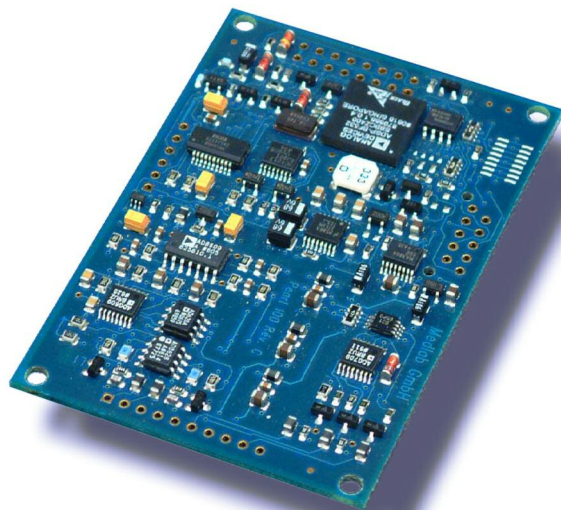


# medlab

## Pulse Oximeter OEM Board

# PEARL100™

### Technical Manual



Copyright © Medlab 2004-2008

Version 1.1 10.6.2008

PEARL = Pulse Enhancement Artefact Rejection Logic

PEARL is a registered trademark of Medlab GmbH

## Revisions:

V0.9	1.08.2004	Preliminary Release
V1.0	02.03.2005	Added Probe Pictures
V1.01	12.07.2006	Added DSUB-JP1 Adapter drawing
V1.10	10.06.2008	Minor changes, new probe pictures

Medlab medizinische Diagnosegeräte GmbH  
Erzbergerstrasse 115  
D 76133 Karlsruhe  
Germany  
Tel. +49 721 62512 0  
FAX + 49 721 62512 12  
oemsales@medlab-gmbh.de  
www.medlab-gmbh.de

This manual and all contents are copyright by Medlab GmbH 2004-2008

## Contents:

Overview	4
PEARL Algorithm	5
Technical Data	6
Mechanical Dimensions	7
Hardware Interface	
Serial Interface	8
Power Supply	9
Connector Assignment	10
Software Protocols	
Version 1	11
Version 2 (EG00302 native)	12
Examples for data streams	13
C source code for host	
Protocol 1	14
Protocol 2, PC	15
Protocol 2, Keil C51	16
Test Kits	17
Regulatory Considerations	18
Available Probes	19

## Overview

The scope of this document is the description and specification of Medlab's PEARL100 pulse oximeter module. It is intended to help to integrate the board into another medical device manufacturer's electronic system.

The probes (sensors) that can be used together with the PEARL100 are described on the last pages of this document.

The document describes the basic technology used, the mechanical and power supply considerations and the software protocol for interfacing the PEARL's UART to a host system.

The PEARL is an electronic PCB, fully pre-tested, that connects to one of the different Medlab SpO<sub>2</sub> probes to measure a patient's arterial oxygen saturation and his pulse rate.

The interface to the host system needs a filtered and regulated DC power supply of 2.8 to 5.5 volts and an asynchronous, serial connection to the host system.

Depending on the selected protocols, the baud rate varies between 4800 and 115200 baud, and also depending on protocol, the connection is uni- or bidirectional.

## PEARL Algorithm

The PEARL100 module uses the proprietary PEARL algorithm, that constantly shifts a time window of six seconds over a data buffer that contain samples of the red and infrared waveforms. The algorithm detects the correct pulse rate by convoluting a template over the waveform at different phase angles. This technology is also known as "cross correlating" the signal to another in digital signal processing. On the analog side, the PEARL uses a lock-in amplifier that enables the module to work even under extreme conditions, as high ambient light and other sources of disturbance. The basic operation frequency of that amplifier is 20 kHz, and the used analog front end very effectively suppresses noise that is not in the band around 20kHz. The 20 kHz noise potentially present is then removed by digital and analog filtering of the signal.

The PEARL algorithm leads to the effect that real time pulse detection is not any more coupled directly to calculation of correct saturations, since the mentioned six second buffer is used for all calculations.

Since the algorithm itself does not detect the pulse in real time, a traditional pulse detector is applied to the input signal and is responsible for the realtime pulse detection that can be used by the host system to generate a pulse tone.

Once per detected pulse (dependant on the protocol) , the new calculated  $SpO_2$  and pulse values are transmitted to the host system. If an audible pulse tone is needed, the point in time of data transmission can be used for that purpose.

## Technical Data (Specifications)

**Mechanical Data:** 6-layer PCB, thickness 1.5 mm  
Attachment: Four holes in the corners of the PCB

Weight: 23 g

**Power Supply:**  
Operating Voltage: 2.8 - 5.5 Volt DC,  
33...60 mA depending on interface

Power Consumption: 100mW to 150mW, depending on operating voltage  
lowest power consumption at 3.3 volts (100mW)

### Environmental:

Temperature:	Storage	-30°C to 90°C
	Operation	-10°C to 50°C
Humidity	Storage	0 .. 95%, non condensing
	Operation	5 .. 90%, non condensing

**SpO<sub>2</sub> :**  
Measuring range: 30%..100% of SpO<sub>2</sub>

Accuracy:	90%..100%	:	1% , +/- 1 Digit
	80%..89%	:	2% , +/- 1 Digit
	70%..79%	:	3% , +/- 1 Digit
	below 70%	:	not specified

Averaging: Fixed or adjustable through serial interface, depending on protocol

**Pulse Rate:**  
Measuring range: 30 .. 250 bpm

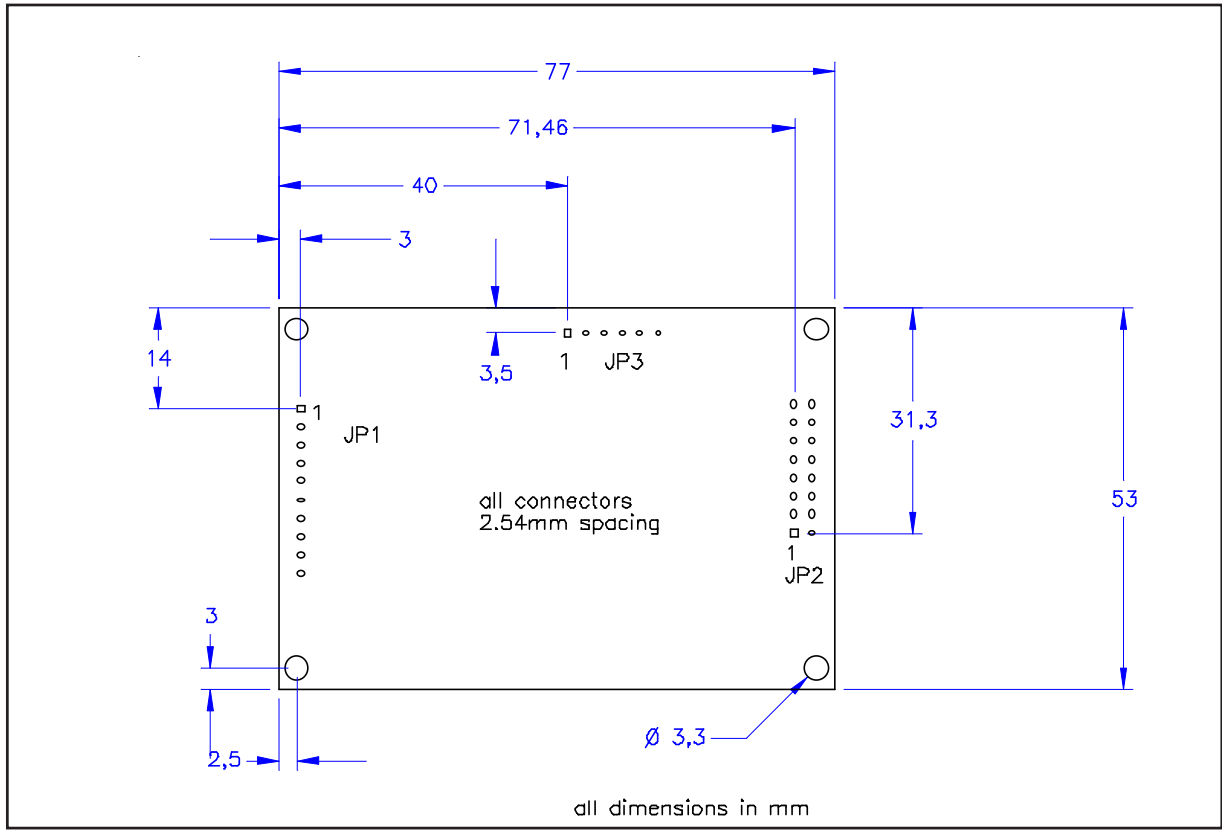
Accuracy: +/- 1%, +/- 1 Digit

Averaging: Fixed to 8 beats

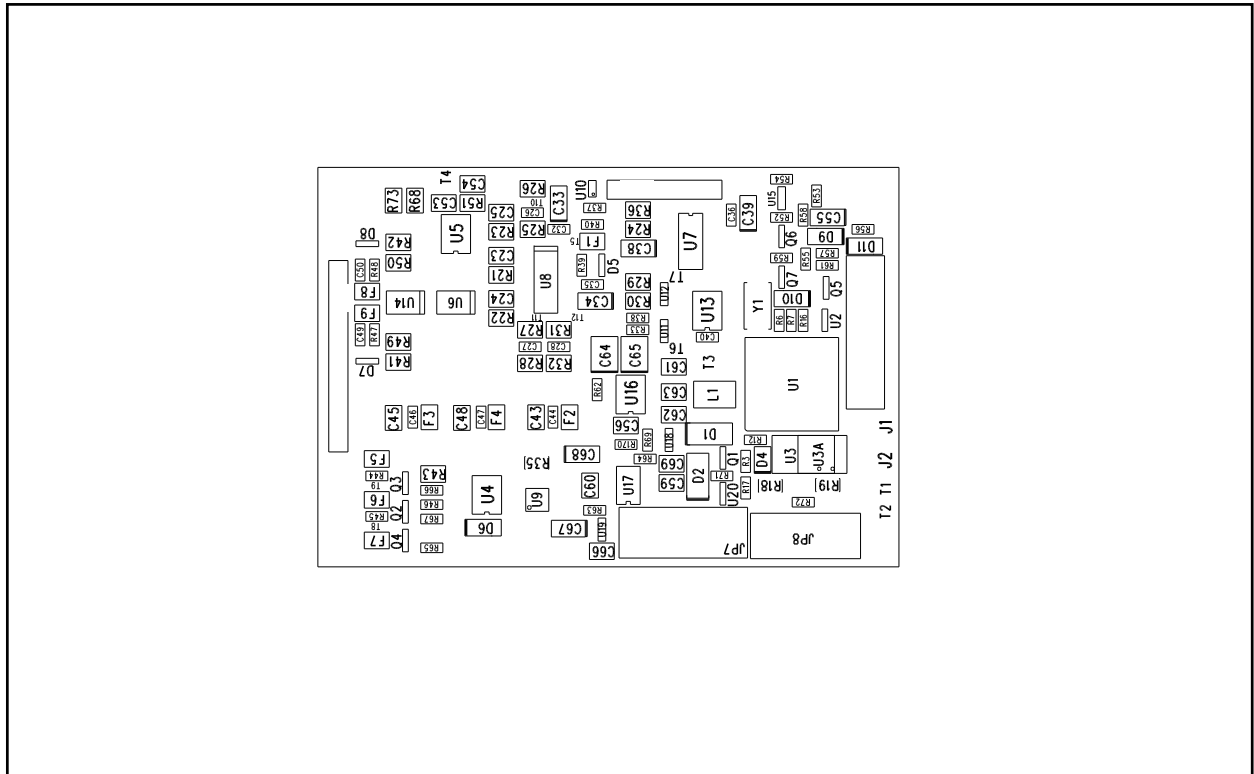
**Interface:** Asynchronous, serial interface with TTL or RS232 levels  
baudrate and data format depending on protocol.

**Protocol:** Two standard protocols, with or without  
plethysmogram transmission; see description on  
following pages; native EG00302 protocol supported  
Custom protocol versions available on request

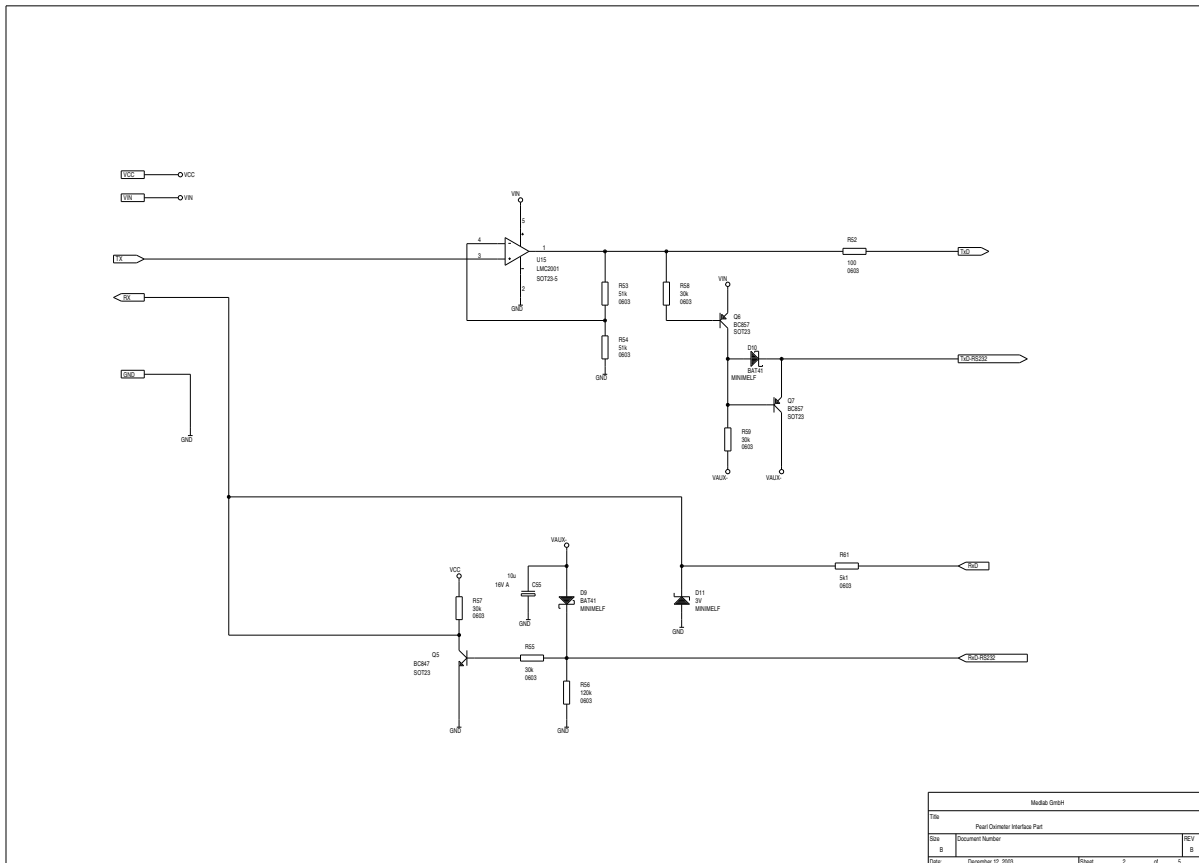
# Mechanical Dimensions



Mechanical drawing of top view of the PCB (original size)



# Hardware Interface



Interface part of the pulse oximeter's electronics

## Serial Interface

The connection to the board is done through a serial, asynchronous communication channel with protocol dependend baudrate and transmission parameters.

Both CMOS and RS232 voltage levels are available.

As can be seen on the schematic on top of the screen, the board adapts to the input voltage of the system. That means, if the board is powered by 5V DC, the RxD, TxD levels are 0 and 5 Volts, +/- 50 mV. If the board is powered by 3.3V DC, the levels are 0 and 3.3V, +/- 50 mV.

The RS232 option is in fact not a real RS232 interface, but is helpful during evaluation of the board, which can be done using an ordinary PC and a special test software. The board uses the positive supply as the "0" output level and uses the negative voltage from the incoming RS232RxD pin to generate the negative "1" output level. Therefore, RS232RxD and RS232TxD must be connected to the host, or no RS232 level output will be available. The board should be powered with 5VDC when using this option, otherwise, the positive level is not high enough to trigger the receiver UART reliably.

The connection in the customer's final system should preferably be done through CMOS levels.

In the two standard protocols, only a unidirectional interface (PEARL100 ---> host system) is necessary, but there are options available where, for example, the averaging can be set with a command.

**Note:**

The serial protocol supports a flashing LED that is shortly lit with each detected pulse. It is located close to the JP2 connector of the system. This is helpful for evaluation but consumes additional power. If in the final system, this is no longer needed, jumper J100 at the bottom of the PCB can be opened.

## **Power Supply**

The PEARL module internally works with a 3.3V power supply and generates all other necessary voltage from these 3.3 V. To ensure compatibility with the old EG00302 module, the input range had to be extended to accept 5V DC also.

Therefore, the 3.3V for the board internally are generated by a step up /step down converter that enables the PEARL to run on voltages ranging from 2.8 to 5.5 V DC.

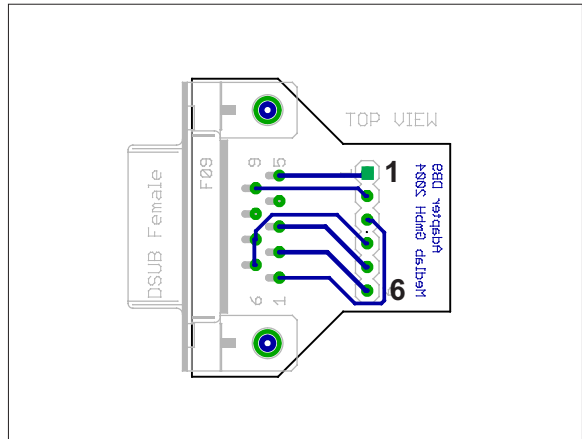
Best efficiency is reached when powering the module with 3.3 Volts DC. The 3.3 V should be connected to two special pins on JP2. By connecting the voltage to these pins, the internal step up/step down converter is disabled and the module is powered directly by the 3.3 V source. It is then very important to regulate this supply to an accuracy better than +/- 5%, or permanent damage will occur to the module.

# Connectors

(see drawing on page 7)

## Header for Probe Connection:

- JP1:
- 1 Positive Input Photodiode
  - 2 Negative Input Photodiode
  - 3 Sensor Coding  
(shorted to GND or connected to coding resistor in the probe)
  - 4 Ground
  - 5 Led Output 1
  - 6 Led Output 2
  - 7 don't connect
  - 8 don't connect
  - 9 don't connect
  - 10 don't connect



Adapter for DSUB 9 probe to PEARL100 board

## Header for Host Connection:

- JP2:
- 1 Ground
  - 2 Ground
  - 3 Txd (CMOS level)
  - 4 RS232Txd (RS232 level)\*
  - 5 Rxd (CMOS level)
  - 6 RS232RxD(RS232 level)\*
  - 7 n.c.
  - 8 n.c.
  - 9 n.c.
  - 10 n.c.
  - 11 /Powerdown
  - 12 Aux Port
  - 13 3.3 VDC input
  - 14 3.3 VDC input
  - 15 VCC input, 2.8 - 5.5 VDC\*\*
  - 16 VCC input, 2.8-5.5 VDC\*\*

## Alternative Header for Host Connection :

- JP3
- 1 /Powerdown (not in Rev. 1
  - 2 RxD (TTL Level)
  - 3 n.c.
  - 4 TxD (TTL level)
  - 5 Ground
  - 6 VCC input (2.8-5.5VDC)

\* see description under "Hardware Interface"

\*\* use only pin 13 and 14 **OR** 15 and 16 for supply, never use both!

# Serial Transmission (Protocol 1)

The transmission is not synchronized to any event in the SpO2 calculation, but is sent 60 times per second, in a five byte block. The power consumption of the PEARL is slightly higher than in other protocols, due to the relatively large amount of data to be sent.

**Advantage :**

Since the board always transmits 300 bytes per second, it is easily recognized if transmission fails for some reason. Easy to decode

**Disadvantage :**

The host CPU has to process a relatively high amount of mostly redundant data because data is received 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								Definition
	7	6	5	4	3	2	1	0	
1	1	x		x				x x x x	a "1" in bit 7 indicates the start of a new block pulse trigger (1 for about 100 ms if pulse detected not used not used 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		x				x x x x	Bit 7 of Pulse not used problem with probe 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 2)

The second protocol implements the same functionality than the first one without the large redundant data overhead of the first protocol.

Versions with and without plethysmographic data transmission are available.

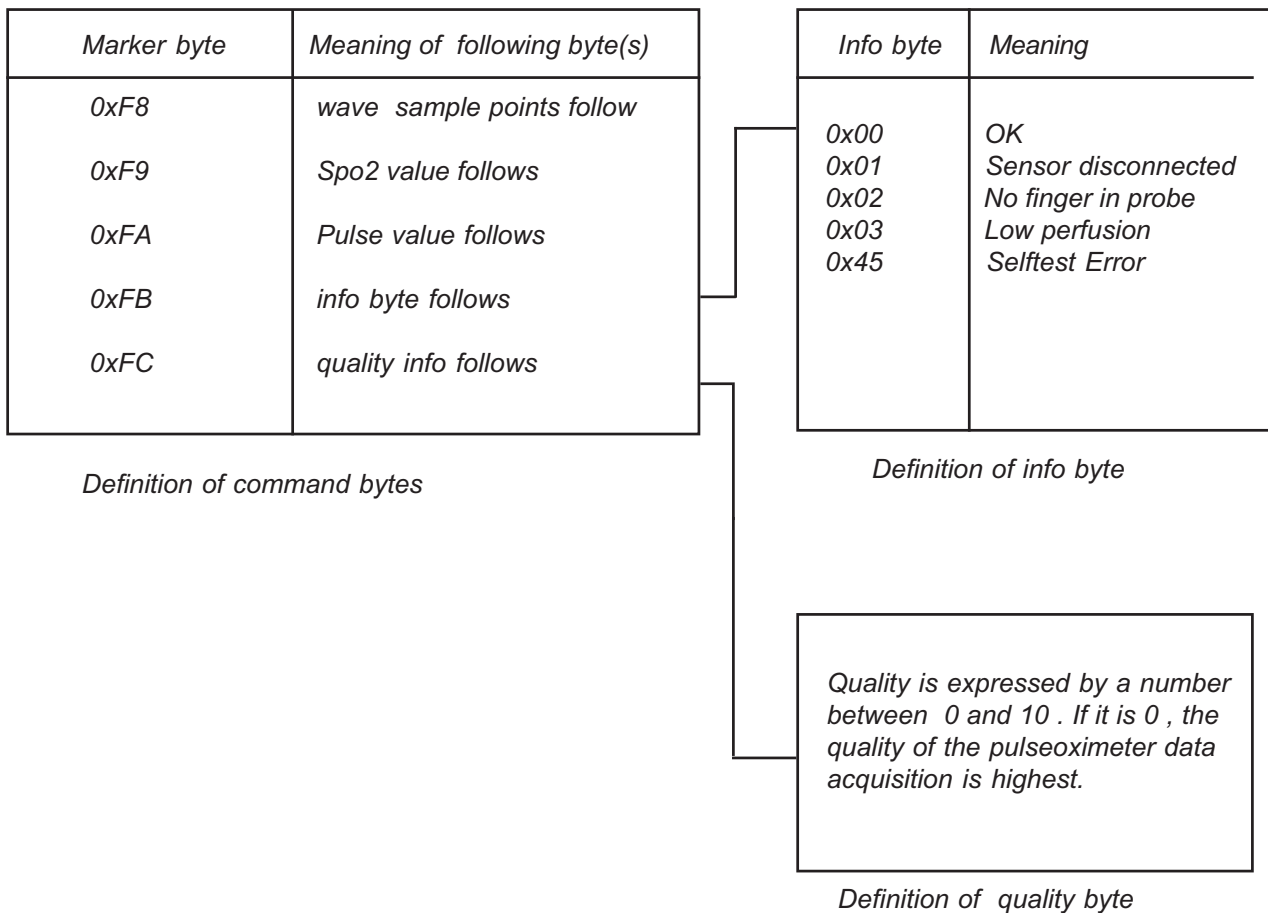
This is important if no pulse wave is required and the receiving host system would be overloaded by a large data stream.

Plethysmographic waveform data is available at 50 Hz transmission rate. This was selected since higher frequencies do not produce smoother waves, lower frequencies are leading to an incorrect impression of the waveform.

Data format:

**9600 baud, 8 bits, 1 stop bit, no parity**

On each detected pulse, a block with new saturation, pulse rate and quality information is transmitted. The pulse wave sample points are transmitted continuously with 50 bytes per second. Their values are located between 0 and 0xF7. Values that are higher than 0xF8 are used for marking the following data byte as a new data value with the following definition:



If a version without plethysmogram is selected, all information and marker bytes concerning plethysmographic waveform are skipped and are not transmitted.

# Examples of Data Transmission

## Protocol Version 1:

<i>BLOCK n</i>		
0xC7	<i>new block, pulse trigger, signal strength = 7</i>	
0x47	<i>plethysmogramm sample (between 0 and 99)</i>	
0x08	<i>bargraph sample point is 8, pulse bit 8 = 0</i>	
0x60	<i>pulse = 96</i>	
0x61	<i>spo2 = 97</i>	
<hr/>		
<i>BLOCK n+1</i>		
0x86	<i>new block, no pulse trigger, signal strength = 6</i>	
0x49	<i>plethysmogramm sample (between 0 and 99)</i>	
0x46	<i>bargraph sample point is 6, pulse bit 8 = 1</i>	
0x30	<i>pulse = 48 + 128 = 176</i>	
0x60	<i>spo2 = 96</i>	
.....		
.....		
.....		

## Protocol Version 2:

0xF9, 0x61,	0xFA, 0x80,	0xFB, 0x00,	0xFC, 0x00,	0xF8, 0x23, 0x25, 0x26, 0x28, 0x30
.....				
Spo2 = 97%,	Pulse = 128,	Status = OK,	Quality = 0,	Waveform, sample points follow

*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 occurrence, the waveform, does not have to be marked in any form. Nevertheless, it is savely possible to distinguish between different data types. If no finger is inserted or no probe is connected, zero is transmitted for both Spo2 and Pulse, once per second.*

## C Source Code Examples for Protocol 1

The following C sourcecodes are intended to help integrate the Medlab OEM pulse oximeter board into the customers 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.

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

    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 */
```

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 customers system. The first example is part of the sourcecode we used for writing our PC demo program and is written in 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 is 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 according 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;
  }
}

```

C 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 */

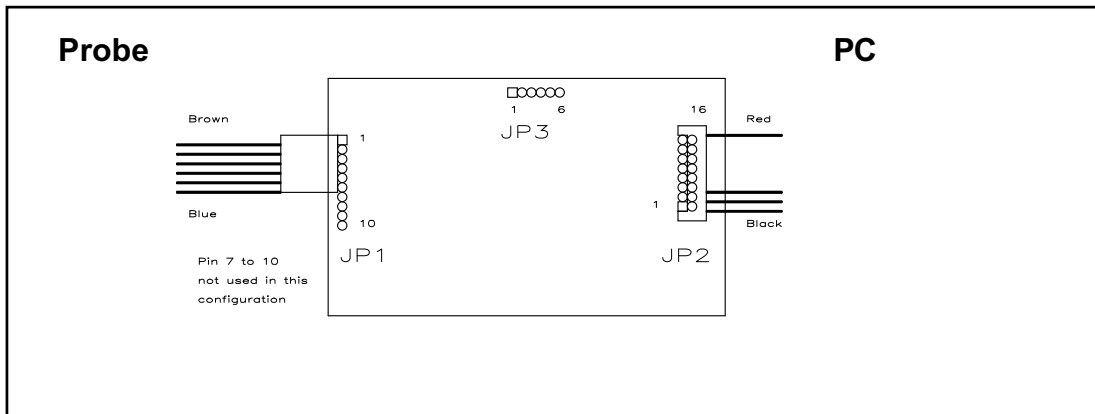
    Serval = SBUF;                          /* get value from serial buffer register */
    if (Serval > 0xF5)                       /* is it a code ? */
    {
        switch (Serval)                     /* yes */
        {
            case 0xF8: rcvptr = &Oxgraph;    /* next time get ox curve */
                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 microcontroller

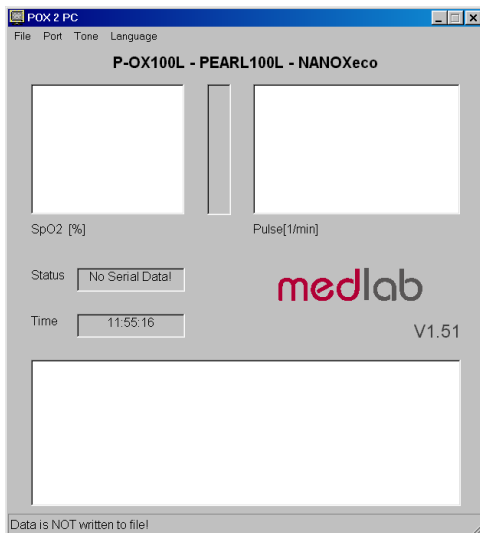
# Testkit

To ease evaluation of the module, there is a complete, ready-to-run testkit available: it comes together with a PC software that is adapted to the pulse oximeter 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 is included in this kit. It also contains one fingerclip probe P-100 and an adapter to connect the probe to the PEARL100 board



Connection of the Pulse oximeter to probe and PC adapter

## Usage



- Connect the red cable to a regulated 5VDC voltage source and the black cable to the power supplies GND.
- Connect the serial cable to a COM port of the computer. Only Ground, TxD and RxD are used in the interface. If you use a USB-RS232 adapter, please install the "virtual serial port" driver delivered with the adapter.
- Connect the other side of the cable to the PCB as shown in the drawing
- Connect the probe as shown in the drawing, using the adapter
- Turn on the power supply
- Turn on PC

## Regulatory Considerations

The PEARL100 module 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 does not have to be - and cannot be CE-marked. The customer has to undertake the procedure of CE-marking the final product that contains the PEARL100 module.

One important part of this is EMC testing the complete medical device. It is important to supply the module with a properly filtered, stable DC voltage. The probe inputs and LED outputs are fed through small ferrite beads on the module already.

The module complies to the following standards :

IEC60601-1:1996  
IEC60601-1-4:2001  
EN ISO9919: 2005

If the final medical device is mains powered or the device can be connected to another mains powered part while a patient is connected, an isolation is required, since ISO9919 requires the patient connection of a pulse oximeter to be of type BF or CF. That means that the power supply and the communication channels have to be galvanically isolated from the main part of the medical device. We can support our customers with the design of a simple, reliable and cost effective solution for this.

# Available Probes



P 100  
Fingerclip probe



WR 100  
Wrap probe



PS 100  
Small Fingerclip probe



Y 100  
Universal Y-probe



R 100  
Oxiflex large finger probe



EA 100  
Small earclip probe



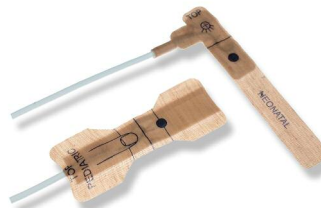
RM 100  
Oxiflex medium finger probe



V 100  
Tongue probe - for veterinarian applications



RS 100  
Oxiflex small finger probe



Disposable probes  
Delivery:  
Package of 24 pieces

