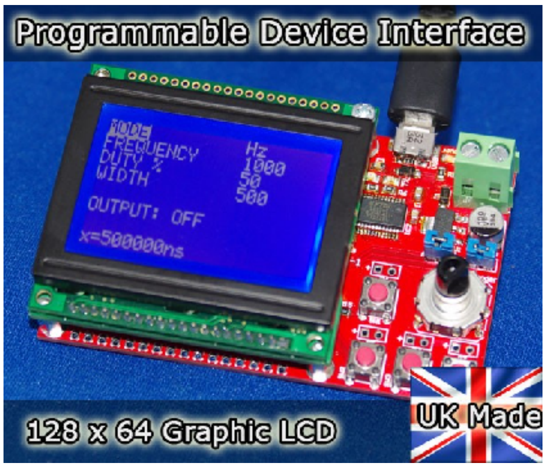
Programmable Device Interface PDI-1

A Versatile Hardware Controller with USB interface



Features and Specifications

- Arduino compatible for simple USB Programming
- 126 x 64 Graphic LCD
- 12x Digital IO ports*
- 6x 10-bit Analogue Inputs
- MicroUSB connection for computer
- 4x 3.5A H-Bridge motor controllers
- Rotary Encoder with switch
- 4x pushbutton switches
- 6V to 20V input supply voltage.
- 16MHz ATMega328 MCU

** Some I/O functions share a single connection and can be configured in the code*

The PDI-1 is a simple programmable device which is capable of directly interfacing to hardware such as motors, LEDs and sensors without the need for extensive external electronics.

With the Arduino bootloader pre-installed this device is fully compatible with Arduino software making it very easy to set up and get coding.

The built in graphic LCD is controlled over I^{2}C and combined with the rotary encoder it is great for making menu based control systems or plotting data direct to the screen.

The four 3.5 amp H-Bride motor controllers allow direct control of up to 8 loads such as high power LEDs or four DC motors in both directions. With built in over current protection and thermal overload protection, the power outputs offer great versatility.

Example Applications

- Machine Automation
- LED Lighting Systems
- Vivarium / Environmental Control
- Robotics
- Adjustable Pulse Control
- Dashboard Controller for Car Mods
- And more…

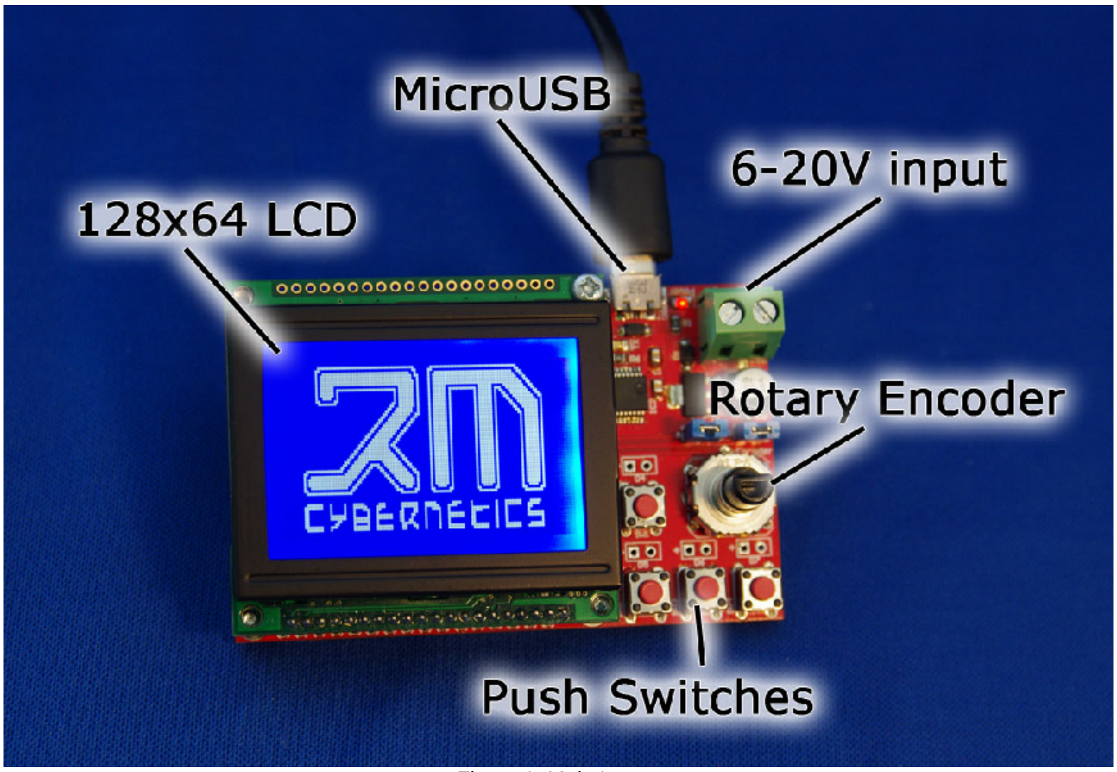


Figure 1: Main Layout

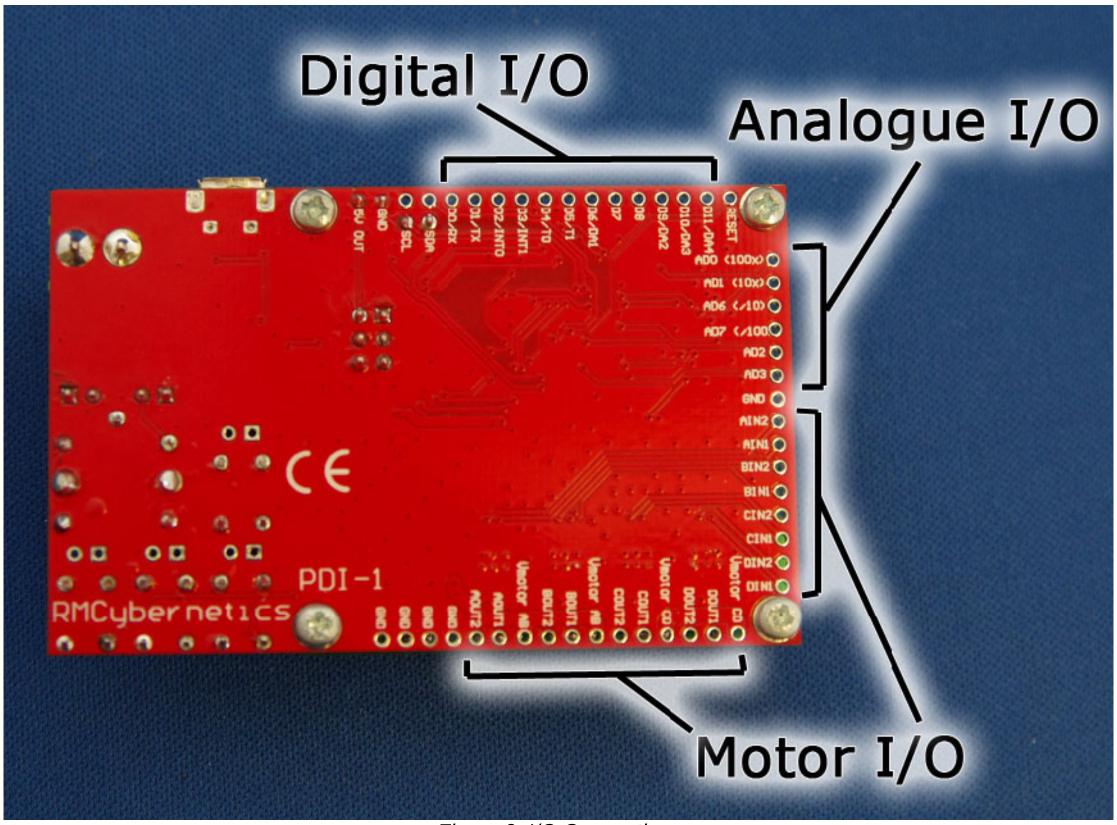


Figure 2: I/O Connections

ELECTRICAL CHARACTERISTICS

NB: Figures may vary under different loading conditions and environments.

Symbol	Parameter	Min	Max
V_{in}	Input Supply Voltage	6V	20V [1]
V_{IO}	Standard I/O Port Voltage	-	5V
I_{sup}	Supply Current	100mA	130mA
V_{motors}	Load supply voltage for H-Bridges	8V	40V
I_{motors}	H-Bridge source/sink current max	-	3.5A [2]
I_{IO}	Max I/O port current per pin	-	40mA

[1] *12V max for normal use. Higher voltages will cause excess heating in the 5V regulator and are not recommended.*
[1] *3.5A peak current. Continuous power will vary with operating conditions. Devices have built in Current and Thermal potection.*
Table 1: Electrical Characteristics

PINOUT REFERENCE

Pin Connection	Function	Arduino Equivalent Pin
SCL	I^2C Clock/ 10-bit Analogue Input 0V to 5V / Digital I/O	A5
SDA	I^2C Data/ 10-bit Analogue Input 0V to 5V / Digital I/O	A4
D0/RX	Digital I/O / Serial Rx	D0 / RX
D1/TX	Digital I/O / Serial Tx	D1 / TX
D2/INT0	Digital I/O - Interrupt Capable	D2
D3/INT1	Digital I/O - Interrupt Capable	D3
D4/T0	Digital I/O / Timer 0	D4
D5/T1	Digital I/O / Timer 1	D5
D6/DA1	Digital I/O / 8-bit PWM Output (DAC)	D6
D7	Digital I/O	D7
D8	Digital I/O	D8
D9/DA2	Digital I/O / 8-bit PWM Output (DAC)	D9
D10/DA3	Digital I/O / 8-bit PWM Output (DAC)	D10
D11/DA4	Digital I/O / 8-bit PWM Output (DAC)	D11
RESET	Resets the Microcontroller	RESET
AD0 (100x)	10-bit Analogue Input 0V to 50mV	A0
AD1 (10x)	10-bit Analogue Input 0V to 500mV	A1
AD6 (/10)	10-bit Analogue Input 0V to 50V	A6
AD7 (/100)	10-bit Analogue Input 0V to 500V *	A6
AD2	Digital I/O / 10-bit Analogue Input 0V to 5V	A2
AD3	Digital I/O / 10-bit Analogue Input 0V to 5V	A3

AIN2	Motor A signal input 2	*N/A*
AIN1	Motor A signal input 1	*N/A*
BIN2	Motor B signal input 2	*N/A*
BIN1	Motor B signal input 1	*N/A*
CIN2	Motor C signal input 2	*N/A*
CIN1	Motor C signal input 1	*N/A*
DIN2	Motor D signal input 2	*N/A*
DIN1	Motor D signal input 1	*N/A*
Vmotor CD	Load Supply Voltage for motor outputs C and D	*N/A*
VmotorAB	Load Supply Voltage for motor outputs A and B	*N/A*
DOUT1	H-Bridge power output D1	*N/A*
DOUT2	H-Bridge power output D2	*N/A*
COUT1	H-Bridge power output C1	*N/A*
COUT2	H-Bridge power output C2	*N/A*
BOUT1	H-Bridge power output B1	*N/A*
BOUT2	H-Bridge power output B2	*N/A*
AOUT1	H-Bridge power output A1	*N/A*
AOUT2	H-Bridge power output A2	*N/A*
5V OUT	5V from internal voltage regulator	5V
GND	Circuit ground. Common with all other GND connections	GND

** For transient voltages only. 100V DC max.*
Table 2: Pinout Reference

Digital I/O
The digital I/O pins when used as an output can be set as either high (5V) or low (0V). When used as an input, they will respond to the same voltage levels. It is important not to exceed 5V on these pins or the circuit could become damaged.

Analogue Outputs (DAC) / PWM Output
Some of the digital I/O pins can also be used as a pseudoanalogue output. The analogue outputs use a fixed frequency (about 500Hz) PWM signal that can be adjusted between 0 and 100% duty which can represent an equivalent voltage of 0V to 5V. This is ideal for motor speed control or LED brightness control. The I/O ports can only handle 40mA at most. If you need more current the PWM signals can be connected to the motor signal inputs so that you can control higher powered loads up to 3.5A.

Analogue Inputs

The analogue inputs have a 10-bit resolution which means the input voltage will be represented by a number between 0 and 1023. To calculate the real voltage from the ADC value, some simple math is required

*Vin = ADC Value * (ADC Max / 1024)*

Where ADC Value is the value returned by the AnalogueRead function, and ADC Max is the maximum votage for that ADC port.

For example: Using the port "AD6 (/10)" the max voltage is 50V. Therefore;

*Vin = ADC Value * (50 / 1024)*

It is important to consider the variable types you use when performing calculations as integer types cannot store fractions.

Motor/Power Outputs

The motor H-Bridge circuits are not connected to the I/O ports of the microcontroller. This allows you to use either external signals, or the onboard ones if required. It also leaves the Digital I/O ports free to use if the power outputs are not needed.

To control the power outputs with your code, you must connect a wire from one of the digital I/O ports to a motor signal input port using a short length of wire. You must also connect the H-Bridge circuits to a suitable power source that can deliver enough current, and has a voltage between 8V and 40V.

The output drivers are all low resistance (about 0.5 ohms) drivers that feature internal synchronous rectification to reduce power dissipation. The current in the output bridge is automatically limited with fixed off-time pulse width modulated (PWM) control circuitry. The IN1 and IN2 inputs allow two-wire control for the bridge. Protection circuitry includes internal thermal shutdown, and protection against shorted loads, or against output shorts to ground or supply. Undervoltage lockout prevents damage by keeping the outputs off until the driver has enough voltage to operate normally.

Motor braking can be achieved by setting both of the inputs to one driver high. If both are set low, the motor can freewheel.

Programming

Programming over USB can be achieved when using the Arduino IDE. When connected to the computer, a standard USB serial port will be installed. This port should be the one chosen in the Arduino software. The PDI-1 uses an ATMega328 microcontroller. An equivalent Arduino board would be the Arduino Nano which can be selected from the menu in the Arduino software. See arduino.cc to download the IDE and further details for programming. You can also find programming examples and projects on our website in the Programming Projects section.

It is also possible to program the microcontroller directly using the ICSP header. For this you will need a suitable programmer device. Programming via the ICSP header allows you to have it run without the Arduino bootloader which can reduce startup time.

Example Program

The code below can be copied into the Arduino IDE for testing your device. Y

```
#include <Wire.h>
#include <SPI.h>
#include <I2C_KS0108C_GLCD.h>

I2C_KS0108C_GLCD lcd;

byte picture [] PROGMEM = {
 0x1C, 0x22, 0x49, 0xA1, 0xA1, 0x49, 0x22, 0x1C,  // Item1
 0x10, 0x08, 0x04, 0x62, 0x62, 0x04, 0x08, 0x10,  // Item2
 0x4C, 0x52, 0x4C, 0x40, 0x5F, 0x44, 0x4A, 0x51,  // Item3
};


void setup ()
{
  lcd.begin ();

  // draw all available letters
  for (int i = ' '; i <= 0x7f; i++)
    lcd.letter (i);

  // black box
  lcd.clear (6, 40, 30, 63, 0xFF);

  // draw text in inverse
  lcd.gotoxy (35, 40);
  lcd.string ("I2C KS0108 GLCD", true);

  // bit blit in a picture
  lcd.gotoxy (70, 56);
  lcd.blit (picture, sizeof picture);

  // draw a framed rectangle
  lcd.frameRect (35, 49, 127, 63, 1, 1);

  // draw a diagonal line
  lcd.line (6, 40, 30, 63, 0);

}  // end of setup

void loop ()
{}
```