Manual

Six transmitter data communication

If you are using our evaluation software "bioMON" this document is not relevant for you.

You may download "bioMON" as a Windows installer package at **www.jobst-technologies.com/download/software/bioMON4_Setup.zip** and get advise for installation and operation watching the video tutorials available at **www.jobst-technologies.com/support-2/biomon-video-tutorials/.**

You will have either the regular version with an USB cable or the OEM version without housing and cable. For the USB version you may download the driver from: **www.prolific.com.tw/US/ShowProduct.aspx?p_id=225&pcid=41** that provides you a virtual COM port.

For the OEM version connect ground and supply voltage (3.3-5.5 V) to your power supply. Connect the Read data (Rx) pin of your microprocessor to the Send data (Tx) pin of the Six transmitter. The pin assignment can be found in datasheet PDD0113 (www.jobst-technologies.com/support-2/downloads).
There is no need to connect the Rx pin of the Six transmitter as the data is sent by the transmitter automatically every 1.7 seconds.

For the virtual serial port via USB interface or the TTL serial port (UART) with OEM version use port settings of 9600 baud, no parity, 8 data bits, and 1 stop bit (9600,n,8,1).

Having opened the port and powered up the Six transmitter it will send the data telegrams that you need to convert according to the following pages into the reading of the six sensor channels and the temperature.

Disclaimer: Not For Medical, Diagnostics And Usage On Humans. For Evaluation Use Only.

If you have an B.LV5 glucose & lactate sensor from us you may convert the readings into concentrations by:

- from channel 2 (glucose electrode 1) subtract channel 1 (blank electrode 1)
- from channel 3 (lactate electrode 1) subtract channel 1 (blank electrode 1)
- from channel 5 (glucose electrode 2) subtract channel 4 (blank electrode 2)
- from channel 6 (lactate electrode 2) subtract channel 4 (blank electrode 2)

You should have received factory calibration data from us as Definition text files like “216 1C.def”

Its content shows like this (line numbers 1-7 added here for description):

1. .0763,**.1391,.0617**,.0763,**.1419,.0594**,1,1,100,100,100,100,100,100,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
2. 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
3. -1,1,1,-1,1,1,18,0,17,17,17,17,17,17,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
4. **"Blank1","Glucose1","Lactate1","Blank 2","Glucose 2","Lactate 2","Temperature"**
5. 0,**1,1**,0,**4,4**,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
6. 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
7. 0,**3.8,3.2**,0,**3.8,3.2**,0,**32**,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

The relevant numbers for you are highlighted in bold.

Line 4 holds the configuration of the chip you are using. In the example electrode 1 is blank sensor, etc.

Line 5 the channels to subtract. For chn1 none. For chn2 use chn1, etc.

Line 1 holds the actual calibration data. First entry 0.0763 converts chn1 readings into nA while the second entry is used to convert the difference of the AD counts of chn2-chn1 into mM of glucose, etc.

Line 7 holds the temperature sensitivity factors (e.g. **3.8 %/degC** for chn2 = glucose 1) and the reference temperature (here **32** degC).

So using a 25nA range Six the concentrations in **mM** calculate as:

glucose sensor 1 = **(Chn2-Chn1) * 0.1391 / exp(0.038 * (Temperature – 32))**

lactate sensor 1 = **(Chn3-Chn1) * 0.0617 / exp(0.032 * (Temperature – 32))**

glucose sensor 1 = **(Chn5-Chn4) * 0.1419 / exp(0.038 * (Temperature – 32))**

lactate sensor 1 = **(Chn6-Chn4) * 0.0594 / exp(0.032 * (Temperature – 32))**

While using a 50nA range Six (info on range found on the backside label) use all values *2 yielding:

glucose sensor 1 = **(Chn2-Chn1) * 0.2782 / exp(0.038 * (Temperature – 32))** etc.

(the basic code example on last page just calculates the current in nA by: Chn = 25 * AD counts /2^15)

Disclaimer: Not For Medical, Diagnostics And Usage On Humans. For Evaluation Use Only.

Serial Protocol

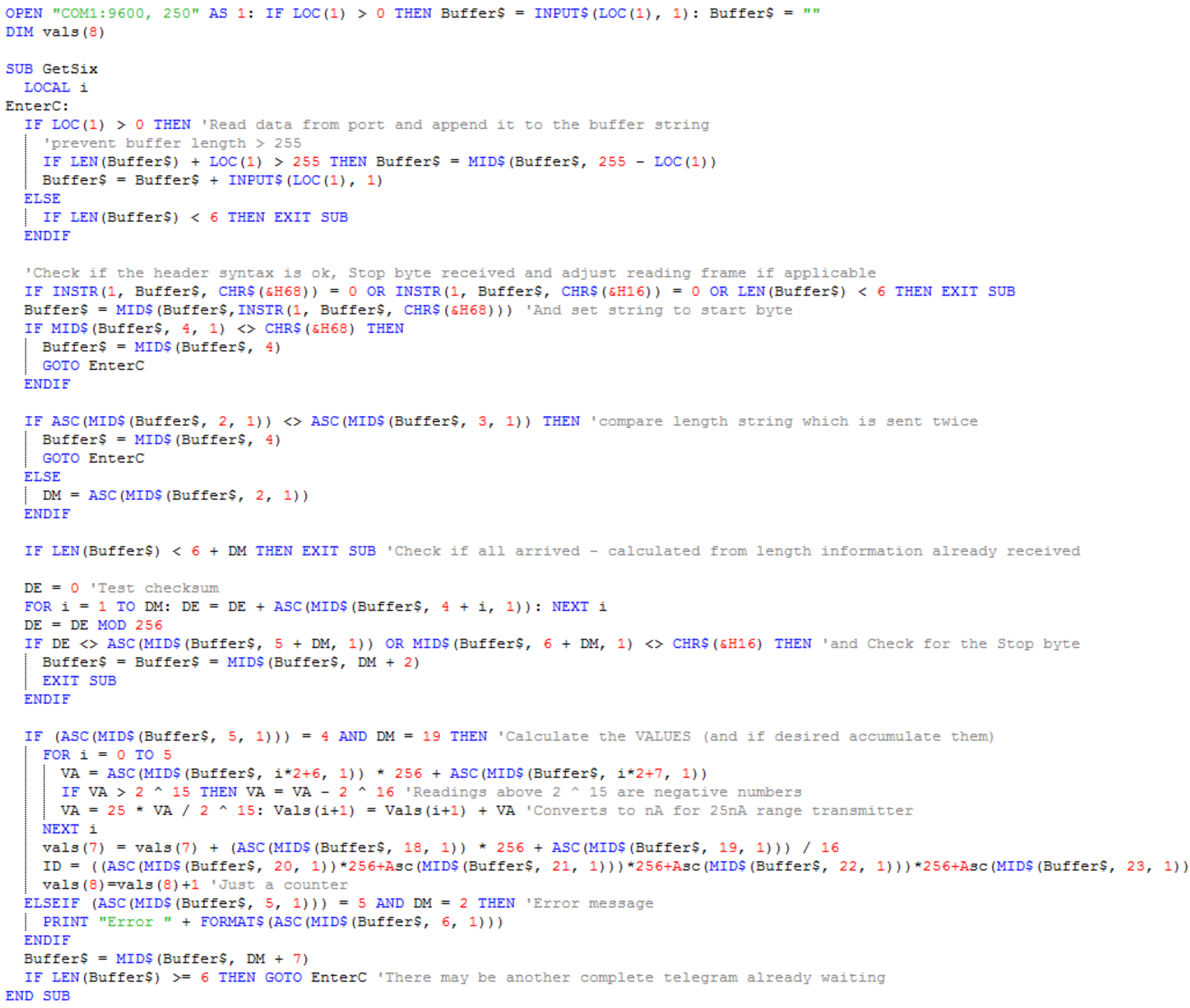
Data Telegram format

byte#	hex	Description	Comments
1.	68	Start	
2.	nn	Length	The length byte is the length from the number of data bytes + 1 (the message type id byte). Decimal value is 19.
3.	nn	Length	
4.	68	Start	
5.	04	Message type ID	Message type id is a unique number identifying the which message should be used to interpret the frame. Value is 4.
6.	xx	Channel1 Hi Byte	Frame data. If the value of one of the six channels is above 2^15 counts this represents a negative number. Then subtract 2^16. That gives a reading range of -2^15 to +2^15 counts. That range is equivalent to either +-25nA or +-50nA (check your Six transmitters label). If the values are exactly 32767 or -32768 this indicates a value above/below the measurement boundaries i.e. an error.
7.	xx	Channel1 Lo Byte	
8.	xx	Channel2 Hi Byte	
9.	xx	Channel2 Lo Byte	
10.	xx	Channel3 Hi Byte	
11.	xx	Channel3 Lo Byte	
12.	xx	Channel4 Hi Byte	
13.	xx	Channel4 Lo Byte	
14.	xx	Channel5 Hi Byte	
15.	xx	Channel5 Lo Byte	
16.	xx	Channel6 Hi Byte	
17.	xx	Channel6 Lo Byte	
18.	xx	Temperature Hi Byte	Divide the value by 16 to get the temperature in deg C.
19.	xx	Temperature Lo Byte	
20.	xx	ID Highest Byte (MSB)	
21.	xx	ID Byte 2	
22.	xx	ID Byte 3	
23.	xx	ID Lowest Byte (LSB)	
24.	xx	Checksum	The checksum is the last byte of sum of the values of the bytes of the message type id and the frame data (Bytes 5-23).
25.	16	Stop byte	

Error Telegram format

1.	68	Start	
2.	nn	Length	The length byte is the length from the number of data bytes + 1 (the message type id byte). Decimal value is 2.
3.	nn	Length	
4.	68	Start	
5.	05	Message type ID	Value for Error message is 5.
6.	xx	Error code	
7.	xx	Checksum	The checksum is the last byte of sum of the values of the bytes of the message type id and the frame data (Bytes 5-6).
8.	16	Stop byte	

Example Code in Basic



```
OPEN "COM1:9600, 250" AS 1: IF LOC(1) > 0 THEN Buffer$ = INPUT$(LOC(1), 1): Buffer$ = ""
DIM vals(8)

SUB GetSix
  LOCAL i
EnterC:
  IF LOC(1) > 0 THEN 'Read data from port and append it to the buffer string
    'prevent buffer length > 255
    IF LEN(Buffer$) + LOC(1) > 255 THEN Buffer$ = MID$(Buffer$, 255 - LOC(1))
    Buffer$ = Buffer$ + INPUT$(LOC(1), 1)
  ELSE
    IF LEN(Buffer$) < 6 THEN EXIT SUB
  ENDIF

  'Check if the header syntax is ok, Stop byte received and adjust reading frame if applicable
  IF INSTR(1, Buffer$, CHR$(&H68)) = 0 OR INSTR(1, Buffer$, CHR$(&H16)) = 0 OR LEN(Buffer$) < 6 THEN EXIT SUB
  Buffer$ = MID$(Buffer$,INSTR(1, Buffer$, CHR$(&H68))) 'And set string to start byte
  IF MID$(Buffer$, 4, 1) <> CHR$(&H68) THEN
    Buffer$ = MID$(Buffer$, 4)
    GOTO EnterC
  ENDIF

  IF ASC(MID$(Buffer$, 2, 1)) <> ASC(MID$(Buffer$, 3, 1)) THEN 'compare length string which is sent twice
    Buffer$ = MID$(Buffer$, 4)
    GOTO EnterC
  ELSE
    DM = ASC(MID$(Buffer$, 2, 1))
  ENDIF

  IF LEN(Buffer$) < 6 + DM THEN EXIT SUB 'Check if all arrived - calculated from length information already received

  DE = 0 'Test checksum
  FOR i = 1 TO DM: DE = DE + ASC(MID$(Buffer$, 4 + i, 1)): NEXT i
  DE = DE MOD 256
  IF DE <> ASC(MID$(Buffer$, 5 + DM, 1)) OR MID$(Buffer$, 6 + DM, 1) <> CHR$(&H16) THEN 'and Check for the Stop byte
    Buffer$ = Buffer$ = MID$(Buffer$, DM + 2)
    EXIT SUB
  ENDIF

  IF (ASC(MID$(Buffer$, 5, 1))) = 4 AND DM = 19 THEN 'Calculate the VALUES (and if desired accumulate them)
    FOR i = 0 TO 5
      VA = ASC(MID$(Buffer$, i*2+6, 1)) * 256 + ASC(MID$(Buffer$, i*2+7, 1))
      IF VA > 2 ^ 15 THEN VA = VA - 2 ^ 16 'Readings above 2 ^ 15 are negative numbers
      VA = 25 * VA / 2 ^ 15: Vals(i+1) = Vals(i+1) + VA 'Converts to nA for 25nA range transmitter
    NEXT i
    vals(7) = vals(7) + (ASC(MID$(Buffer$, 18, 1)) * 256 + ASC(MID$(Buffer$, 19, 1))) / 16
    ID = ((ASC(MID$(Buffer$, 20, 1))*256+Asc(MID$(Buffer$, 21, 1)))*256+Asc(MID$(Buffer$, 22, 1)))*256+Asc(MID$(Buffer$, 23, 1))
    vals(8)=vals(8)+1 'Just a counter
  ELSEIF (ASC(MID$(Buffer$, 5, 1))) = 5 AND DM = 2 THEN 'Error message
    PRINT "Error " + FORMAT$(ASC(MID$(Buffer$, 6, 1)))
  ENDIF
  Buffer$ = MID$(Buffer$, DM + 7)
  IF LEN(Buffer$) >= 6 THEN GOTO EnterC 'There may be another complete telegram already waiting
END SUB
```

Disclaimer: Not For Medical, Diagnostics And Usage On Humans. For Evaluation Use Only.