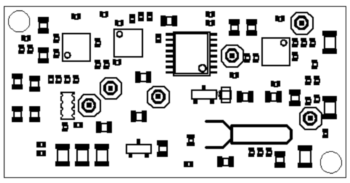
Pulse Oximeter OEM Board

EG00352

Technical Manual



Version 3.1 23.10.2014

medlab

Medlab medizinische Diagnosegeräte GmbH
Helmholtzstrasse 1
76297 Stutensee (Karlsruhe)
Germany
Tel. +49(0)7244 741100
support@medlab.eu
www.medlab.eu

Contents:

Overview:

The scope of this document is the description and specification of Medlab's EG00352 pulse oximeter board. It will enable the reader to integrate the board into a medical electronic system.

The probes (sensors) that can be used together with the EG00352 are pictured on the last pages of this document.

The document describes the basic technology used, the mechanical and the power supply considerations, as well as the software protocol for interfacing the EG00352 through its UART to a host system.

The EG00352 is a fully pre-tested PCB that connects to one of different probes to measure a patient's arterial oxygen saturation and pulse rate.

The interface to the host system consists of a connection to a well regulated and filtered DC power supply of 3.3 volts and an asynchronuous, serial connection to the host system.

The connection is unidirectional, e.g. after power up, the module starts to transmit data to the host without any needs for commands or other setup.

Transmitted are the SpO_2 value, the pulse rate, the plethysmographic curve and several status bytes, in an easily decodable format.

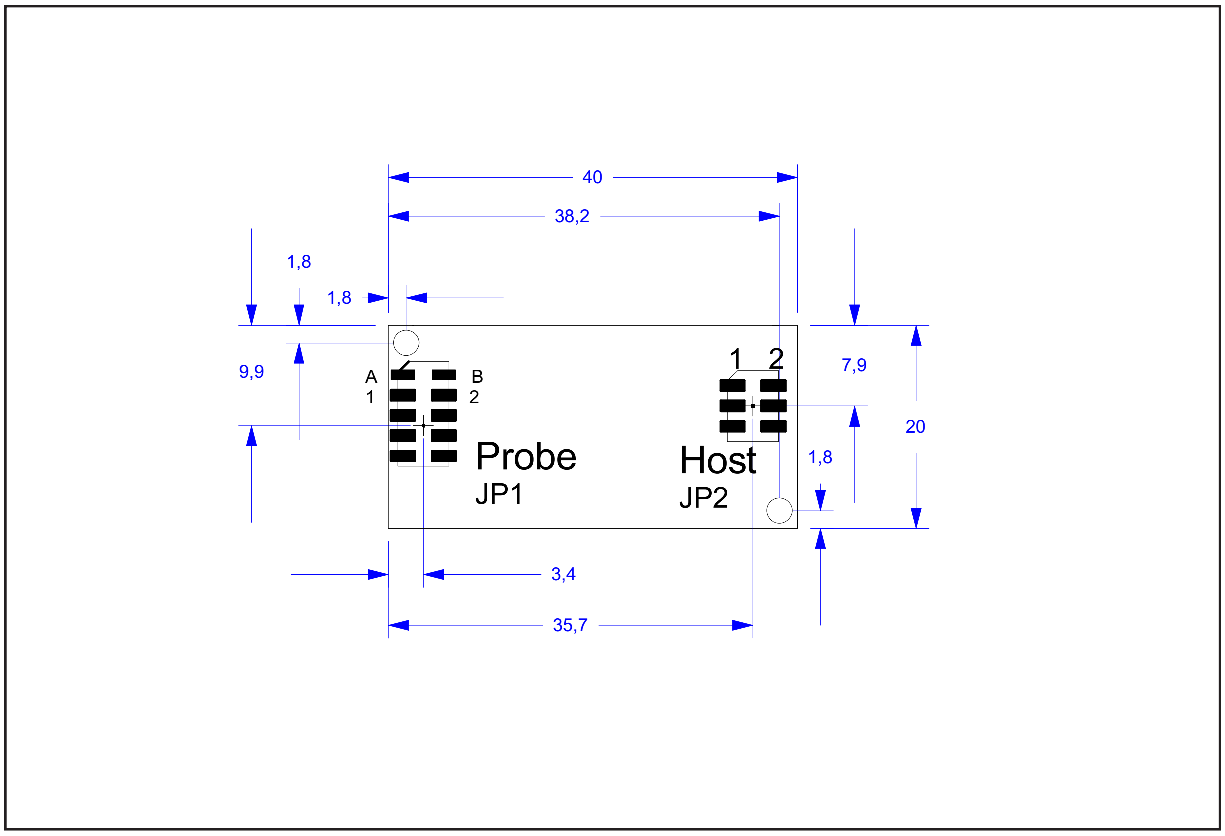
Sample code for decoding this data stream is included in this manual, a test software for displaying the data on a PC can be downloaded from the website *oem.medlab-gmbh.de*. Also the source code of this application is available there.

This manual describes two different software protocols. The standard protocol delivered is version 2, version 1 is available upon request only.

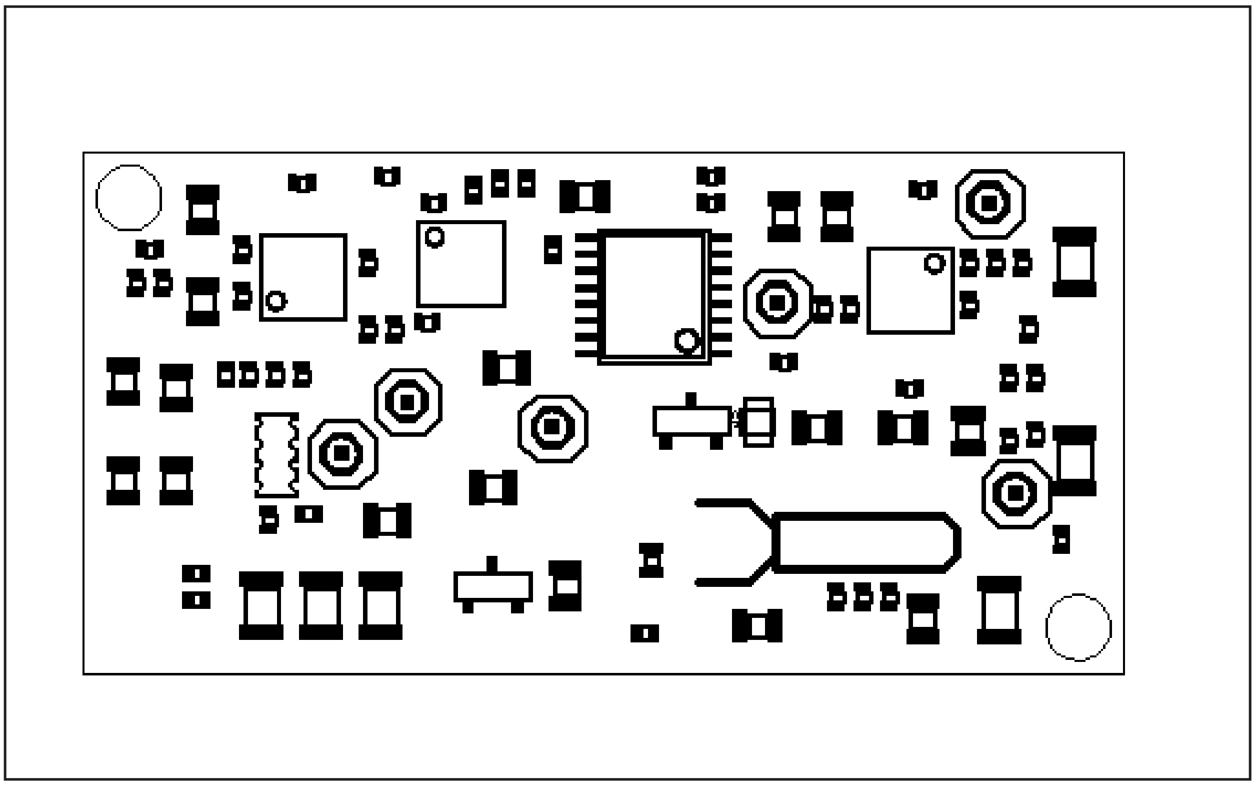
Technical data (Specifications):

Algorithm:	Classical "Split Pulse Wave Algorithm"		
Mechanical data:	see attached sheet with board drawing. 4 layer PCB, 40x20 mm, thickness 1 mm		
Attachment:	two M2.5 screws in the corners of the PCB. alternative: 2 parts Richco CHDLCBST-3-01		
Weight:	4 g		
Operating voltage:	3.3 Volt DC, +200 mV, - 100 mV 15...20 mA		
Power consumption:	50 mW to 65 mW, depending on light absorbance at measurement site		
Environmental:	Temperature		
	Storage	-30 °C to 90 °C	
	Operation	-20 °C to 50 °C	
	Humidity		
	Storage	0 .. 95%, non condensing	
	Operation	5 .. 90%, non condensing	
SpO_2			
Measuring range:	0%..100% of SpO2		
Accuracy:	70%..100%	:	+/- 2 %
	below 70%	:	not specified
Averaging:	fixed to 8 seconds		
Pulse Rate			
Measuring range:	30 .. 248 min^{-1}		
Accuracy:	+/- 3min^{-1}		
Averaging:	fixed to 8 beats		
Interface:	asynchronuous, serial interface with CMOS levels 9600 baud, 8N1		
Protocol:	simple unidirectional standard protocol Custom protocol versions available on request		

Mechanical dimensions of the module



*Mechanical drawing of the PCB (**Top View**), DXF file available upon request. Dimensions in mm*



Hardware Interfaces:

Serial Transmission

The host is connected to the board over a serial, asynchronuous channel with a baud rate of 9600 baud, 8N1.
CMOS voltage levels are used. The levels are at 0 and 3.2 volts, +- 100 mV.

In the standard protocol, only a unidirectional interface (EG00352 ---> host system) is necessary, but there are options available where, for example, the averaging can be set with a command sent to the EG00352.

Power Supply

The EG00352 module works with a 3.3 VDC power supply and generates all other necessary voltage from these 3.3 VDC on board. It is very important to regulate this supply to an accuracy better than +/- 5%, or permanent damage might occur to the module. The power supply should be well filtered.

Mechanical Connection

The EG00352 module uses two low-profile Samtec connectors spaced at 2mm to connect to the host PCB. One carries the signals to the host, the other is the connection to the probe (Sensor). The host system needs to include the opposite parts on a carrier PCB. The parts to be used are:

Samtec TMMH-103-05-F-DV	**Host Connector**
Samtec TMMH-105-05-F-DV	**Probe Connector**

(These are the male parts that have to be mounted in the host system)

Furthermore, the carrier board needs to include the connections of the EG00352 module to the DSUB9 probe connector. These connections carry very sensitive signals and should be kept as short as possible (< 30 mm). The serial output is less sensitive, so it is recommended to place the EG00352 as close to the probe connector as possible.

Connectors:

(see drawing on previous page for location)

Header for Probe connection JP1:

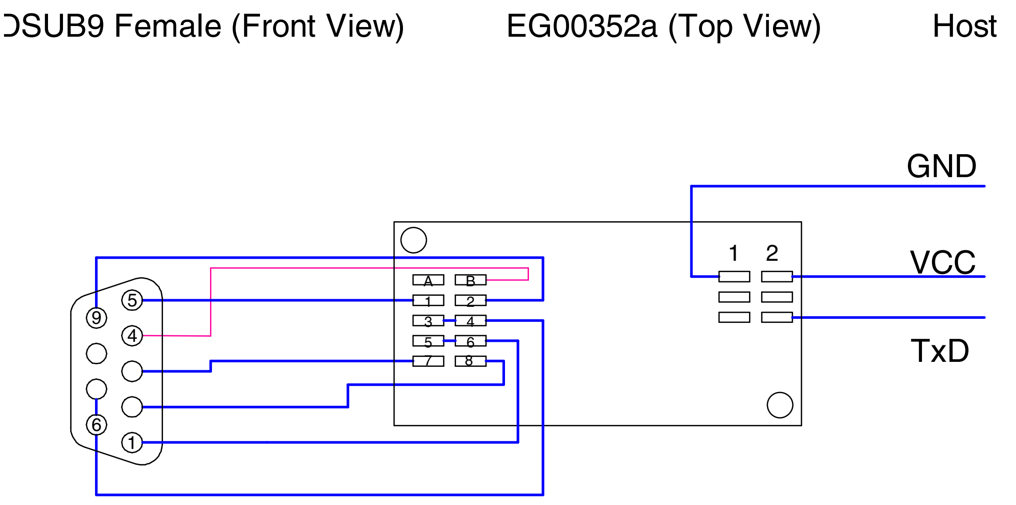
Probe:

A	not connected	
B	Sensor Coding1	(DSUB 9 pin 4)
1	Input Photodiode 1	(DSUB 9 pin 5)
2	Input Photodiode 2	(DSUB 9 pin 9)
3	GND	(DSUB 9 pin 6 and pin 7)
4	GND	(DSUB 9 pin 6 and pin 7)
5	Sensor Coding	(DSUB 9 pin 1)
6	Sensor Coding	(DSUB 9 pin 1)
7	Led Output 1	(DSUB 9 pin 3)
8	Led Output 2	(DSUB 9 pin 2)

Header for Host connection JP2:

Host:

1	Ground
2	VCC, 3.3 V DC
3	
4	
5	
6	TxD (CMOS level)



Serial Transmission (Protocol Version 1)

The transmission is not synchronized to any event in the SpO_2 calculation, but is sent in a five byte block 60 times per second. Most of this block contains redundant data, because, except for the wave, nothing changes so fast in the calculated and transmitted values. The power consumption of the EG00352 is slightly higher as during the usage of other protocols, due to the relatively large amount of overhead data that is sent.
Advantage:
Since the board always sends 300 bytes per second, it is easily recognized if transmission fails for some reason. Easy to decode.
Disadvantage :
The host CPU has to do much work for the incoming values because it is interrupted 300 times per second.

Data format:
4800 Baud, 1 Startbit, 8 Databits, even parity bit, 1 Stopbit

Bit 7 in Byte 1 is used for synchronisation of the blocks.

Byte	*Bit* *7 6 5 4 3 2 1 0*	*Definition*
1	*1*	*a "1" in bit 7 indicates the start of a new block*
	x	*pulse trigger (1 for about 100 ms if pulse detected)*
	x	*not used*
	x	*not used*
	x x x x	*Signal strength (0..7), 0x0f is bad signal*
2	*0*	
	x x x x x x x	*Plethysmogram sample value (0..99)*
3	*0*	
	x	*Bit 7 of Pulse*
	x	*not used*
	x	*problem with probe*
	x x x x	*Bargraph 0..15*
4	*0*	
	x x x x x x x	*Pulse bit 0 to 6*
5	*0*	
	x x x x x x x	*Spo2 (0..99)*

This 5 Byte block is transmitted with 60 Hz = 300 bytes/second.

Serial Transmission (Protocol Version 2)

The second protocol implements the same functionality as the first one without the large redundant data overhead of the first variant.
The plethysmographic waveform data is sent with a 50 Hz transmission rate. This rate was selected because higher frequencies do not produce smoother waves, and lower frequencies would lead to incorrect impressions of the waveform.

Data format:
9600 baud, 8 bits, 1 stop bit, no parity
On each detected pulse, a block with new saturation value, pulse rate and quality information / perfusion index and info is transmitted. The pulse wave sample points are transmitted continuously at 50 samples per second. Their values are located between 0 and 247, that means wave data points are always <=0xF7. Byte values larger than 0xF7 are used for marking the next byte in the stream as a new value with the following definition:

Marker byte	*Meaning of following byte(s)*
0xF8	*Wave sample points follow*
0xF9	*Spo2 value follows*
0xFA	*Pulse value follows*
0xFB	*Info byte follows*
0xFC	*Quality & Perfusion Index*

Definition of marker bytes

Info byte	*Meaning*
0x00	*OK*
0x01	*Sensor disconnected*
0x02	*No finger in probe*
0x03	*Low perfusion*
0x45	*Selftest Error*

Definition of "info" byte

Bits	b7	b6	b5	b4	b3	b2	b1	b0
	0	p2	p1	p0	q3	q2	q1	q0

Quality is expressed by a number between 0 and 10 . If it is 0, the quality of the pulse oximeter data acquisition is best. The bits q3 to q0 form this number.

P2, P1, P0 are used to transmit the Perfusion index (PI):

001	*Perfusion <= 0.25%*
010	*0.25% < Perfusion < 0.5%*
011	*0.5% < Perfusion < 1%*
100	*1% < Perfusion < 2%*
101	*2% < Perfusion < 4%*
110	*4%< Perfusion < 8%*
111	*Perfusion >=8%*

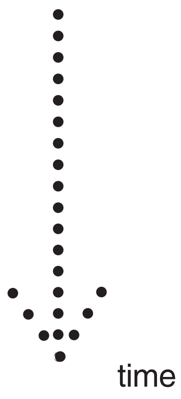
Remark: PI = 000 is unused to make it possible to distinguish between new and old firmware versions. If PI is 000, an old firmware is used and no PI is transmitted.

Definition of "quality" and perfusion byte

Examples for data streams:

Protocol Version 1:

BLOCK n	
0xC7	*new block, pulse trigger, signal strength = 7*
0x47	*plethysmogramm sample (between 0 and 99)*
0x08	*bargraph sample point is 8, pulse bit 8 = 0*
0x60	*Pulse = 96*
0x61	*SpO_2 = 97*
BLOCK n+1	
0x86	*new block, no pulse trigger, signal strength = 6*
0x49	*plethysmogramm sample (between 0 and 99)*
0x46	*bargraph sample point is 6, pulse bit 8 = 1*
0x30	*pulse = 48 + 128 = 176*
0x60	*SpO_2 = 96*
....	
....	
....	



Protocol Version 2:

`0xF9, 0x61,`	`0xFA, 0x80,`	`0xFB, 0x00,`	`0xFC, 0x70,`	`0xF8, 0x23, 0x25, 0x26, 0x28, 0x30`
SpO_2 = 97%,	*Pulse = 128,*	*Status = OK,*	*quality = OK, PI >= 8%*	*waveform,*

then sample points are following, until the next marker byte is inserted

The advantage of this protocol is the fact that the data type with the highest occurance, the waveform, does not have to be marked in any form. Nevertheless, it is possible to savely distinguish between different data types. If no finger is inserted or no probe is connected, zero is transmitted for both SpO_2 and Pulse once per second.

C Source Code Examples for Protocol 1

The following C source code is intended to help integrating the Medlab OEM pulse oximeter board into the customer's system. The data is received in the PC's serial interrupt and the values are copied in a data queue that is processed during the main program. The example is part of the source code we used for writing our PC demo program. It is written in Borland C.

```
void  decode_data(void)
        {
          while (!((val = getccb()) & 0x80));       /*  wait for sync bit                */
          if (val & 0x40)
            printf("!Puls!");                       /*  puls trigger active              */

          y = getccb();                             /*  get plethysmogram sample         */

          val = getccb();                           /*  get pulse bar sample             */
          puls_hbit = (val & 0x80)?1:0;             /*  store bit 7 of pulse             */
          bar_graph = val & 0x0F;                   /*  store bar_graph value            */

          printf("Puls %03u",0x80*puls_hbit + getccb());
                                                    /*  print pulse                      */

          printf("SpO2 %03u",getccb());             /*  print spo2                       */
     }


/* getccb() returns the next serial value from a queue that gets filled during the PC´s
serial interrupt */
```

Turbo C example for decoding of data protocol 1

C Source Code Examples for Protocol 2

The following C source code is intended to help integrate the Medlab OEM pulse oximeter board into the customer's system. The first example is part of the source code we used for writing our PC demo program and is written in Borland C.
The second example was originally written for usage with a 8051-family controller using the Keil C51 compiler. The data is received in the serial interrupt and the values are copied to global variables that can be processed during the main program.

```
/* getccb() returns the next serial value from a queue that gets filled during the PC´s serial interrupt */

 while (1)
 {
  if ((val=getccb()) == 0xF8)
   {
    while((val=getccb()) < 0xF0)
       {
                 /* here "val" always contains a new plethysmogram sample              */
                 /* process it acccording to your needs .........                      */
       }
   }

  switch(val)    /* now val contains a marker that means that the next byte is a special value  */
   {
     case 0xF9:
           printf(”%02u”,getccb());                /* print SpO2                     */
           break;
     case 0xFA:
           printf(”%03u”,(unsigned char)getccb()); /* print pulse                    */
           break;
     case 0xFB:
           switch(getccb())
           {
            case 0: gotoxy(20,23);
                  printf(“        OK !     “);      /* print messages                */
                  break;
            case 1: gotoxy(20,23);
                  printf(“   No sensor connected !   “);
                  break;
            case 2: gotoxy(20,23);
                  printf(“   No finger in probe !    “);
                  break;
            case 3: gotoxy(20,23);
                  printf(“   Low perfusion     !    “);
                  break;
           }
           break;
     case 0xFC:
           val = getccb();
           printf(”%02u”,getccb());              /* print signal strength */
           break;
   }
 }
```

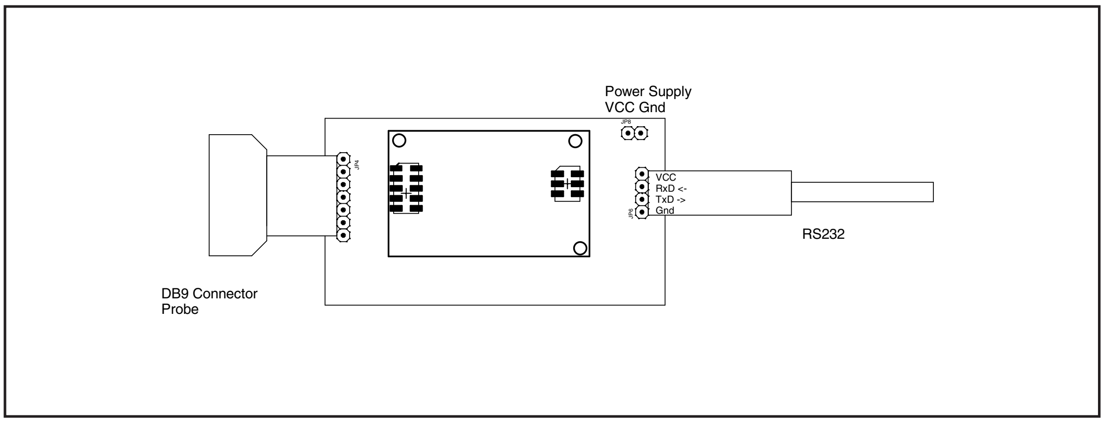
TurboC Example for PC decoding of protocol 2

```
data byte data *rcvptr;
data byte Oxval;
data byte Oxgraph;
data byte Oxpuls;
data byte Oxinfo;
data byte Oxqual;

data bit  Tbit;
data byte Serval;

void serial_int() interrupt 4 using 2
{

 if(TI)                                   /* transmitter int ?                   */
  {
   TI = 0;
   Tbit = TRUE;
    return;                               /* nothing to do                       */
  }

 RI = 0;                                  /* else must be receiver int           */

  Serval = SBUF;                          /* get value from serial buffer register */
   if (Serval > 0xF5)                     /* is it a code ?                      */
    {
     switch (Serval)                      /* yes                                 */
      {

       case 0xF8:    rcvptr = &Oxgraph;   /* next time get ox curve              */
                     return;
        case 0xF9:   rcvptr = &Oxval;     /* next byte is get ox value           */
                     return;
        case 0xFA:   rcvptr = &Oxpuls;    /* next byte is puls value             */
                     return;
        case 0xFB:   rcvptr = &Oxinfo;    /* next byte is ox info                */
                     return;
        case 0xFC:   rcvptr = &Oxqual;    /* next byte is quality information    */
                     return;
        default :   return;
       }
     }
   else
      *rcvptr = Serval;                   /* byte is no code, so store it where pointer points */
  }
 return;
}
```

Code for interrupt driven decoding of protocol 2 using an 8051 microcontroller

Plug-and-Play Test Kit

To ease the work of evaluating the unit, there is a complete, ready-to-run testkit available. It is easily possible to evaluate the module with a PC software that is adapted to the pulse oximeter's interface protocol. The software displays all relevant data that is transmitted in the protocol version. It runs on each PC under windows. Also a complete set of cables and an adpater for connection of the probe to the board is included in this kit.
The source code of the PC application can be downloaded from the web.



Connection of the pulse oximeter to the probe and to the PC adapter

Usage:

Connect the power supply connector to a well regulated 3.3 volt DC supply. Connect the serial cable to COM1 or COM2 of a PC. Only Ground, TxD, and RxD are used in the interface. The voltage levels of the signals are +/-8 volts, generated by a small electronic circuit in the connector.

- connect the other side of the PC cable to the PCB as shown in the drawing
- connect the probe to the probe adapter and this adapter to the carrier board
- turn on the power supply
- turn on PC

Download the test program on oem.medlab-gmbh.de. The program does not need to be installed, you can start the "pox-2-pc.exe" file directly. The download also includes the source code in VB6.
Put your finger into the probe and the values and the plethysmogram will be displayed shortly.

Note: This setup should be used for internal testing only. It does not comply to the power supply isolation requirements of IEC601 or ISO9919 (patient connection should be of type BF).

Regulatory considerations

The device that has been described in this document is not a final medical product. That means that it cannot be used as a standalone unit to do pulse oximetry measurements.
Therefore, the OEM pulse oximeter has not been - and also cannot be - CE-marked. The customer has to undertake the procedure of CE-marking with the final product that contains the EG00352 module. However, several products on the market have successfully passed this certification.

In regard to EMC testing, it is important to supply the module with a properly filtered voltage. The probe inputs and LED outputs are fed through small ferrite beads on the EG00352 already.

The module complies to the following standards, as far as applicable:

DIN EN ISO 60601-1:1996
DIN EN ISO9919:2005

A copy of the desaturation study performed with the EG00352 module at the university hospital of Lübeck is available to customers upon request.

During testing and certification of a product, also the user manual of the final product needs to be certified. The user manual has to contain certain technical data and warnings to the end users. We can support customers by supplying material for the manual that has been used during the certification process of Medlab's own line of pulse oximeters.

The EG00352 module does not integrate any power supply isolation. The ISO 9919 pulse oximeter standard asks for a type BF patient connection. Depending on your application, you might need to integrate such an isolation into your product. Please contact us for further information.

Available probes

The following pictures show only a selection of the available probes:

Fingerclip

Small Fingerclip Sensor

Wrap Probe

Vet Probe (Tongue)

Universal Y-Probe

Manual Revisions:

V0.9	1.11.2005	Preliminary Release
V0.91	2.11.2005	Corrected Protocol Detail in Protocol 2
V1.0	30.12.2005	Released without further changes
V2.0	30.12.2006	Changed mechanical description to new board revision
V2.5	17.07.2009	Added perfusion index description
V2.6	11.06.2012	Changed manufacturer address
V3.0	30.06.2012	Changed probe connector description, added Coding1
V3.0	06.11.2012	Removed ear sensor on page 17
V3.1	23.10.2014	Changed accuracy of pulse rate to bpm from %

Board Revisions:

2005	Release (Board Rev. A)
2006	Mechanical Changes (Board Rev. B and C)
2009	Protocol Change: Perfusion index included (Board Rev. D)
2012	Added Coding 1 to probe connector (Board Rev. E)