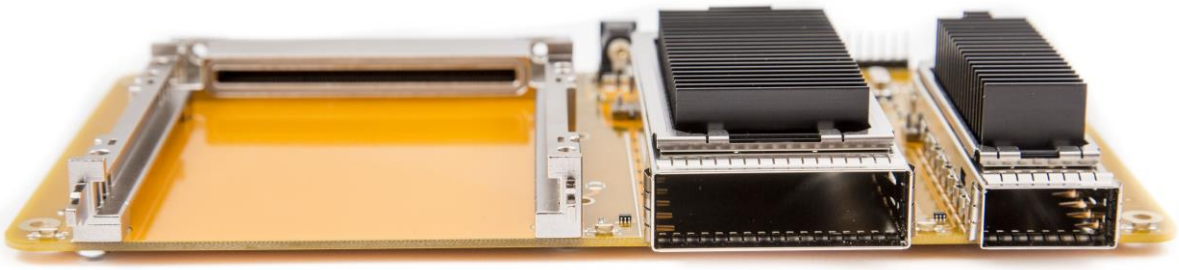
OpticalProgrammer.com
PROGRAMMING SOLUTIONS FOR OPTICAL MODULES

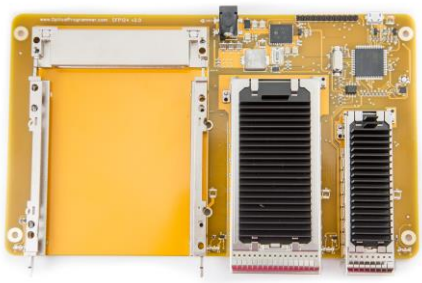
Tel: +44 750 200 4611 14 Brandon Grove, Stoke-on-Trent. ST4 2EW. UNITED KINGDOM
Email: sales@opticalprogrammer.com Web: www.opticalprogrammer.com
UK VAT: GB201699514 EORI: GB201699514000
OpticalProgrammer.com is a brand name of Ktizo Technology Solutions

CFP 1-2-4 Programmer

Datasheet

v2.2 May 2018
PCB v2.1





Product photographs for illustration only

Contents

Revision History

v2.2

- Page 9: Added MDIO speed switch.
- Page 12: GPIO11 was inverted behaviour – now corrected.
- Page 13: pin 11 is now connected to GND.
- Page 16: Added note about pull-ups.
- Page 23: Updated mechanical drawing and added note about rubber feet.

v2.1

- Changes to reflect v2.1 of the software, including new file formats.
- Minor formatting changes.

v2.0

- Various formatting changes.
- Page 1: Updated product photo.
- Page 5: Added section on demo software.
- Page 11: Added section on header pins
- Page 12: Moved the section about SDK installation.
- Page 21: Added photo to section about removing heatsinks.

v1.1

- Page 4: Removed sentence about detecting the presence of the 5V supply with the PWRGOOD flag.
- Page 6: Added footnote about use of the PWRGOOD flag.
- Page 6: Added footnote about power for the LEDs.
- Page 11: Added new sections about performing I^2C transactions.
- Page 13: Added a section describing how to read current and voltage.

v1.0

- Original issue

A note about I^2C addresses

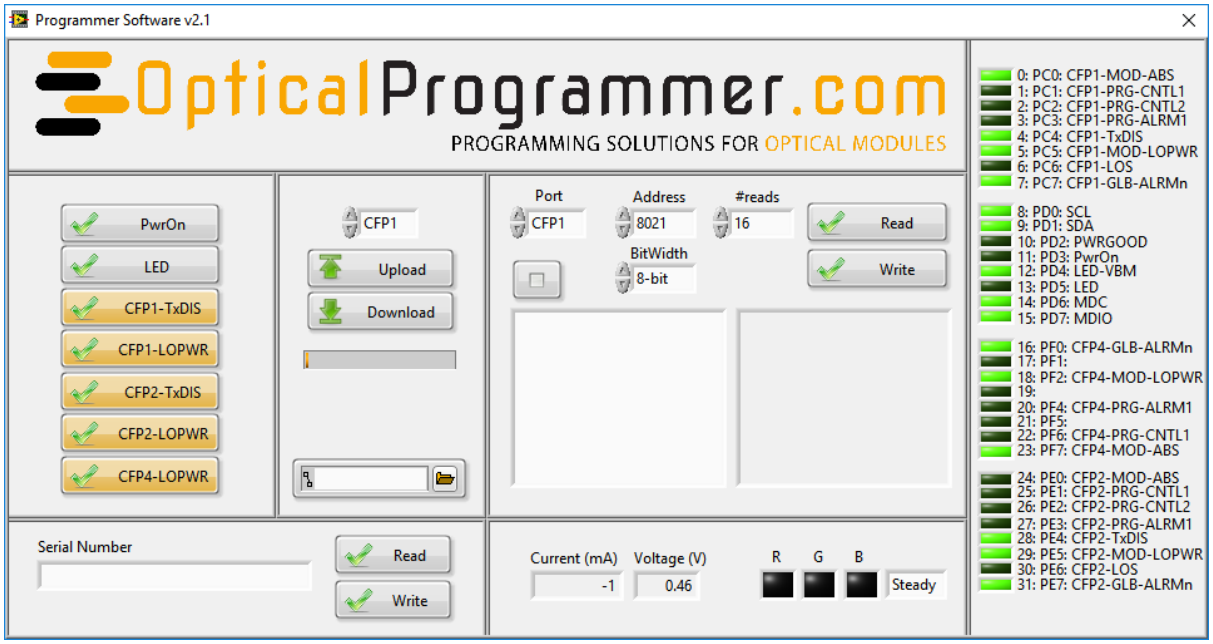
All I^2C addresses in this document are in the 7-bit format. The eighth bit determines the read/write status of the command.

Key Features

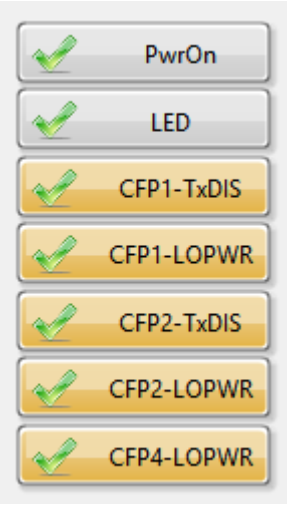
- Designed to program the EEPROM / firmware of CFP1, CFP2 and CFP4 optical transceivers.
- Allows virtually all CFP variants to be fully powered-up.
- Comprehensive demo software provided.
- Available with short lead-time “off the shelf”.

Demo Software

The software installer will create a shortcut under OpticalProgrammer.com.



Input controls



PwrOn controls the power to the CFPs. LED controls the LED to the right of the CFP4 transceiver. The other buttons control the status of various CFP hardware control lines.

Voltage & current monitoring

These monitor the power to the CFP slots.

Upload / Download EEPROM

Use the Upload button to load a data file into the CFP. A dialog box will open, where the file to be uploaded is specified.

Use the Download button to read the CFP EEPROM contents to a file. To specify which addresses to read, edit the Config.txt file. The location of the file is specified with the file browse control.

A note about 8 and 16-bit-width registers

Some registers use a 16-bit value, others use an 8-bit value, as determined by the CFP MSA.

Read / Write

File format for Config.txt file

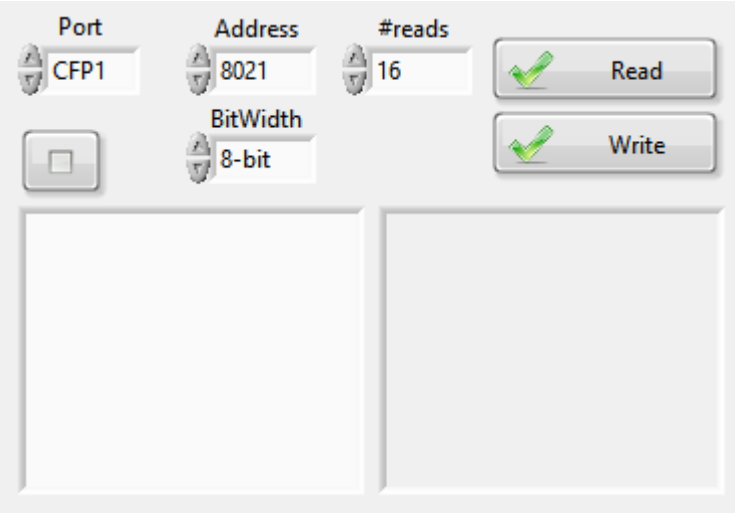
```
Address(hex),BitWidth(dec),#Bytes(dec)
8000,8,16
8080,8,32
AC00,16,1
END
```

With this file, when the Download button is pushed, the following reads are done:
16 bytes, each in 8-bit width format, starting at address 8000h
32 bytes, each in 8-bit width format, starting at address 8080h
1 byte, with 16-bit width, starting at address AC00h

File format for files to upload

```
Address(hex),BitWidth(dec),Data(hex)
A000,16,2C568F8D
8000,8,30313233
8080,8,414243444546
END
```

This file would upload the following:
Address A000h, data = 2C56h
Address A001h, data = 8F8Dh
Address 8000h, data = 30h
Address 8001h, data = 31h
Address 8002h, data = 32h
Address 8003h, data = 33h
Address 8080h, data = 41h
Address 8081h, data = 42h
Address 8082h, data = 43h
Address 8083h, data = 44h
Address 8084h, data = 45h
Address 8085h, data = 46h



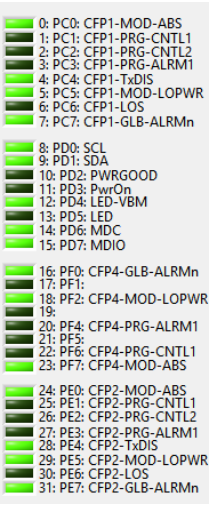
This section is for reading and writing individual bytes in the CFP EEPROM.

To read, specify the address and # of reads required.

To write, enter the hex data in the left-hand box – as many bytes as required, the address, and specify the BitWidth.

The data can be cleared using the clear button.

Raw control hardware data



The CFP 1-2-4 board allows read and write control of many hardware lines. The current status is shown here.

Read / Write serial number

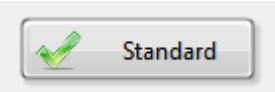
The board serial number can be read here. It is not write protected, and after a warning, it can be changed.

RGB LED controls

With the demo software, the three colours can be individually selected, and 2 flash modes, plus stead can be selected.

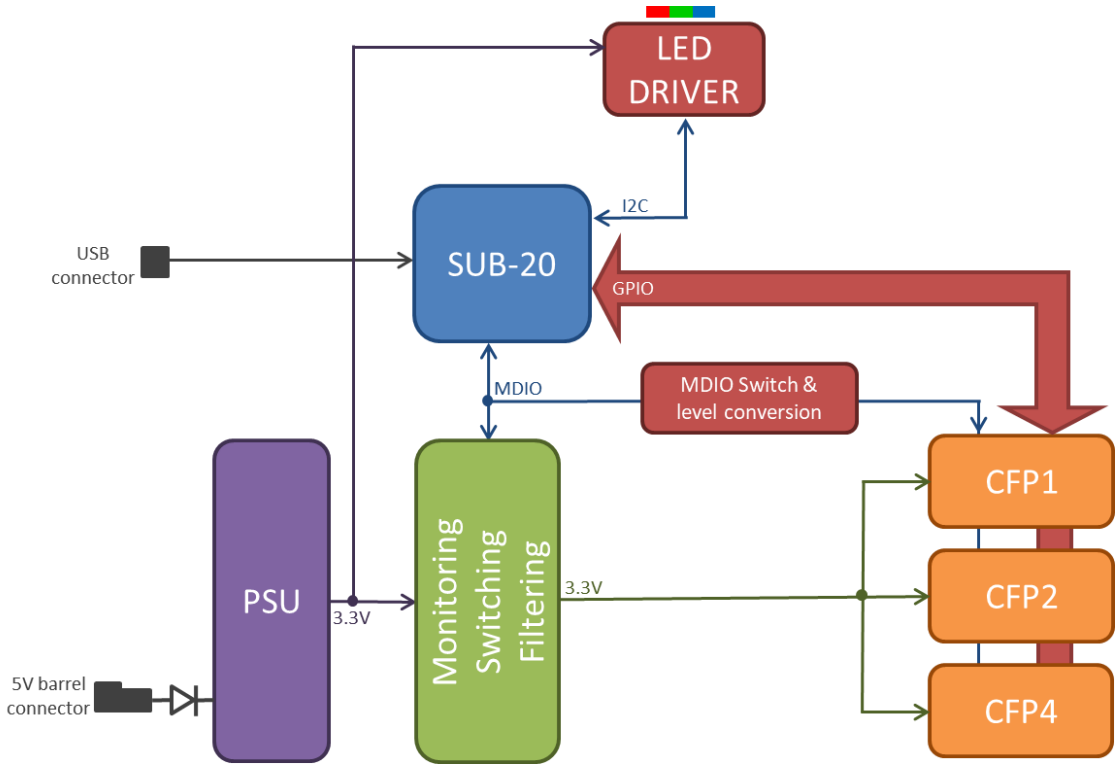
With the current demo software, changing brightness is not possible although this can be done with custom software.

MDIO clock speed



Board version 2.0 and below has a fixed MDIO bus speed of ~1.4MHz. Board version 2.1 and above also have a hi-speed option of ~4.0MHz.

Hardware Overview



Powering the board

The board is powered from an external 5V DC power supply, via a 2.5mm barrel jack.

In any event, the EEPROM and GPIO are always powered from the USB supply.
All three supplies have a 2.7A hard current limit for safety purposes.

The SUB-20 microcontroller is powered from the USB supply and therefore communications with the board can still occur when the external 5V supply is not connected.

Internal PSU

Internal power supply characteristics: Regulated 3.3V supply, compatible with the CFP MSAs. Up to 19.8W (6.0A) can be supplied. This is enough for:
CFP1: Class 1 & Class 2 transceivers
CFP2: All classes
CFP4: All classes

Note that all three CFP slots can be set to run in low-power mode, which allows the EEPROM to be programmed, but does not necessarily allow the laser to be powered on.

We recommend that *PWRGOOD* is polled regularly. If the power supply is in overload for whatever reason, indicated by *PWRGOOD* = low, the power supply should be disabled by setting *PwrOn* to low. If an over-current fault persists, the board will enter a thermal-protection mode and could power-cycle until the fault is removed.

Power monitoring & switching

There is a power monitor in the programmer, which monitors the power supply to the module slots. The current drawn by the pull-up resistors associated with each of the slots are included in the current measurement. The power to all three module slots is switched together, via a GPIO line *PwrOn.*

The power monitor is at I^2C slave address 0x40.

MDIO Switch and Level Conversion

A switch enables the MDIO bus to the 3 module slots, but only when the power is applied to the slots. This arrangement prevents unpowered modules from locking up the bus, which can happen with some optical modules.

This block also voltage shifts the SUB-20's native 3.3V logic to the 1.8V LVTTL that the CFP MSA requires.

RGB LED drivers

The LED driver is a constant current device, ensuring long and consistent LED life. Each LED can be turned on or off independently, and the brightness set. The driver is also capable of setting a large number of pulsating modes. See the software section for more details.

LED-VBM indicates when a pulse cycle is about to start. Since this will be when the LED is off, this is a good time to change colours, if abrupt changes are to be avoided. For more information, see the LED controller datasheet: http://www.issi.com/WW/pdf/31FL3193.pdf

Module slots

Each module slot has an associated yellow LED.

If more than one optical module is inserted, there is a possibility of overloading the power supplies. Therefore, the software should monitor the *CFP1-Mod-Abs*, *CFP2-Mod-Abs*, and *CFP4-Mod-Abs* GPIO lines and should only enable *PwrOn* if one and only one module is inserted.

The Port Address is configured as follows:
CFP1: Port Address = 0
CFP2: Port Address = 1
CFP4: Port Address = 2

EEPROM

The serial number of the board is programmed into the SUB-20's internal EEPROM. The rest of the EEPROM is available for read/write. NB. The pre-programmed serial number is not write-protected.

GPIO Lines

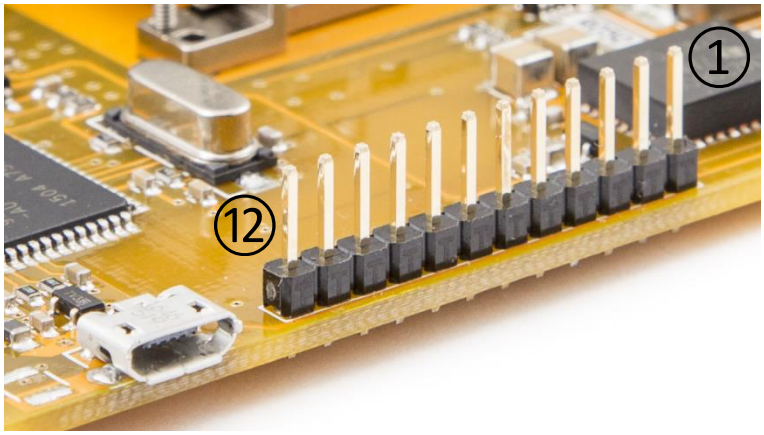
Before attempting to read or write, the ports must be configured in software.

GPIO	NAME	I/O	FN	DETAILS
0	CFP1-MOD-ABS	IN	CFP1	0 = CFP module inserted, 1 = not inserted
1	CFP1-PRG-CNTRL1	OUT	CFP1	See CFP MSA for details of use
2	CFP1-PRG-CNTRL2	OUT	CFP1	See CFP MSA for details of use
3	CFP1-PRG-ALRM1	IN	CFP1	See CFP MSA for details of use
4	CFP1-TxDIS	OUT	CFP1	See CFP MSA for details of use
5	CFP1-MOD-LOPWR	OUT	CFP1	0 = Full power mode, 1 = low power mode
6	CFP1-LOS	IN	CFP1	See CFP MSA for details of use
7	CFP1-GLB-ALRMn	IN	CFP1	See CFP MSA for details of use
8	Used internally (SCL)	I/O	SYSTEM	
9	Used internally (SDA)	I/O	SYSTEM	
10	PWRGOOD	IN	SYSTEM	0 = PSU fault, 1 = PSU OK[1]
11	PwrOn	OUT	SYSTEM	0 = CFP power off, 1 = CFP power on
12	LED-VBM	IN	SYSTEM	
13	LED	OUT	SYSTEM	0 = Green LED off, 1 = LED on[2]
14	Used internally (MDC)	I/O	SYSTEM	
15	Used internally (MDIO)	I/O	SYSTEM	
16	CFP4-GLB-ALRMn	IN	CFP4	See CFP MSA for details of use
17	Not used	OUT		
18	CFP4-MOD-LOPWR	OUT	CFP4	0 = Full power mode, 1 = low power mode
19	Not used	OUT		
20	CFP4-PRG-ALRM1	IN	CFP4	See CFP MSA for details of use
21	Not used	OUT		
22	CFP4-PRG-CNTRL1	OUT	CFP4	See CFP MSA for details of use
23	CFP4-MOD-ABS	IN	CFP4	0 = CFP module inserted, 1 = not inserted
24	CFP2-MOD-ABS	IN	CFP2	0 = CFP module inserted, 1 = not inserted
25	CFP2-PRG-CNTRL1	OUT	CFP2	See CFP MSA for details of use
26	CFP2-PRG-CNTRL2	OUT	CFP2	See CFP MSA for details of use
27	CFP2-PRG-ALRM1	IN	CFP2	See CFP MSA for details of use
28	CFP2-TxDIS	OUT	CFP2	See CFP MSA for details of use
29	CFP2-MOD-LOPWR	OUT	CFP2	0 = Full power mode, 1 = low power mode
30	CFP2-LOS	IN	CFP2	See CFP MSA for details of use
31	CFP2-GLB-ALRMn	IN	CFP2	See CFP MSA for details of use

[1] If the external 5V supply is entirely absent, PWRGOOD will be high, indicating no fault. It is not possible to use this pin to determine whether or not the external supply is present.

[2] The power for this LED is derived from the USB socket. Hence it will function even when the external 5V supply is absent. The other LEDs will only function when the external 5V supply is used.

Header pins



A 0.1” header is provided with access to the following connections:

Pin #	Connection
1	5V0
2	3V3
3	1V2
4	SCL
5	SDA
6	MDIO (to microcontroller)
7	MDC (to microcontroller)
8	MDIO (to CFP)
9	MDC (to CFP)
10	PwrOn
11	GND (N/C in board v2.0 and below)
12	GND

SDK Installation

At the heart of the CFP 1-2-4 board is a SUB-20 protocol converter. This provides the necessary interfacing between USB and the MDIO, I^2C and GPIO busses.

SUB-20 requires a USB driver:
32-bit: http://www.xdimax.net/download/SUB-20-driver-120622-x32.exe
64-bit: http://www.xdimax.net/download/SUB-20-driver-120622-x64.exe

After installing the driver, plug in a USB lead. Windows should automatically find the correct drivers and complete the installation.

A Software Development Kit (SDK) is also available:
http://www.xdimax.net/download/SUB-20-151016.exe

Linux and comprehensive LabVIEW drivers are also available.

DLL

All communications with the board are achieved through the Sub20.dll, which is found in C:\Windows\System32 after the drivers have been installed.
The coding examples here depend on libsub.h, which is found in Program Files > SUB-20

Further instructions

Various software functions are described in this document. For further and more detailed instructions, please see the SUB-20 manufacturer's instruction manual:
http://www.xdimax.com/sub20/doc/sub20-man.pdf

Command Set

sub_open

Opens SUB-20 device.

Usage
int **sub_open**(NULL)

Parameters
None

Return value
On success, function returns sub_handle that should be used in all subsequent calls to SUB-20 API functions. If successful, sub_handle is always non-zero. If unsuccessful, the function returns NULL.

Example of opening the SUB-20 device

```
handle = sub_open(0);
if( !handle )
{
printf("sub_open: %s\n", sub_strerror(sub_errno));
return -1;
}
```

sub_close

Closes references to the SUB-20

Usage
int **sub_close**(sub_handle hndl)

Parameters
hndl – obtained from **sub_open**

Return value
On success, the function returns 0. Any other value indicates an error.

sub_eep_read

Reads the SUB-20 EEPROM, which includes the board serial number.

Usage
int **sub_eep_read**(sub_handle hndl, int addr, char* buf, int sz)

Parameters
hndl – obtained from **sub_open**
addr – read start address
buf – buffer to store read data
sz – read size, max 64 bytes.

Return value
On success, the function returns 0. Any other value indicates an error.

A read of 23 characters from address 0, will yield a 23-character hex-encoded ASCII string **wwyyvvmbnnnnnpppppppppp,** where:

ww is week number of manufacture
yy is year of manufacture
vv is major hardware version
m is minor hardware version
b is hardware build number
nnnnn is the serial number
pppppppppp is the product name, left justified with spaces (E.g. “CFP124 ”)

sub_gpio_config

Configure GPIO lines as input or output.

Usage
int **sub_gpio_config**(sub_handle hndl, int set, int* get, int mask)

Parameters
hndl – obtained from **sub_open**
set - Bits 0..31 of this parameter correspond to the 32 GPIO lines available on the SUB20. If GPIOn configuration bit is "1" then GPIOn direction is output, otherwise it is input.
*get - Pointer to store current GPIO configuration read from SUB-20.
mask – Each bit in set parameter will take effect only if the corresponding mask bit is "1". With mask=0, this function will read the current GPIO configuration.

Return value
On success, the function returns 0. Any other value indicates an error.

Example of configuring the SUB20 for use with this board:
rv = sub_gpio_config(hndl, 0x366EEB36, &config, 0xFFFFFFFF);

A note on pull-ups: It is recommended to enable weak pull-ups for all inputs. This is to prevent spurious readings when no optics are inserted. When a GPIO configured as an input is set to “1” using sub_gpio_write, the pull-up is enabled.

sub_gpio_read

Read GPIO input status.
See Table 1 for details of which bits are used on this board.

Usage
int **sub_gpio_read**(sub_handle hndl, int* get)

Parameters
hndl – obtained from **sub_open**
*get - Pointer to store inputs status. Bits 0..31 of *get correspond to the 32 GPIO lines available on the SUB20.

Return value
On success, the function returns 0. Any other value indicates an error.

sub_gpio_write

Set GPIO output status.
See Table 1 for details of which bits are used on this board.

Usage
int **sub_gpio_write** sub_handle hndl, int set, int* get, int mask)

Parameters
hndl – obtained from **sub_open**
set - Bits 0..31 of this parameter correspond to the 32 GPIO lines available on the SUB20 card. If GPIOn configuration bit is "1" then GPIOn direction is output, otherwise it is input.
*get - Pointer to store current GPIO configuration read from SUB-20.
mask – Each bit in set parameter will take effect only if the corresponding mask bit is "1".

Return value
On success, the function returns 0. Any other value indicates an error.

Example of writing to the GPIO lines to enable power to the modules (PwrOn, GPIO11, ON)
rv = sub_gpio_write(hndl, 0x00000800, &config, 0x00000800);

Example of writing to the GPIO lines to disable power to the modules (PwrOn, GPIO11, OFF)
rv = sub_gpio_write(hndl, 0x00000000, &config, 0x00000800);

sub_mdio45

Generate IEEE 802.3 Clause 45 MDIO frame to communicate with a CFP module.

Usage
int **sub_mdio45**(sub_handle hndl, int op, int prtad, int devad, int data, int* content)

Parameters
hndl – obtained from **sub_open**

op – operation code:
SUB_MDIO45_ADDR (0x00) is the ADDRESS operation
SUB_MDIO45_WRITE (0x01) is the WRITE operation
SUB_MDIO45_PRIA (0x02) is the POST-READ-INCREMENT-ADDRESS operation
SUB_MDIO45_READ (0x03) is the READ operation
prtad, the “Port Address” (see “Module Slots” section)
devad, the “Device Address” is 0x01 for all CFP modules
data - 16 bit address or data for ADDRESS or WRITE operation
*content - 16 bit register content placeholder for READ or POST-READ-INCREMENT-ADDRESS operation

Return value
On success, the function returns 0. Any other value indicates an error.

sub_mdio_xfer_ex

Generate multiple IEEE 802.3 Clause 45 MDIO frames to communicate with a CFP module.

Usage
int **sub_mdio_xfer_ex**(sub_handle hndl, int channel, int count, union sub_mdio_frame* mdios)

Parameters
hndl – obtained from **sub_open**
channel – set to either 0x80 (4MHz clock) or 0x00 (1.5MHz clock).
count – number of frames to generate (up to 15)
*mdios - array of **count** sub_mdio_frame unions

```
union sub_mdio_frame
{
        struct
        {
          int op;
          int prtad;
          int devad;
          int data;
        }clause45;
};
```

op – operation code:
SUB_MDIO45_ADDR (0x00) is the ADDRESS operation
SUB_MDIO45_WRITE (0x01) is the WRITE operation
SUB_MDIO45_PRIA (0x02) is the POST-READ-INCREMENT-ADDRESS operation
SUB_MDIO45_READ (0x03) is the READ operation
prtad, the “Port Address” (see “Module Slots” section)
devad, the “Device Address” is 0x01 for all CFP modules

For READ and POST-READ-INCREMENT-ADDRESS operations clause45.data will be filled with data read from the CFP module.

Return value
On success, the function returns 0. Any other value indicates an error.

Example of reading 1 byte from CFP4, address 0x8021

```
union sub_mdio_frame mdios[2];
mdios[0].clause45.op = 00;              // ADDRESS operation
mdios[0].clause45.prtad = 0x02;      //CFP4
mdios[0].clause45.devad = 0x01;
mdios[0].clause45.data = 0x8021;
mdios[1].clause45.op = 0x03;           //READ operation
mdios[1].clause45.prtad = 0x02;
mdios[1].clause45.devad = 0x01;

rc = sub_mdio_xfer( hndl, 2, mdios );
```

sub_i2c_freq

Sets I2C clock frequency, used for communication with the LED driver.

Usage

int **sub_i2c_freq**(sub_handle hndl, int* freq)

Parameters

hndl – obtained from **sub_open**
*freq - Desired frequency – normally 100,000. On return will be filled with resulting frequency

Return value

On success, the function returns 0. Any other value indicates an error.

sub_i2c_write

This function does an optional dummy write to set the memory address, then does a write.

Usage

int sub_i2c_write(sub_handle hndl, int sa, int ma, int ma_sz, char* buf, int sz)

Parameters

hndl – obtained from **sub_open.**
sa - Slave Address – 0x68 (RGB LED), or 0x40 (power monitor).
ma - Memory Address – the address you want to write to.
ma_sz - Memory Address size bytes – #bytes used by ma. If set to zero, the dummy write is skipped and the write is to the previously used memory location.
*buf - Buffer for data to be written.
sz – Write byte count – how many bytes you want to write (maximum 64bytes).

Return value

On success function returns 0. Any other value indicates an error.

sub_i2c_read

This function does a dummy write to set the memory address, then does a read.

Usage

int sub_i2c_read(sub_handle hndl, int sa, int ma, int ma_sz, char* buf, int sz)

Parameters
hndl – obtained from **sub_open.**
sa - Slave Address – 0x68 (RGB LED), or 0x40 (power monitor).
ma - Memory Address – the address you want to write to.
ma_sz - Memory Address size bytes – #bytes used by ma. If set to zero, the dummy write is skipped and the read is from the previously used memory location.
*buf - Buffer for data to be written.
sz – Read byte count – how many bytes you want to read (maximum 64bytes).

Return value
On success function returns 0. Any other value indicates an error.

Programming with the RGB LED

The LED driver can be complicated to set up. This simplified guide is a starting point for accessing the main features. Some settings could potentially damage the LEDs, so proceed with caution. Please ask if you are unsure.

This table shows the required address locations to set up the LEDs. Five typical setups are shown. The LED driver chip is write-only by I^2C.

Address	Function	Green steady	Red steady	Blue steady	Blue slow pulse	Blue fast pulse
0x2F	Reset	0x00	0x00	0x00	0x00	0x00
0x00	Shutdown	0x20	0x20	0x20	0x20	0x20
0x02	Mode	0x00	0x00	0x00	0x20	0x20
0x03	Current limit (17.5mA – do not change)	0x10	0x10	0x10	0x10	0x10
0x04	Green brightness (00 to FF)	0x20	0x00	0x00	0x00	0x00
0x05	Red brightness (00 to FF)	0x00	0x30	0x00	0x00	0x00
0x06	Blue brightness (00 to FF)	0x00	0x00	0x30	0x30	0x30
0x07	Load brightness data	0x00	0x00	0x00	0x00	0x00
0x10	Pulse setting	0x00	0x00	0x00	0x84	0x40
0x11	Pulse setting	0x00	0x00	0x00	0x84	0x40
0x12	Pulse setting	0x00	0x00	0x00	0x84	0x40
0x16	Pulse setting	0x00	0x00	0x00	0x64	0x20
0x17	Pulse setting	0x00	0x00	0x00	0x64	0x20
0x18	Pulse setting	0x00	0x00	0x00	0x64	0x20
0x1C	Load pulse data	0x00	0x00	0x00	0x00	0x00

Example

To set the LED to “Blue slow pulse”:
Do 15 separate I^2C writes as follows…
sa =0x68, *ma* = 0x2F, *buf* = 0x00
sa =0x68, *ma* = 0x00, *buf* = 0x20
sa =0x68, *ma* = 0x02, *buf* = 0x20
sa =0x68, *ma* = 0x03, *buf* = 0x10
sa =0x68, *ma* = 0x04, *buf* = 0x00
sa =0x68, *ma* = 0x05, *buf* = 0x00
sa =0x68, *ma* = 0x06, *buf* = 0x30

sa =0x68, *ma* = 0x07, *buf* = 0x00
sa =0x68, *ma* = 0x10, *buf* = 0x84
sa =0x68, *ma* = 0x11, *buf* = 0x84
sa =0x68, *ma* = 0x12, *buf* = 0x84
sa =0x68, *ma* = 0x16, *buf* = 0x64
sa =0x68, *ma* = 0x17, *buf* = 0x64

VBM Outputs

Each LED driver has an output called VBM. This indicates when a pulse cycle is about to start. Since this will be when the LED is off, it is a good time to change colours, if abrupt changes are to be avoided. For more information, see the LED controller datasheet: http://www.issi.com/WW/pdf/31FL3193.pdf

Read power monitors

Current

Use sub_i2c_read with:
- hndl – obtained from **sub_open**
- sa = 0x40
- ma = 0x01
- ma_sz = 1
- sz =2 (signed 16)

Multiply value obtained by 0.2 to obtain the current in mA

Voltage

Use sub_i2c_read with:
- hndl – obtained from **sub_open**
- sa = 0x40
- ma = 0x02
- ma_sz = 1
- sz =2 (signed 16)

Bit-shift the value obtained right, 3 times. Multiply by 0.004 to obtain the voltage in V

Averaging

By default, the power monitors do not do any averaging. To set averaging to x128:
Use sub_i2c_write with:
- hndl – obtained from **sub_open**
- sa = 0x40
- ma = 0x00
- ma_sz = 1
- *buf = 0x3FFF
- sz = 2 (signed 16)

SUB20 Tool

The manufacturer of the SUB-20 card provides a test tool which is useful for checking the hardware is working properly. Once the SDK is installed, it will appear in Start>SUB20>Sub Tool

Configuration

GPIO Lines

- On the GPIO tab, set the configuration to 366EEB36
- Press the “Set Config” button.
- This sets up the various GPIO lines to be read or write as appropriate.

Turn on power to modules

- Set “Val” to 00000800 and “Mask” to 00000800
- Press “Write”

CFP communication

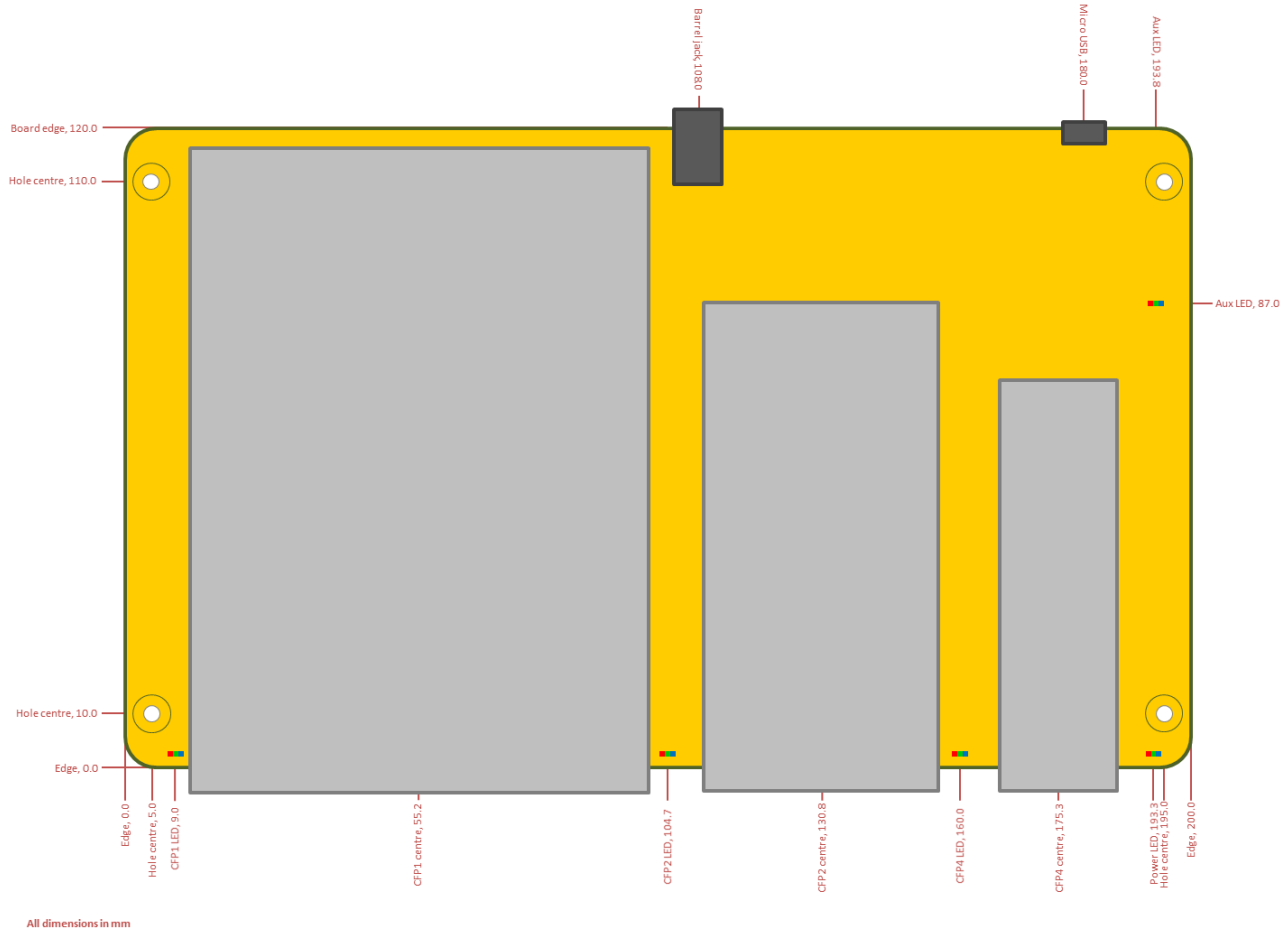
Configure and set address to read from (CFP1)

- Go to the MDIO tab
- All CFPs use “Clause45” communications, with “Device Address” = 1
- Set “Port Address” to 0 (CFP1)
- The “Channel” number should always be set to 0
- The “CFP MSA” box can be checked if needed. This increases the bus speed from about 1.5MHz to 4MHz
- Set the “Address” to 0x8021 (or whatever address you want to read from)
- Set the drop-down function to “Address”
- Press “Execute”

Read data

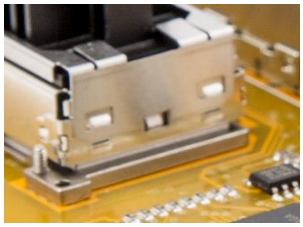
- Set the drop-down function to “Post-Read-Inc”
- Press “Execute” as many times as consecutive addresses you wish to read from.

Mechanical design



Overall board size: 120.0mm x 200.0mm
Board thickness: 1.6mm.
Highest component is the CFP2 cage, which is 16.3mm from the top surface of the board.
The board is supplied with 5x rubber feet.

Cage heatsinks



The board is supplied with a heatsink attached to the CFP2 and CFP4 cages. This can be removed if it is not required. The heatsink bracket can be removed by releasing the two tabs at the rear of the cage, while gently pulling the bracket upwards.

Operating conditions

The board will operate between 0°C and 40°C in non-condensing climatic conditions.

The Small Print

Terms and Conditions of Sale:
In these conditions, “we” and “us” refers to Ktizo Technology Solutions and its associated brand name OpticalProgrammer.com. In placing an order with us, you agree to be bound by the terms and conditions stated herein. The provisions set forth herein are for the sole benefit of the parties hereto, and confer no rights benefits or claims upon any person or entity not a party hereto.

Availability and Pricing
Specifications, availability and pricing are subject to change without notice. Orders are not binding upon us until accepted by us, and until any specified initial payments are received. Prices listed are in British Pounds (GBP). We reserve the right to refuse service, terminate accounts, or cancel orders at our sole discretion.

Payment terms
Payment is due 28 days from the invoice date unless otherwise stated on the invoice.
Payment options are shown on invoices. Other options may be available, on agreement with us.
Title to all goods or services is retained by us until full and final payment is received.

International Orders
Export orders are accepted on the basis of payment in advance unless agreed otherwise by us. We may also require an initial payment, that is, a proportion of the balance in advance of manufacture. Prices are quoted FCA our company office in Stoke-on-Trent, UK in accordance with Incoterms 2010 and do not include insurance, freight, brokerage, duty or taxes, unless otherwise stated.

In placing an order with us, you agree to comply with all applicable export laws, restrictions and regulations of the United Kingdom or foreign agencies or authorities, and shall not export, or transfer for the purpose of re-export, any product to any prohibited or embargoed country or to any denied, blocked, or designated person or entity as mentioned in any United Kingdom or foreign law or regulation. You warrant that you are not prohibited by law from purchasing the products or services hereunder. You shall be responsible to obtain any license to export, re-export or import as may be required.

Cancellation
In the event of cancellation by after ordering, we reserve the right to invoice for a reasonable proportion of the total order value that was originally quoted.

Warranty
Our products are covered by a one year warranty; this covers parts and labour based on the goods being returned to our address, below.

Intellectual Property
We retain all intellectual property rights associated with the design and manufacture of any goods or equipment supplied under this agreement.

Liability
We will not be liable for any loss or damage to any goods or equipment on loan to us. We specifically disclaim any and all warranties, either express or implied, with regards to any licensed products. No warranty will apply if products supplied hereunder are in any way altered or modified after delivery. In no event shall we be liable for any damages, including but not limited to loss of profits, revenues, business, goodwill, data, injury, interruption of business, nor for incidental or consequential loss or fitness of purpose damages related to this agreement.
If we provide you with advice, training, applications support, or other assistance which concern any products supplied hereunder, or any equipment, system or the like in which the product may be installed, our giving of such advice or assistance will not subject us to any liability, whether based on contract, warranty, tort (including negligence) or other grounds.