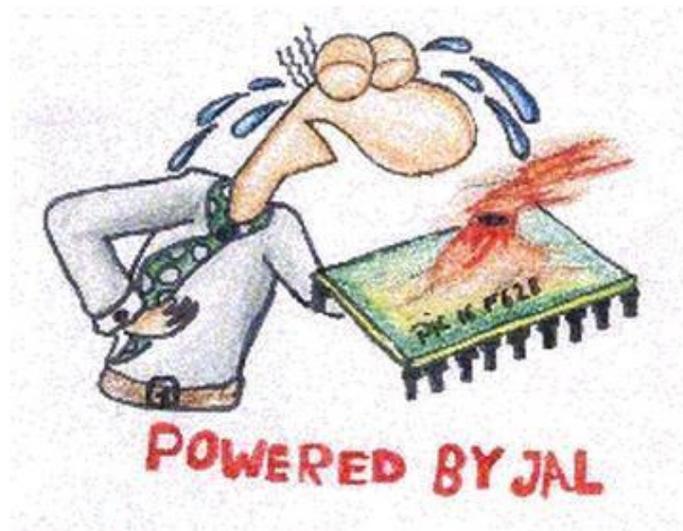


JALPIC One Development Board

Making PIC programming easy



Contents

Introduction	3
Command Set	3
Responses to commands	3
Commands.....	3
Creating a blink-a-led program.....	5
Running your blink-a-led program.....	5
Using JALPIC One with JalEdit.....	6
Using the Python script of jalpic_one.py with Python	6
Using a compiled version of the jalpic_one.py script.....	7
Using the terminal mode	8
No programming in terminal mode	8
Termite terminal emulator.....	9
Sample program to demonstrate the terminal mode.....	10
Hardware connections	11
Power Supply	11
External connections	11
Connector SV1.....	11
Connector SV2.....	11
Connector SV3.....	12
Connector SV4.....	12
Connector SV5.....	12
Appendix A: List of components.....	13
Appendix B: Component Layout.....	14
Appendix C. Porting your program to another PIC	15
The jalpic_one.jal include file	15
Porting the blink-a-led program to a PIC12F615	15

Introduction

The JALPIC One development board can be used to quickly develop a standalone applications in JAL running on a PIC. On this board you will find two PICs, a controller PIC16F1455 which handles the interface with the USB and does the programming of the second PIC16F18857- the application pic- on which the application runs.

Since the application PIC is programmed by the JALPIC One development board there is no need for a separate programmer nor a bootloader.

The JALPIC One development board acts a virtual serial device on the USB Port of the computer. Once connected to the USB port use the following serial settings:

- Baud rate: Any baud rate up to 115200 baud
- Number of bits: 8
- Parity: None
- Number of stop bits: 1
- RTS/CTS: Enabled

Once an application is developed in JAL and compiled successfully it can be downloaded and tested using JALPIC One development board.

Command Set

Responses to commands

The JALPIC One development board uses a simple command set using ASCII characters. After every command that is given, one of the following responses is returned as a string ending with a carriage return plus linefeed:

- '0': The command was executed successfully.
- '1': The command was not executed successfully.
- '2': A timeout occurred. This happens when the program mode or verify mode is active but no hex data was provided within a certain time frame.
- '3': Unknown command.

Depending on the command given, the response time varies. For example performing a blank check will take some time before '1' is returned after a successful blank check.

Commands

Each of the following commands should be preceded by a '!' and terminated by a carriage return and/or a linefeed. The commands are case insensitive. Commands are given in alphabetical order:

'A'. Check if the JALPIC One development board is still alive by checking the device ID of the application PIC. Returns '0' when successful or '1' if not.

'BE'. Perform a blank check of the EEPROM. Returns '0' when the EEPROM is empty or '1' if not.

'BF'. Perform a blank check of the flash. Returns '0' when the flash memory is empty or '1' if not.

'D': This command toggles the debug mode but should not be enabled for normal operation. During debug mode all kind of information is written to the serial port of the controller PIC.

'EE'. Erase the EEPROM of the application PIC. Returns '0' when executed successfully or '1' when not.

'EF'. Erases the flash of the application PIC. Returns '0' when executed successfully or '1' when not.

'P'. Program the application PIC including verification. This command puts the JALPIC One in the programming mode. Returns '0' when executed successfully or '1' when not.

After this command the contents of the hex file has to be provided. Once the complete hex file is successfully programmed '0' is returned. Next to programming a verification is done during programming. If verification during programming fails, '1' is returned. When no hex file data is provided for too long '2' will be returned to indicate a timeout. In all cases the programming mode will end.

'R'. Reset the JALPIC One development board. The application PIC is also reset. The controller PIC is only reset to its initial state so no hardware reset. The application PIC is however reset by activating the MCLR\ pin of that PIC. Returns '0' when executed.

'T'. Terminal mode. When this command is given, all characters which are entered by on the virtual serial device are forwarded to the application PIC using the serial interface between the two PICs. Any character sent by the application PIC via the serial interface is forwarded to the client via the virtual serial device. In this way information from the application PIC can be shown in a terminal emulator program on the computer without the need of a separate serial interface. The command returns a '0'. The serial interface settings are fixed to 115200 baud, 8 bits, one stop bit and no parity. Terminal mode disables all other commands and can be terminated by the client by entering the escape character (0x1B) or ctrl-C (0x03) after which '0' is returned.

'V'. Verify device. This command puts the JALPIC One in the verification only mode as is done with the command 'P'. Returns '0' when executed successfully or '1' when not.

After this command the contents of the hex file must be provided. Once the complete hex file is successfully verified '0' is returned. If verification fails, '1' is returned. When no hex file is provided for too long '2' will be returned to indicate a timeout. In this case the verification mode will end.

Creating a blink-a-led program

Since the configuration of the JALPIC One board is fixed, programming becomes simpler since there is a standard include file that has to be included in the program. A simple blink-a-led program then looks like this:

```
include jalpic_one          -- include the board definition file

enable_digital_io()       -- make all pins digital I/O

alias led is pin_a0        -- alias for pin with LED
pin_a0_direction = OUTPUT

forever loop
  led = FALSE              -- LED is on when port is made low.
  _usec_delay(100_000)
  led = TRUE               -- LED is off when port is made high.
  _usec_delay(400_000)
end loop
```

There is no need to set the pragmas and the target clock since that has all been done in the `jalpic_one` include file. Note that the JALPIC One development board has a LED connected to port A0 for testing purposes. This LED can be enabled or disabled by a jumper that can be installed on the board or removed from the board.

Running your blink-a-led program

After having compiled the blink-a-led program successfully, the created hex file is needed to program the device. A normal programming sequence would be as follows:

- Configure the USB port with the right serial setting. Note that RTS/CTS must be enabled.
- Give the command '!EF' on the virtual serial device to erase the flash and check if response is '0'. Note that if EEPROM data is used in the program then the command '!EE' must also be executed as to erase the EEPROM.
- Give the command '!P' on the virtual serial device to start the programming cycle and check if response is '0'
- Copy of the hex file to the virtual serial device and check if response is '0'.

If no special program is available to control the JALPIC One development board and to check the responses, a batch file can be used to program the application PIC. For Python users a Python script called '`jalpic_one.py`' is available which supports Windows, Linux and MAC and is used as follows for Windows:

```
Python jalpic_one.py myprogram.hex com2
```

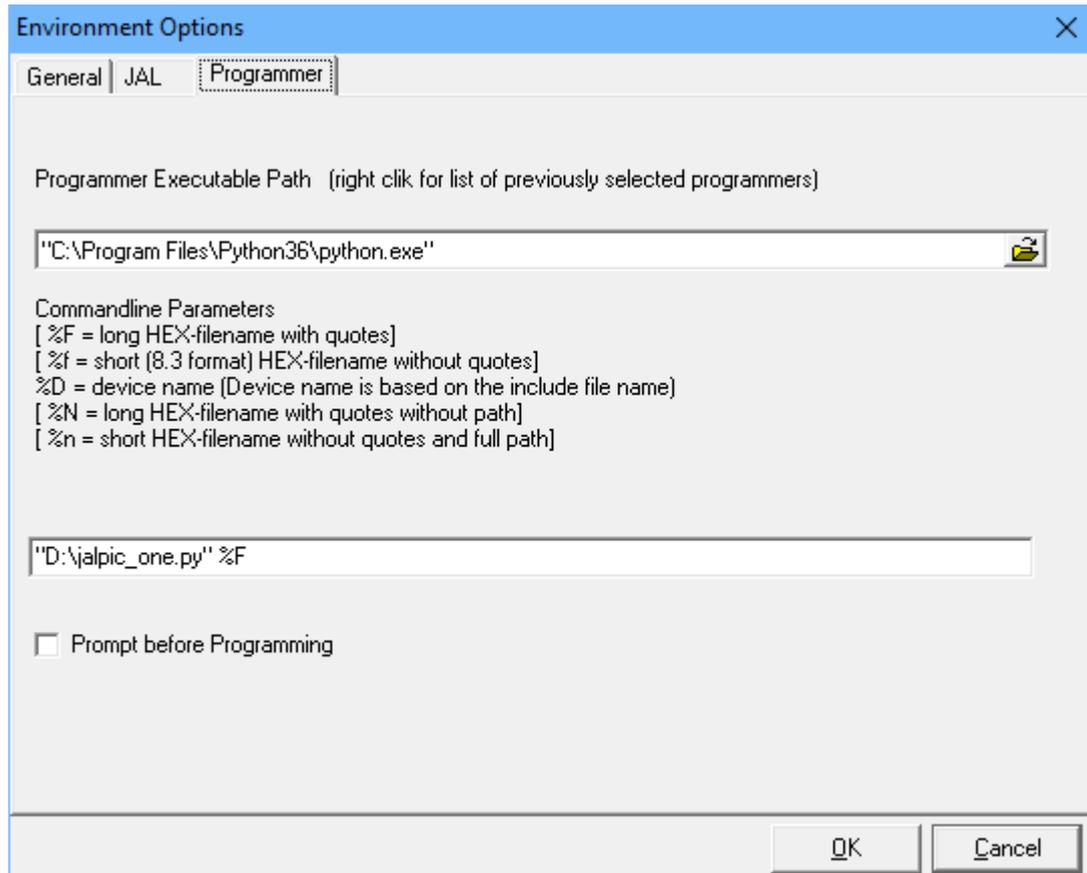
The script will then open serial port com2, erase the flash and EEPROM, start the programming mode and copy the file '`myprogram.hex`' to the serial port com2, which then programs the application PIC. If no serial port is given, the script uses com3. This default can be changed in the script for the various operating systems the script supports. If an error occurs in any of the operations, the script will stop, shows what went wrong and will ask the user to press Enter to continue. If all goes well, that information is also show but the user does not have to press Enter. Note that next to Python you must have installed PySerial too.

Using JALPIC One with JalEdit

For Windows users the JALPIC One development board can be used in combination with JalEdit to make life easier. This makes it possible to compile your program and directly download it to the JALPIC One development board with a single click on the compile-and-program button in JalEdit. There are two ways to do that.

Using the Python script of jalpic_one.py with Python

The jalpic_one.py script can be executed by running python and setting the programmer environment options in JalEdit as follows:



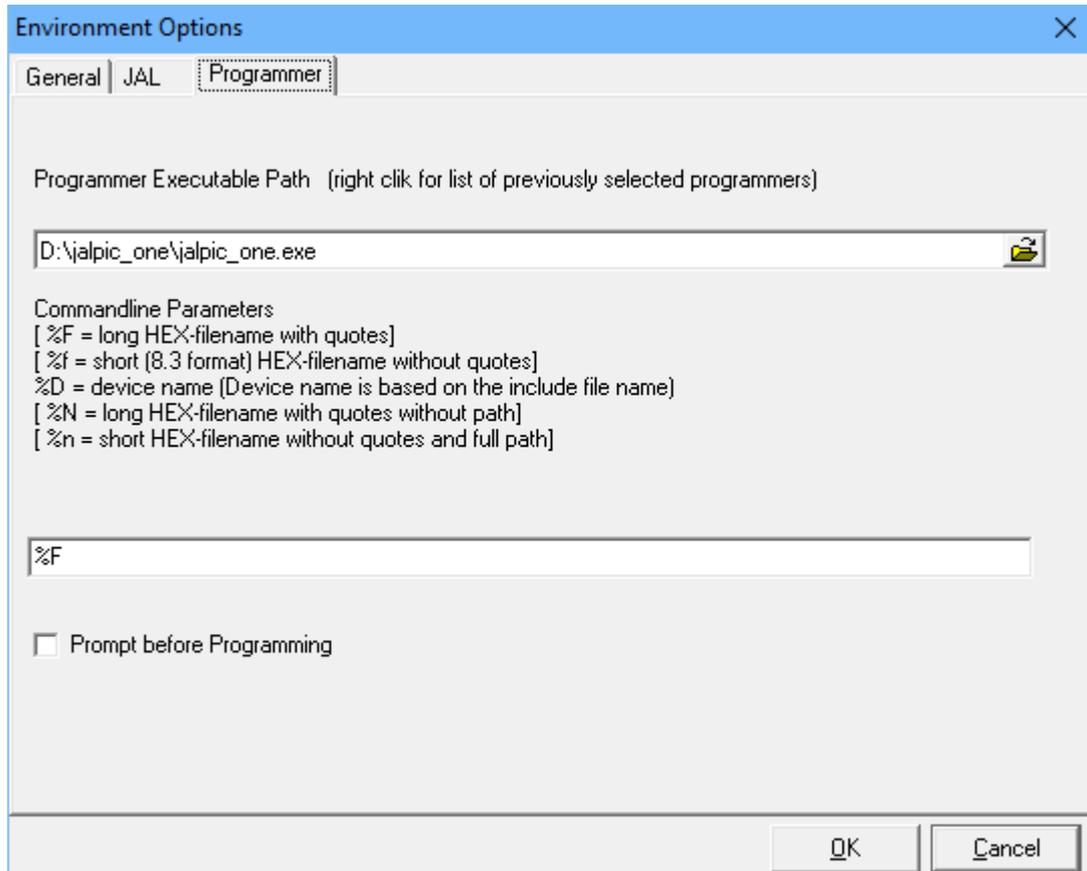
The default COM port of jalpic_one.py script is com3. This can be changed by adding the name of the com port after the %F. When com1 needs to be selected this becomes:

```
"D:\jalpic_one.py" %F com1
```

Using a compiled version of the jalpic_one.py script

A compiled version of jalpic_one.py can be made by using the PyInstaller which will then create an executable. This has to be done once using the command: PyInstaller jalpic_one.py. This solution works faster than when running it from Python.

After running PyInstaller a directory called 'dist' is created with a subdirectory 'jalpic_one'. You can move this directory with all its libraries to any other location on your computer. In JalEdit the programmer environment settings are then as follows:



The default COM port of jalpic_one.exe program is com3. This can be changed by adding the name of the com port after the %F. When com1 needs to be selected this becomes:

```
%F com1
```

Using the terminal mode

As mentioned earlier the JALPIC One development board supports a terminal mode. This means that when the application PIC sends data on its USART connection, this information can be viewed on the computer using the COM port to which the JALPIC One development board is connected. Because of this feature there is no need for a separate RS232 connection on the PC.

In order to be able to use this feature the USART settings for the program running on the application PIC have to be as follows:

- Baud rate: 115200 baud
- Number of bits: 8
- Parity: None
- Number of stop bits: 1

When the JALPIC One development board is placed in terminal mode it will send all data that is received from the application PIC to the virtual serial device. Also all data send to the virtual serial device will be forwarded to the application PIC.

For sending and monitoring the data, a terminal emulation program can be used that connects to the COM port to which the JALPIC One development board is connected. The free program Termite can be used for that purpose. When starting Termite, select the correct COM port and communication settings as mentioned in the Introduction. Once connected type '!T' and press return. This will place the JALPIC One development board in terminal mode. All commands now typed in Termite will be sent do the application PIC and any data from the application PIC is shown in Termite.

As screenshot of a possible use of Termite in combination with the JALPIC One development board is given on the next page. As you can see at the top is that it uses COM port 3 with a baud rate of 115200, 8 bits, no parity and one stop bit.

This screenshot shows the output of a program that runs on the application PIC which sends a hello message every 5 seconds but it will also send all text back that is typed in the lower bar of the Termite program.

No programming in terminal mode

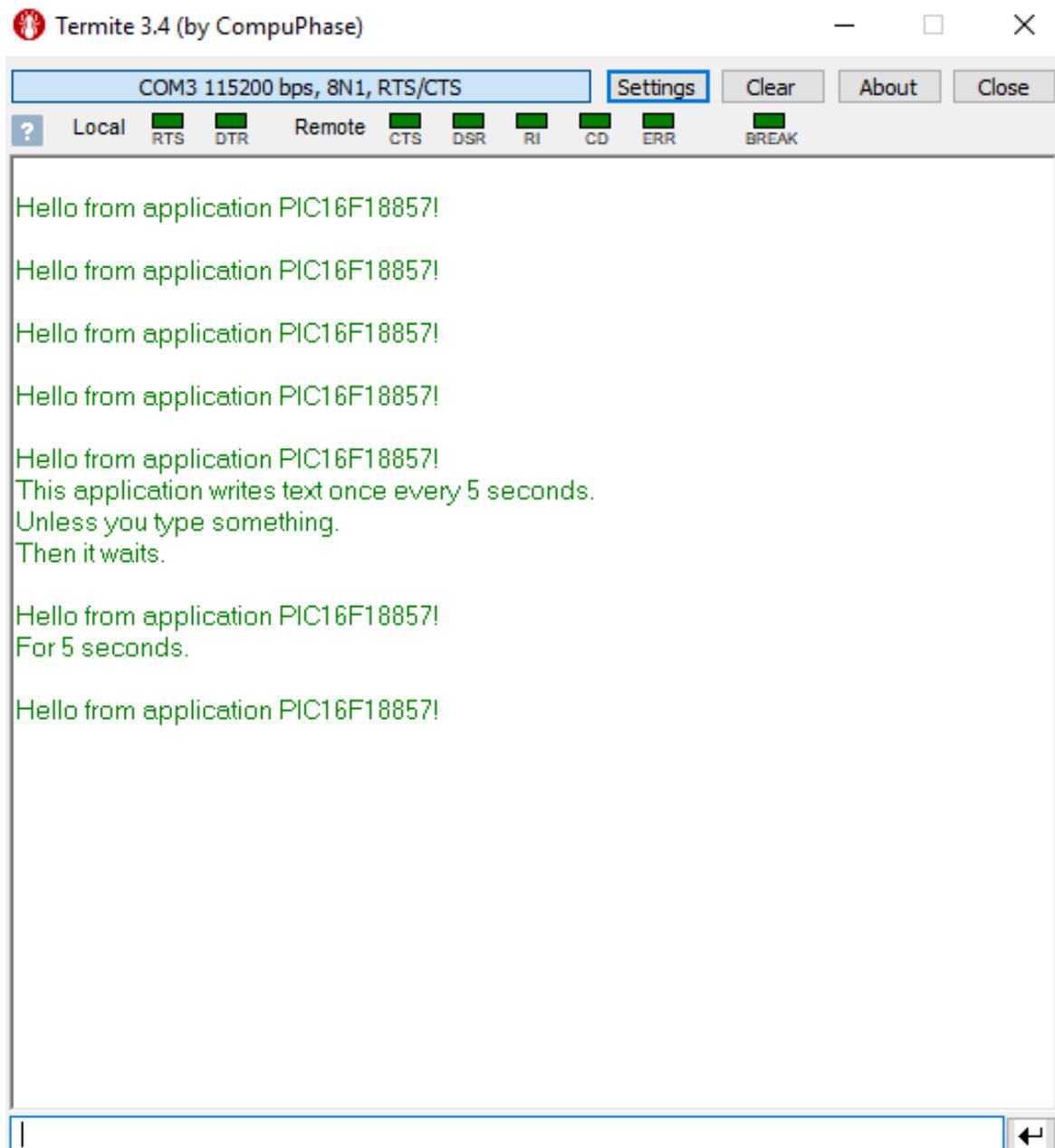
There is one important note to be mentioned. When the JALPIC One development board is in terminal mode, it will not respond to any command since it just sends those commands as text to the application PIC. This means that in terminal mode the JALPIC One development board cannot be programmed unless the terminal mode is stopped.

Stopping the terminal mode is done by entering escape (0x1B) or control-c (0x03). However, in normal mode of Termite pressing these keys have no effect. In order to stop the terminal mode in Termite do the following:

- Go to Settings
- Select 'Hex View' in the section Plug-ins
- It will come up with a filter setting, click OK. Note that all information on the screen is now also shown in hexadecimal format.
- Type 0x03 in the lower bar, followed by return. This will end the terminal mode.

Termite terminal emulator

Below as screenshot of Termite is given showing the terminal emulation mode in action. The application program used is listed in on the next page.



Sample program to demonstrate the terminal mode

The above example is using the following sample program that was programmed into the application PIC before starting the terminal mode.

```
include jalpic_one          -- include the board definition file

-- Definitions needed for serial_hw_int_cts.
alias pin_RX_direction is pin_C7_direction -- Pin 18 for 28 pin DIP
alias pin_TX_direction is pin_C6_direction -- Pin 17 for 28 pin DIP.
const serial_hw_baudrate = 115200 -- Terminal connection.

include serial_hw_int_cts -- We use this version because of its buffer.
include print
include pps

-- Global variables.
var word timer
var byte character
var bit led_value

enable_digital_io()          -- make all pins digital I/O

-- On the JALPIC One board a LED is present.
alias led is pin_a0
pin_a0_direction = OUTPUT

-- Set the pinning for the UART. This is default but still needed.
pps_control_lock(FALSE)
RC6PPS = PPS_TX          -- TX1 re-assigned to C6 (default)
RXPPS = PPS_RC7         -- C7 re-assigned to RX1 (default)
pps_control_lock(TRUE)

serial_hw_init()
led_value = TRUE
timer = 0

forever loop
  _usec_delay(1_000)

  if serial_hw_read(character) then
    serial_hw_data = character -- Echo all received characters.
    timer = 0
  end if

  -- Message sent every 5 seconds.
  timer = timer + 1
  if (timer == 5_000) then
    led_value = !led_value
    timer = 0
    print_string(serial_hw_data,
      "\r\nHello from application PIC16F18857!\r\n")
  end if

  led = led_value
end loop
```

Hardware connections

Power Supply

The JALPIC One development board can be supplied by the USB port via the USB connector X1. When used standalone an external power supply can be connected to the Power Jack J1. The supply voltage on this Jack may vary between DC 7 Volt to DC 20 Volt, but the lower the better since it will prevent too much heat dissipation in the voltage regulators.

Note: Make sure that JP1 is has selected the correct power input. This can be seen by LED3 being lit when the board is powered on.

External connections

There are four connectors on the board to interface the application PIC with the outside world. In the tables below these connections are shown including to which pin of the application PIC they are connected.

Note: Although connectors are meant to support a special type of function, the pin functions can always be changed by configuring the application PIC with the program that runs on it.

Connector SV1

Connector SV1 is meant as power connector.

SV1 Pin	Name	Type	PIC Pin	PIC Pin Name	Remarks
1	Reserved	-	-	-	Not used.
2	VREF-	Input	4	RA2	External ADC/DAC negative reference voltage
3	MCLR\	Input	1	MCLR\	Reset pin of the application PIC, active low
4	+3.3V	Output	-	-	+3.3 Volt power supply
5	+5V	Output	-	-	+5 Volt power supply
6	Ground	-	-	-	Ground power connection
7	Ground	-	-	-	Ground power connection
8	VIN	Output	-	-	Voltage as supplied on the DC Jack J1

Note: The application PIC can also be reset (MCLR\) using the reset button (S1) on the JALPIC One development board. This switch will connect the MCLR\ line directly to ground. This line is also used by the controller PIC during programming. For that reason the line must not be pulled high by an external device.

This reset does not affect the operation of the controller PIC, only the application PIC.

Connector SV2

Connector SV2 is meant as analog connector but can also be used for digital signals.

SV2 Pin	Name	Type	PIC Pin	PIC Pin Name	Remarks
1	A0	I/O	2	RA0	Analog input but also Digital I/O.
2	A3	I/O	5	RA3	Analog input but also Digital I/O.
3	A4	I/O	6	RA4	Analog input but also Digital I/O.
4	C0	I/O	11	RC0	Analog input but also Digital I/O.
5	B3	I/O	24	RB3	Analog input but also Digital I/O.
6	B4	I/O	25	RB4	Analog input but also Digital I/O.

Note: The Test LED1 can be connected to RA0 by installing Jumper JP2. The LED will turn on when RA0 is made low.

Connector SV3

Connector SV3 is meant as interface for serial interfaces like SPI or I2C.

SV3 Pin	Name	Type	PIC Pin	PIC Pin Name	Remarks
1	PGC2	Input	27	RB6	ICSP clock for programming the application PIC
2	PGD2	I/O	28	RB7	ICSP data for programming the application PIC
3	SS1	Input	7	RA5	MSSP1 SPI slave select
4	SDO	Output	16	RC5	MSSP1 SPI serial data
5	SDI	Input	15	RC4	MSSP1 SPI serial data
6	SCK	I/O	14	RC3	MSSP1 SPI clock input/output
7	Ground	-	-	-	Ground connection
8	VREF+	Input	3	RA1	External ADC/DAC negative reference voltage.
9	SDA	I/O	23	RB2	MSSP1 I2C serial data input/output
10	SCL	I/O	22	RB1	MSSP1 I2C clock input/output

Note: PGC2 and PGD2 are shared with RB6 and RB7 and are used by the controller PIC to program the application PIC. For that reason these pins should not be heavily loaded by an external device since it may negatively impact the programming of the application PIC.

Connector SV4

Connector SV4 is meant as interface for digital I/O, USART and PWM connections.

SV4 Pin	Name	Type	PIC Pin	PIC Pin Name	Remarks
1	RX	Input	18	RC7	USART Receive
2	TX	Output	17	RC6	USART Transmit
3	INT	Input	21	RB0	External interrupt
4	PWM1	Output	13	RC2	Pulse Width Modulator 1
5	B5	I/O	26	RB5	Digital I/O
6	PWM2	I/O	12	RC1	Pulse Width Modulator 2
7	B6	I/O	27	RB6	Digital I/O, shared with PGC2
8	B7	I.O	28	RB7	Digital I/O, shared with PGD2

Connector SV5

Connector SV5 is meant to program the controller PIC of the board via the ICSP interface. The connections are based on the use of a PicKit3 programmer

SV5 Pin	Name	Type	Remarks
1	MCLR\VPP	Input	Reset and Programming voltage
2	+5V	Output	+5 Volt power supply
3	Ground	-	Ground connection
4	PGD1	I/O	ICSP data for programming the controller PIC
5	PGC1	Input	ICSP clock for programming the controller PIC
6	-	-	Not connected

Appendix A: List of components

IC

- 1 * LM2940CT-5.0: IC1
- 1 * LM3940IT-3.3: IC2
- 1 * PIC16F18557P: IC3
- 1 * PIC16F1455P: IC4

Crystal

- 1 * 20 MHz: Q1
- 1 * 12 MHz: Q2

Diode

- 1 * 1N4004: D1
- 1 * 1N4148: D2

LED

- 1 * Yellow LED: LED1
- 1 * Amber LED: LED2
- 1 * Red LED: LED3

Connector

- 1 * Power Jack: J1
- 1 * USB Connector: X2
- 2 * 6-pin header: SV2, SV5
- 2 * 8-pin header: SV1, SV4
- 1 * 10-pin header: SV3
- 1 * 3-pin jumper: JP1
- 1 * 2-pin jumper: JP2

Capacitor

- 4 * 22 pF: C1, C3, C11, C13
- 5 * 100 nF: C2, C6, C7, C8, C9
- 1 * 470 nF/Ceramic: C10

Electrolytic Capacitor

- 3 * 10 uF/25V: C4, C5, C12

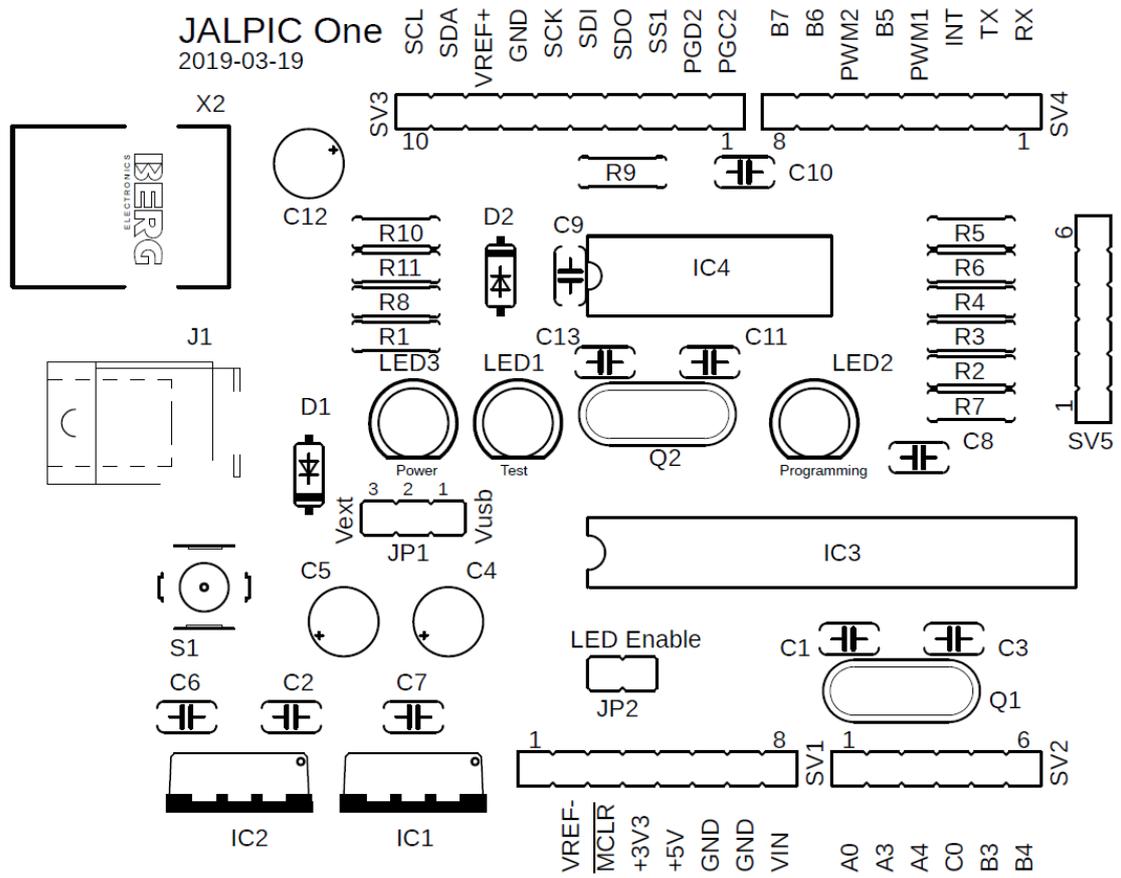
Resistor

- 2 * 22 Ohm: R10, R11
- 2 * 330 Ohm: R1, R8
- 6 * 1 kOhm: R2, R3, R4, R5, R6, R7
- 1 * 33 kOhm: R9

Switch

- 1 * Omron Pushbutton: S1

Appendix B: Component Layout



Appendix C. Porting your program to another PIC

After you have developed your program on the JALPIC One development board, you may want to port it to another PIC, e.g. because your application can be handled with a much smaller PIC and you do not need all other hardware which is on the JALPIC One development board.

Note that you need a programmer like a PicKit3 as to be able to program the other PIC.

The `jalpic_one.jal` include file

In your initial program you have included the `jalpic_one.jal` include file. This file takes care of the following:

- Include the device file for the PIC16F18857.
- Set the pragmas in line with the hardware of the JALPIC One development board for the PIC16F18857.

This `jalpic_one.jal` file looks like this:

```
include 16f18857                                -- target PICmicro
pragma target clock 20_000_000                  -- oscillator frequency
pragma target OSC HS                            -- crystal or resonator
pragma target RSTOSC EXT1X                      -- power-up clock select: OSC
pragma target CLKOUTEN DISABLED                -- no clock output
pragma target WDT DISABLED                     -- no watchdog
pragma target BROWNOUT DISABLED               -- no brownout reset
pragma target FCMEN DISABLED                  -- no clock monitoring
pragma target CSWEN ENABLED                   -- allow writing OSCCON1 NOSC and NDIV
pragma target LVP ENABLED                     -- use low voltage programming
pragma target MCLR EXTERNAL                   -- external reset
```

The definition and the descriptions of the used pragmas can be found in the datasheet of the PIC and in the included device file of the PIC16F18857.

Porting the blink-a-led program to a PIC12F615

If we want to port our blink-a-led program to a smaller PIC12F615 we need include the right device file and we must define our own pragmas. In this example we use the internal oscillator of the PIC12F615, with clock frequency of 4 MHz, using the internal reset. The `jalpic_one.jal` include file must be left out and needs to be replaced by something like this:

```
include 12f615                                  -- target PICmicro
pragma target clock 4_000_000                   -- oscillator frequency 4 MHz
pragma target OSC INTOSC_NOCLKOUT              -- Internal Clock
pragma target PWRTE Enabled                    -- Power up timer
pragma target MCLR Internal                    -- Reset internal
pragma target WDT Disabled                     -- No watchdog
pragma target BROWNOUT Disabled               -- No brown-out
pragma target IOSCF5 F4MHZ                     -- internal oscillator 4 MHz
```

The rest of the blink-a-led program remains the same. After this modification the blink-a-led program can be compiled and the generated hex file can be used to program the 12F615 using a separate programmer. Note that each PIC is different and so are the pragmas.