a := record address in memory
buf[4] := ts := 1..12 (unit: min) set SAMPLE TIME interval to write to QUEUE
buf[5..8] := tn :=  now time (unit: ms)
buf[9..10] := v := voltage (unit: mV)
buf[11..12] := i := current (unit: mA)
buf[13] := s := status (0: OFF, 1: ON)
buf[14] := pQ := on 0xAA, just FORMATED, point to QUEUE record location
             := on 0x01..0x78 (1..120), point to the NEXT location for QUEUE writing
buf[15] := p := pointer to next record location (JB_pwKeep)
buf[15..16] := r := reserved
buf[18] := cs := check sum
buf[19] := T := tail
//*****************************************************************************
```

```
//*****************************************************************************
```

Symbol Summary:

```
tn := Now Time,                 (4Bytes, ms)
tc := Cut Off Time,             (4B, ms),          on {tc timeOut} cutOffDevice
ts := Sample Time,              (1B, min, 1..12)   on {ts timeOut} write data to QUEUE
TE := TIMER ENABLE,             (1B, 0 := DISBALE, 1 := ENABLE)
TC := TIMER CUTOFF INITIAL VALUE(4B, ms)
in := Now Current,              (2B, mA)
Hi := High Current Limit Value Index,  (1B, 0.1 A, 1..50),on {in >= ib} cutOffDevice
Li := Low Current Limit Value Index,   (1B, mA, 0..255)
Lt := Low Current Limit Time Index,    (1B, min, 0..255)
Le := Low Current Limit Enable Index,  (1B, none, 0..1)
Ltc := Low Current Limit Time Down Counter Index,   (1B, min, 0..255)
OE := Offline Enable (1B, 0 := DIABLE, 1 := ENABLE)
//*****************************************************************************
```