PPEB076 Operating Guide

Application note: ICR operating board

The PPEB076 is developed to work with the Neophotonics ICR / micro-ICR and the Neophotonics daughterboard. It provides a serial programming interface to control the ICR from within automated test setups.

Included features:

- Individual power supply for each TIA with enable/disable, including current measurement.
- Read out of current on each photo-diode (9)
- Auto-ranging or manual ranging of the photo-current detection
- Built-in VOA control

1. Contents

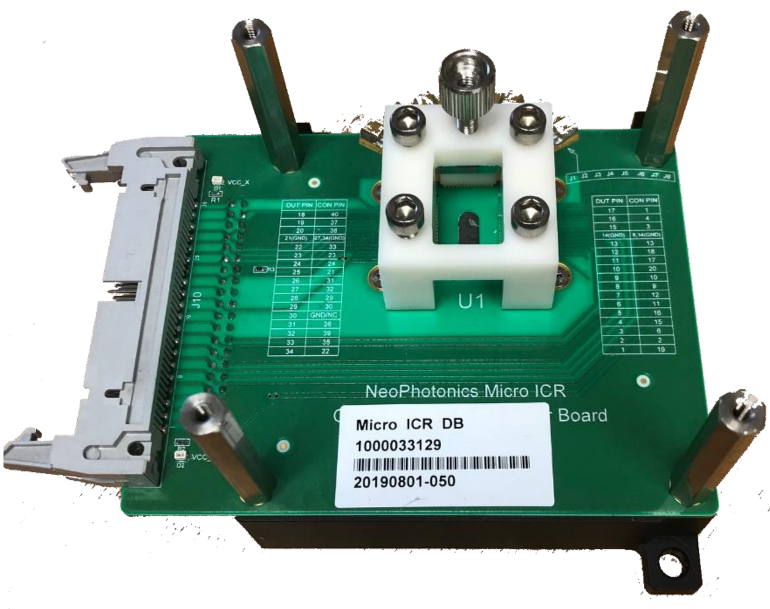
2. Installation procedure for PPEB076 ... 3
3. Electrical Connectors ... 6
4. Communications Port ... 7
5. Communications Protocol ... 8
6. Registers ... 10
Register 0x9A – PD resistance ... 11
7. Command Line Interface ... 12
8. Execfile code ... 13
9. Firmware Upgrade ... 16

2. Installation procedure for PPEB076

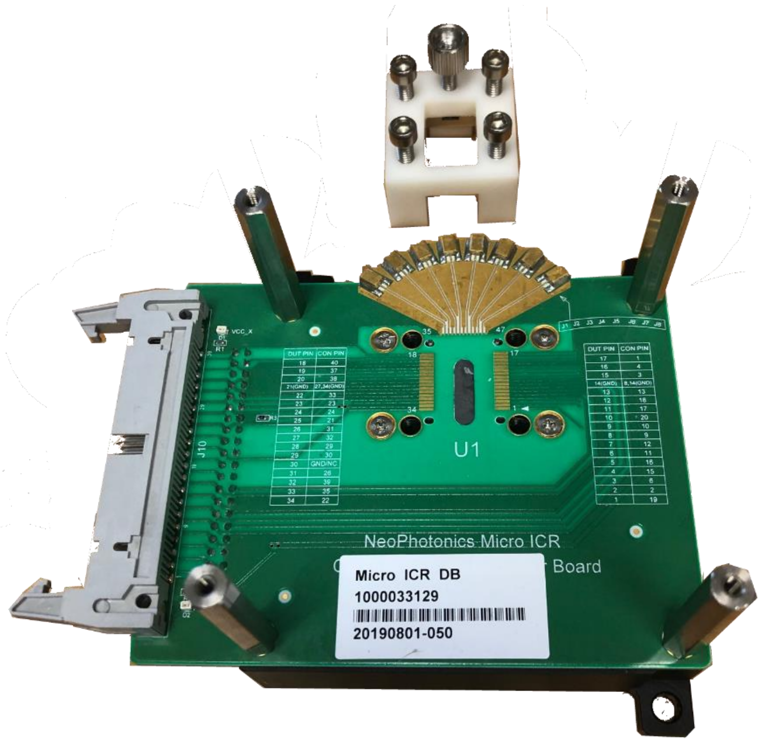
The original Neophotonics daughterboard is shown below (micro-ICR).

- Remove the 4 hex screws on the top of the board and keep them

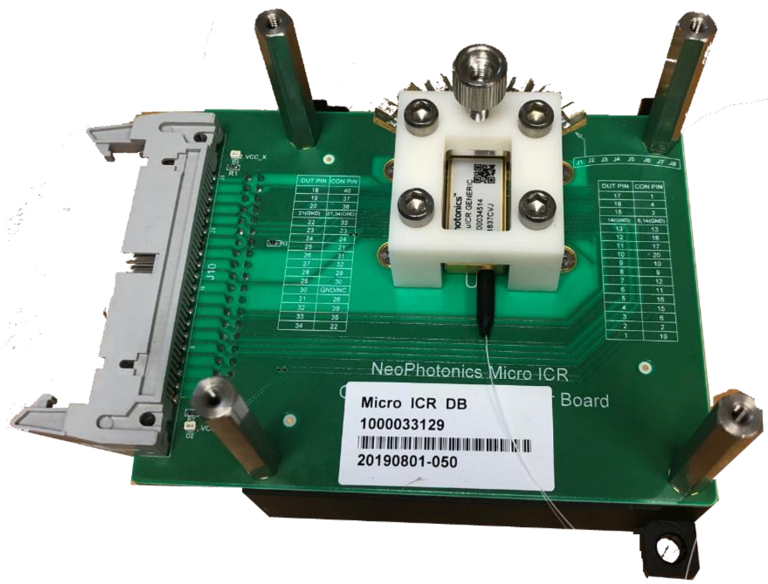
- Replace them with the 1.5 inch long hex standoffs



- Remove the pyrex holder for the (micro-ICR)



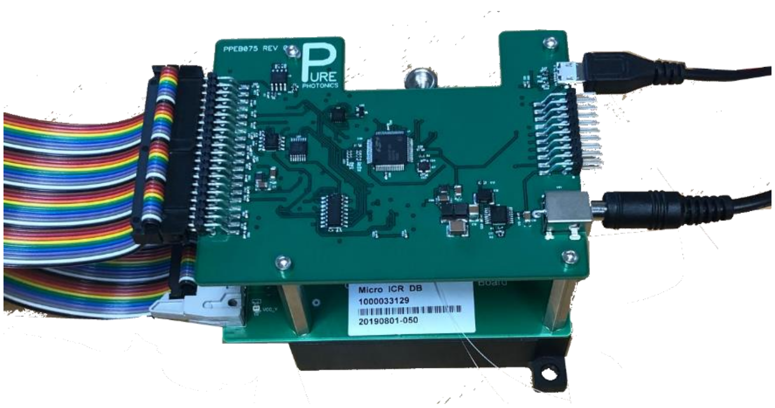
- Install the (micro-)ICR



\

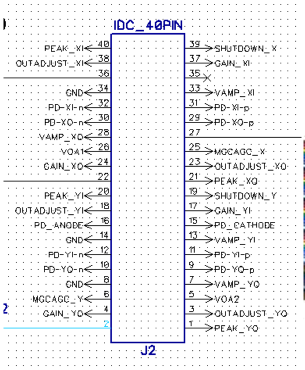
- Install the PPEB076 board on the stand-offs and use the 4 hex screws that were removed in the first step

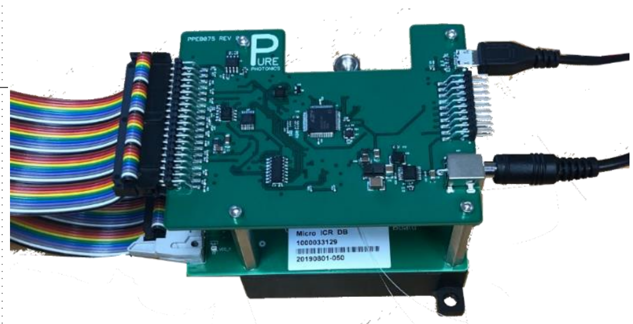
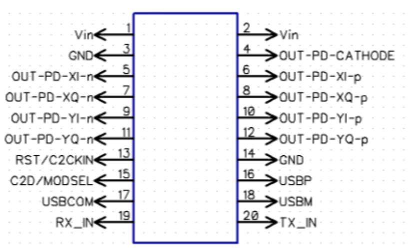
- Install the ribbon cable, USB connector (needs to be done before the power plug is inserted) and the power plug

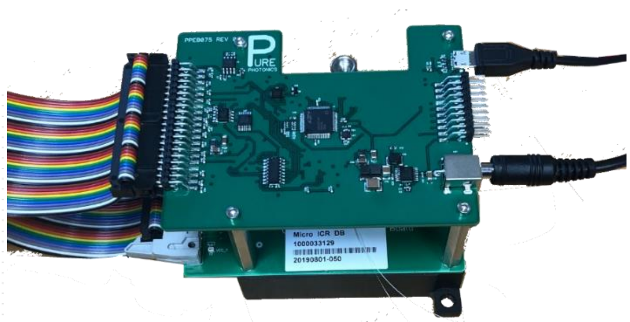


3. Electrical Connectors

The board has 2 IDC connectors

- 40 pin connector to connect to ICR. In the below image, this is the connector on the left. Pin 1 is on the top of the board closest to the observer (pin 39 is at the top of the board, closest to the Pure Photonics logo)
  - 


- 20 pin connector to connect to customer system (operation possible without using this). In the image below this connector is on the right. Pin 1 is on top of the board furthest from the observer (pin 19 is on the top of the board, next to the barrel plug).
  - 



In addition, there is a barrel plug power supply (**make sure that either the pins on the 20 pin connector are connected or this barrel plug. Never connect two power supplies**) with input voltage requirement 10-25V. Typical power consumption is less than 2W.

And there is a micro-USB communications port.

4. Communications Port

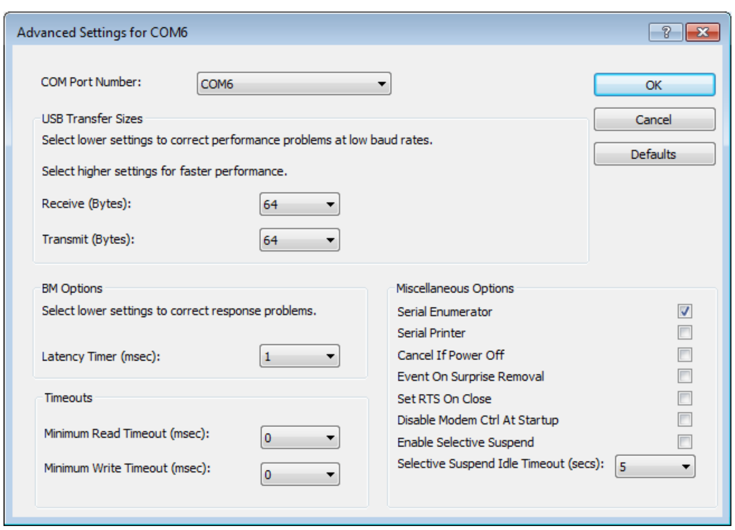
The ICR communications goes over the micro-USB interface. When connected to a computer the interface will install as a serial port (FTDI VCP – Virtual COM Port - driver required, if not already installed on computer).

The USB interface can be alternatively controlled through the pins on the input connector or can be overruled completely by a LVTTL RS-232 interface through the input connector. In either case, care needs to be taken that only 1 interface is connected at any time, to prevent contention.

The standard USB interface is not optimized for serial communications on the RS-232 interface. Due to differences in protocol and optimization algorithms for the USB port, it installs with non-optimal settings. For most operations, this is not a problem, however for a firmware upgrade a proper configuration is required.

We recommend to make the below changes when the USB interface gets registered. On every computer this is only needed once for each USB interface/device.

a. Open the 'Windows Control Manager'
b. Open 'Hardware and Sound'
c. Open 'Device Manager'
d. Find the COM ports
e. Right-click the USB serial port and select 'Properties'
f. Select the tab 'Port Settings' and click the 'Advanced' button
g. **Set the latency timer value to 1msec.**
   a. You can also select the COM-port designation here (we recommend port # <10).
   b. We recommend to set the 'USB transfer sizes' to the lowest possible setting (do this for both receive and transmit)
h. Close the windows and start using the device



5. Communications Protocol

The communication with the device follows the definition from the OIF MSA for tunable lasers. Basically 4 bytes are sent per command and 4 bytes are returned (a strict handshake model).

The OIF document can be found at https://www.oiforum.com/wp-content/uploads/2019/01/OIF-ITLA-MSA-01.3.pdf.

The user to module command is defined as below (more details in MSA, section 8 and 9):

31	30	29	28	27	Bits 26:0
Checksum				LstRsp	Command packet being framed

Inbound Byte 0

31	30	29	28	27	26	25	24
0x0 (To be defined by transport layer)					0x0		RW (R=0, W=1)

Inbound Byte 1

23	22	21	20	19	18	17	16
Register Number (0x00 – 0xff)							

Inbound Byte 2

15	14	13	12	11	10	9	8
Data 15:8							

Inbound Byte 3

7	6	5	4	3	2	1	0
Data 7:0							

The module to user command is defined as below (more details in MSA, section 8 and 9)

31	30	29	28	27	26	Bits 25:0
Checksum				CE	1	Response packet being framed

Outbound Byte 0

31	30	29	28	27	26	25	24
0x0 (To be defined by transport layer)						Status	

Outbound Byte 1

23	22	21	20	19	18	17	16
Register Number (0x00 – 0xff)							

Outbound Byte 2

15	14	13	12	11	10	9	8
Data 15:8							

Outbound Byte 3

7	6	5	4	3	2	1	0
Data 7:0							

Note the checksum in the first 4 bits of the first byte. The checksum is calculated as pe the OIF MSA protocol (for tunable lasers) and reproduced below.

```
unsigned char calcBIP4( unsigned char* data )   {
   int i;
   unsigned char bip8=(data[0]& 0x0f) ^ data[1] ^ data[2] ^ data[3];
   unsigned char bip4=((bip8 & 0xf0) >>4) ^ (bip8 & 0x0f);
   return bip4;
}
```

6. Registers

The following ‘tunable laser’ registers are available on the PPEB076

- 0x01 DevType
- 0x02 MFGR
- 0x03 Model
- 0x04 SerNo
- 0x06 Release
- 0x08 GenCfg
- 0x09 AEA-EAC
- 0x0A AEA-EA
- 0x0B AEA-EAR
- 0x0d IOCap
- 0x0E EAC
- 0x0F EA
- 0x10 EAR
- 0x13 LstResp
- 0x14 DLConfig
- 0x15 DLStatus
- 0x33 MCB

The following Registers are specific for the ICR Control

- 0x80 Enable/Disable TIA
- 0x81 Photo-diode mode (automatic/manual)
- 0x82 Photo-diode reading
- 0x83 ACG-MCG
- 0x84 Output Adjust setting
- 0x85 Gain Setting
- 0x86 Shutdown
- 0x87 Peak Voltage
- 0x88 VOA Setting
- 0x89 TIA current
- 0x8D Sets the target voltage over the photodiodes (mV)
- 0x8E Sets the hysteresis range for the photodiodes voltage (mV)
- 0x9A PD Resistance

The registers are described in more detail on the next page:

SEND

		WRITE																READ																
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TIA_Enable	0x80													TIA3	TIA2	TIA1	TIA0																	0 = Disable ; 1 = Enable
PD_Mode	0x81								PD	Yqp	Yqn	Yip	Yin	XQp	XQn	Xip	Xin																	0 = Manual; 1= Auto
PD_Value	0x82	Not writeable																CHANNEL																
MGC_AGC	0x83															Y	X																	0 = MGC; 1 = AGC
Output_Adjust	0x84	CHANNEL								Value								CHANNEL																0 - 255 for (0V - 3.3V)
Gain	0x85	CHANNEL								Value								CHANNEL																0 - 255 for (0V - 3.3V)
Shutdown	0x86															Y	X																	0 = Disable; 1 = Enable
Peak_Value	0x87	Not writeable																CHANNEL																
VOA_Voltage	0x88	Value																																0.01V (0 - 5V)
TIA_Currrent	0x89	Not writeable																														TIA#		TIA 0-3
PD Resistance	0x9A	CHANNEL												Value				CHANNEL																PD 0-8; value 0-255 (underlying resistor 100k=256)

RESPONSE

		WRITE																READ																
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TIA_Enable	0x80													TIA3	TIA2	TIA1	TIA0													TIA3	TIA2	TIA1	TIA0	0 = Disable ; 1 = Enable
PD_Mode	0x81								PD	Yqp	Yqn	Yip	Yin	XQp	XQn	Xip	Xin								PD	Yqp	Yqn	Yip	Yin	XQp	XQn	Xip	Xin	0 = Auto; 1 = Manual
PD_Value	0x82	Not writeable																Value																microAmp
MGC_AGC	0x83															Y	X															Y	X	0 = MGC; 1 = AGC
Output_Adjust	0x84	CHANNEL								Value																Value								0 - 255 for (0V - 3.3V)
Gain	0x85	CHANNEL								Value																Value								0 - 255 for (0V - 3.3V)
Shutdown	0x86															Y	X															Y	X	0 = Disable; 1 = Enable
Peak_Value	0x87	Not writeable																								Value								mV (2.5V on input corresponds to 900mV pkpk out)
VOA_Voltage	0x88	Value																Value																0.01V (0 - 5V)
TIA_Currrent	0x89	Not writeable																Value																0.01mA
PD Resistance	0x9A	CHANNEL												Value				CHANNEL												Value				PD 0-8; value 0-255 (underlying resistor 100k=256)
		Don't care																																
		Value required																																

Register 0x9A – PD resistance

The photo-diodes are reverse biased at about -5V. Between this voltage and the photodiode there is a variable resistor, reducing the bias voltage by a voltage equal to resistance x photo-current.

In the automatic mode, the bias voltage is kept at a value between 4.3 and 4.7V to ensure strong reverse biasing. This is accomplished by adjusting the resistance value which can take values between 0 and 100 kOhm. A read on register 0x9A gives the value of the resistor as return value * 100kOhm / 256.

In the manual mode, a write to register 0x9A can set the resistance value (at value * 100kOhm / 256). This is not recommended but can be helpful where you want to monitor the power variation and need to ensure that the divider in the equation (the resistance) is stable. In such cases we recommend to first have the auto mode get to the correct resistance value and then switch to manual mode.

Register 0x8D – target photodiode voltage

The voltage over the photodiodes is controlled by a variable resistance. In the automatic mode the resistance is adjusted to control the voltage over the photodiodes to a specific range (see 0x8E) around a target. The voltage target is set and read in mV in register 0x8D. It can be saved to permanent memory by the genCfg command. Note that the default target is 4V for class 20 devices. It is 3.3V for class 40 devices.

Register 0x8E – target photodiode voltage range

The range provides a hysteresis over which no additional action is taken to adjust the resistance values. The smaller this value, the more the resistance will be modified, which can result in additional noise.

7. Command Line Interface

The CLI is available for download on the Pure Photonics webside (under support). This program allows for command line communication with the ICR evaluation board. A special script is made available to implement the specific ICR command

For the older CLI version (v3.0.2)

The specific commands for the ICR can be loaded by saving the ‘ICR.py’ file in the CLI directory and then running the command execfile(‘ICR.py’). The code for the file is in the next section.

New CLI version (v3.2.0)

The CLI commands can be enabled by it.SetICR(True). After that the ICR commands are available. The commands are included in the CLI help function and in the manual.

8. Execfile code

```
readpacket=None
writepacket=None
connected=False

def ICRConnect(port=1,baudrate=9600):
    global readpacket,writepacket,connected
    it.connect(port,baudrate)
    it.mcb()
    readpacket=it.toModulePacket()
    it.mcb()
    writepacket=it.toModulePacket()
    writepacket.buffer('\x11\x81\x00\x00')
    connected=True

def ICRDisconnect():
    global connected
    it.disconnect()
    connected=False

def ICRTIA(value=-1):
    if not(connected):
        print 'Not connected'
        return
    if value==-1:
        readpacket.register(0x80)
        test=it.packet(readpacket)
        return test.data()
    else:
        writepacket.register(0x80)
        writepacket.data(value&0x000f)
        it.packet(writepacket)

def ICRPDMode(value=-1):
    if not(connected):
        print 'Not connected'
        return
    if value==-1:
        readpacket.register(0x81)
        test=it.packet(readpacket)
        return test.data()
    else:
        writepacket.register(0x81)
        writepacket.data(value&0x01ff)
        it.packet(writepacket)
```

```
def ICRPDValue(ch=0):
    if not(connected):
        print 'Not connected'
        return
    readpacket.register(0x82)
    readpacket.data(ch&0x0f)
    test=it.packet(readpacket)
    return test.data()

def ICRMGCAGC(value=-1):
    if not(connected):
        print 'Not connected'
        return
    if value==-1:
        readpacket.register(0x83)
        test=it.packet(readpacket)
        return test.data()
    else:
        writepacket.register(0x83)
        writepacket.data(value&0x0003)
        it.packet(writepacket)

def ICROutputAdjust(ch=0,volts=-1):
    if not(connected):
        print 'Not connected'
        return
    if volts==-1:
        readpacket.register(0x84)
        readpacket.data(ch*256*16)
        test=it.packet(readpacket)
        return test.data()*3.3/256
    else:
        if volts>3.3:volts=3.3
        writepacket.register(0x84)
        writepacket.data(ch*256*16+int(256*volts*1.0/3.3))
        it.packet(writepacket)

def ICRGain(ch=0,volts=-1):
    if not(connected):
        print 'Not connected'
        return
    if volts==-1:
        readpacket.register(0x85)
        readpacket.data(256*16*ch)
        test=it.packet(readpacket)
        return test.data()*3.3/256
```

```
    else:
        writepacket.register(0x85)
        if volts>3.3:volts=3.3
        writepacket.data(ch*256*16+int(256*volts*1.0/3.3))
        it.packet(writepacket)

def ICRShutdown(value=-1):
    if not(connected):
        print 'Not connected'
        return
    if value==-1:
        readpacket.register(0x86)
        test=it.packet(readpacket)
        return test.data()
    else:
        writepacket.register(0x86)
        writepacket.data(value&0x03)
        it.packet(writepacket)

def ICRPeakV(ch=0):
    if not(connected):
        print 'Not connected'
        return
    readpacket.register(0x87)
    readpacket.data(ch*256*16)
    test=it.packet(readpacket)
    return test.data()

def ICRVOA(value=-1):
    if not(connected):
        print 'Not connected'
        return
    if value==-1:
        readpacket.register(0x88)
        test=it.packet(readpacket)
        return test.data()
    else:
        writepacket.register(0x88)
        writepacket.data(value)
        it.packet(writepacket)
```

9. Firmware Upgrade

The Command Line Interface is a tool to directly access the registers of the tunable laser. The CLI is available for download on the Pure Photonics website, under the download section. The zip file needs to be downloaded and extracted to a separate directory. In the directory there will be a .exe file to run the program.

Note that the firmware is serial number specific. Please make sure you use the correct firmware.

Use the following sequence to perform firmware upgrade:

- it.connect(1,9600)
  - The first parameter is the COM port number. This may vary dependent on your configuration
  - The second parameter is the current baudrate of the device. Most devices start up with baudrate 9600
- it.release()
  - make sure that you get a response here. If not, something is seriously wrong and more trouble shooting is required. Contact Pure Photonics.
  - If the response is FW 0.0.0, then the temporary firmware version is active, indicating that the previous firmware upgrade did not terminate as intended.
- it.baudrate(115200)
  - 115200 is the highest available baudrate, resulting in the fastest upgrade
- it.upgrade('application',r'c:\.....\.....ray')
  - The item within parenthesis is the path to the .ray file that you wish to upload

After completion of the upgrade the interface will say: '*Seconds elapsed 140. Init_Run OK'.*

It may be that the unit needs to be configured after upgrade. To do this, follow the following sequence in the CLI:

- it.connect(x,9600)
- it.channel(1)
- test=it.toModulePacket()
- test.register(0x5e)
- test.data(1)
- it.packet(test)