

Dwarf Boards

DN002 : Jal library

(c) Van Ooijen Technische Informatica
version 1.0

PICmicro, In-Circuit Serial Programming and ICSP are registered trademarks of Microchip Technology Inc.

Introduction

Jal is a simple but effective high level language for programming PICmicro controllers. The Jal compiler is available under GPL license from <http://www.sourceforge.net/projects/jal>.

A special Jal library is available for the Dwarf Boards product family. This document describes the basic part of this library. Board specific aspects are described in the document that describes the board.

Controller board identification

In order to use the features of the Dwarf Board library you must include at least the controller board specific library file. Details about this are mentioned in the document for the specific controller board.

example:

```
-- include the board specific file for a DB016 board with an 18F252 at 40 MHz
include DB016_18F252_40
```

Digital input and output

The Dwarf Board Jal library provides access to I/O pins Dwarf Bus bus names. Using bus name based access hides the fact that busses and PICmicro ports do not always have a one-to-one correspondence, and eases transition between various controller boards.

Access is provided for a bus as a whole (8 bits), the highest or lowest nibble (4 bits), and a single bit. For the 14-bits PICs (12F and 16F series) the I/O is buffered to avoid the read-modify-write problem inherent in the PIC I/O hardware.

Most PICmicro I/O pins can be set as input or output. The power-on default for all pins is input. The table below shows the declarations that are provided to set the direction of the pins of bus_a. In all cases a zero (or false for the bits) sets the I/O to output, a one (or true for bits) sets an I/O to input. The constants input, output, all_input and all_output can be used to improve readability.

| declaration | use |
|---------------------|---------------------------------------------------------|
| bus_a_direction | to read or write the value of a bus (8 bits) |
| bus_a_low_direction | to read or write the value of the lower 4 bits of a bus |

| declaration | use |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------|
| bus_a_high_direction | to read or write the value of the higher 4 bits of a bus note: the result is right-aligned (in the range 0x00 .. 0x0F) |
| bus_a_pin_0_direction | to read or write the value of a single pin |
| bus_a_pin_1_direction | |
| bus_a_pin_2_direction | |
| bus_a_pin_3_direction | |
| bus_a_pin_4_direction | |
| bus_a_pin_5_direction | |
| bus_a_pin_6_direction | |
| bus_a_pin_7_direction | |
| bus_a_pin_8_direction | |
| bus_a_pin_0_direction | |
| input | same as true |
| output | same as false |
| all_input | same as 0b_1111_1111 |
| all_output | same as 0b_0000_0000 |

The tables below show the declarations that are provided for access to the values of the pins of bus_a, similar declarations are provided for other busses.

| declaration | use |
|--------------------|---------------------------------------------------------------------------------------------------------------------------|
| bus_a | to read or write the value of a bus (8 bits) |
| bus_a_low | to read or write the value of the lower 4 bits of a bus |
| bus_a_high | to read or write the value of the higher 4 bits of a bus note: the result is right-aligned (in the range 0x00 .. 0x0F) |
| bus_a_pin_0 | to read or write the value of a single pin |
| bus_a_pin_1 | |
| bus_a_pin_2 | |
| bus_a_pin_3 | |
| bus_a_pin_4 | |
| bus_a_pin_5 | |
| bus_a_pin_6 | |
| bus_a_pin_7 | |
| bus_a_pin_8 | |
| bus_a_pin_0 | |

Some I/O pins can be configured either as analog or as digital pins. In most cases the power-on default for pins that have this capability is analog. The procedure bus_a_digital is provided to set the mode of the bus_a pins to digital. The bus_b and bus_c pins do not have analog capabilities, but dummy bus_b_digital and bus_c_digital procedures are provided anyway.

| declaration | use |
|--------------------|---------------------------------------|
| bus_a_digital | to set the bus_a pins to digital mode |
| bus_b_digital | to set the bus_b pins to digital mode |

| declaration | use |
|---------------|---------------------------------------|
| bus_c_digital | to set the bus_c pins to digital mode |

example:

```

bus_a_digital
bus_a_direction = all_output
bus_a = 0b_1010_1010

bus_b_digital
bus_b_pin_0_direction = output
bus_b_pin_0 = false

```

Analog input

The use of the Analog Digital converter requires three steps:

1. the hardware must be configured, which determines the pins that will be used as analog inputs, reference inputs, or not used for analog purposes (available for digital I/O)
2. the pins that are used for analog purposes must be configured as input (which is the default, so just do not configure them as output)
3. an analog pin is read

The procedure `adc_configure` can be used to configure the A/D converter hardware. The argument must be one of the `adc_x_ana_y_ref` constants mentioned in the table below.

| declaration | use |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>adc_configure(conf)</code> | configure the pins that are (and are not) used for analog purposes |
| <code>adc_8_ana_0_ref</code> | values that can be used as arguments to <code>adc_configure</code> refer to the PIC datasheet (A/D converter section) for the effect of each value |
| <code>adc_7_ana_1_ref</code> | |
| <code>adc_5_ana_0_ref</code> | |
| <code>adc_4_ana_1_ref</code> | |
| <code>adc_3_ana_0_ref</code> | |
| <code>adc_2_ana_1_ref</code> | |
| <code>adc_0_ana_0_ref</code> | |
| <code>adc_6_ana_2_ref</code> | |
| <code>adc_6_ana_0_ref</code> | |
| <code>adc_5_ana_1_ref</code> | |
| <code>adc_4_ana_2_ref</code> | |
| <code>adc_3_ana_2_ref</code> | |
| <code>adc_2_ana_2_ref</code> | |
| <code>adc_1_ana_0_ref</code> | |
| <code>adc_1_ana_2_ref</code> | |

The procedure `adc_read_10` can be used to read the full 10 bit result of an A/D conversion. The channel argument determines which pin will be read. This must be a pin that has been configured as analog input (not as reference) and it must be configured as input. The result is returned in `result_1` (higher 2 bits) and `result_0` (lower 8 bits).

| declaration | use |
|--------------------------------------------|------------------------------------------------------------|
| adc_read_10(channel, result_1, result_0) | perform an A/D conversion and get the result |
| adc_bus_a_pin_0 | values that can be used as channel argument to adc_read_10 |
| adc_bus_a_pin_1 | |
| adc_bus_a_pin_2 | |
| adc_bus_a_pin_3 | |
| adc_bus_a_pin_5 | |

HD44780 compatible LCD

The LCD libraries support using an HD44780-compatible LCD in 4 bit mode. Using these libraries requires the following steps:

1. declare the constant `lcd_lines`
2. include one of the files `lcd_on_a`, `lcd_on_b`, `lcd_on_c`
3. use one of the LCD declarations to write to the LCD

The constant `LCD_lines` is used only to distinguish between a 1x16 display (which needs some special handling because the one line is actually implemented as two lines x 8 characters).

The files `lcd_on_a` etc. contain the definitions of the IO pins used when a DB015 LCD interface is connected to port a etc. These files in turn include the file `lcd.jal`, which contains the actual IO code. When you want to connect an LCD in a different way you can copy one of the `lcd_on_x` files and modify it to suit your needs.

| declaration | use |
|------------------------------------|--------------------------------------------------------------|
| const <code>LCD_lines</code> = ... | must be put before the 'include <code>lcd_on_x</code> ' line |
| <code>LCD_Init</code> | initialize the LCD |
| <code>LCD_Clear</code> | clears the LCD and puts the cursor at line 1 position 0 |
| <code>LCD_Write</code> | writes a character to the LCD |
| <code>LCD_Instruction</code> | writes an instruction to the LCD |

Check DN008 for an example of the use of the LCD interface.

Standard Jal library

The standard Jal library provides declarations that can be useful in developing your application. The table below summarises these libraries. Refer to the Jal documentation for details.

| library file name | provides |
|-------------------------|------------------|
| <code>jdelay.jal</code> | delay procedures |

Change notes

the latest version of this document can be downloaded from <http://www.voti.nl/dwarf>

| version | date | notes |
|----------------|-------------|---------------|
| 1.0 | 2003-11-04 | first version |