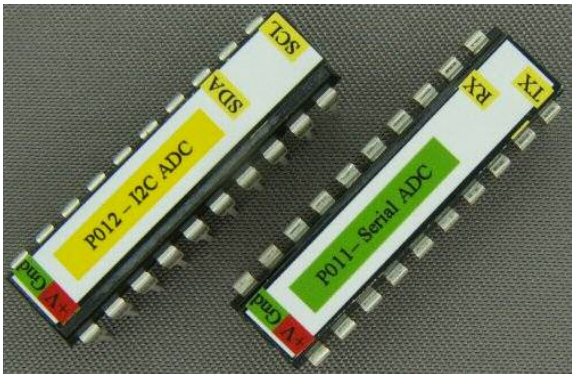
I2C or Serial ADC P011/2



P011/2

I2C or Serial ADC

Product specification Oct. 2013 V0.a

Contents

Rev	Change
Oct 2013	Preliminary
Jul 2014	Command 'p' added

1. Introduction

The P011 is a serial ADC and the P012 is an I2C ADC. This is supplied and an IC but there is also a PCB.

The ADC has 8 channels 10 bits wide (0 to 1023). In addition to this there are 3 digital outputs. All of these can be conveniently be controlled with 2 wires(serial or I2C)

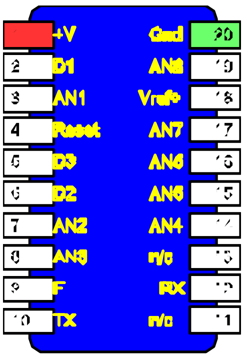
More data and examples with free software can be found at www.pichips.co.uk

2. Features

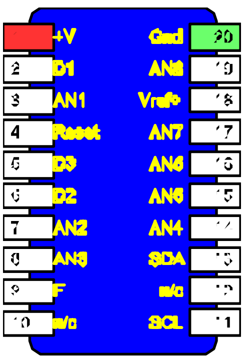
- Wide voltage range 2.5V to 5.5V
- 8 ADC channels, 10bit
- 3 digital outputs
- Accurate voltage ref built in
- Simple serial or I2C protocol
- User configurable EEPROM

3. Electrical interface

The IC is a 20 pin DIL for ease of use and has two versions:



PO11 Serial IC



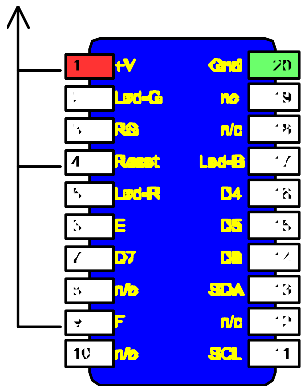
PO12 I2C IC

There are two versions of the IC, the only electrical difference is that one has TX,RX pins and the other has SCL, SDA pins.

3.1. IC Pin out

Pin	Description
1	+V, this can range from 2.5V to 5.5V
2,5 & 6	Digital outputs that provide a signal from 0 to +V
3,7,8, 14,15, 16,17, 19	Analogue inputs, acceptable maximum voltage is +V
4	Reset, This **must** be tied to +V, taking this low will reset the IC. It can be tied to a resistor if a reset needs to be implemented. The IC will reset on power up.
9	F this **must** be tied to +V for normal operation
10	TX (output) line for serial devices no connection for I2C devices
11	SCL (clock) for I2C devices, no connection for serial devices
12	RX (input) for serial devices no connection for I2C devices
13	SDA (data) line for I2C devices no connection for serial devices
20	Ground

IC pin out Table for P017/8



For correct operation Pins 4 and 9 must be tied to +V.

3.2. The F pin

The IC is controlled by values in the EEPROM and it may be possible that the user could change critical values that would stop the IC from communication. If this happens:

1) Remove power
2) Tie pin F to ground
3) Apply power

4) Remove power

This procedure will restore the first few bytes of EEPROM that will restore normal communications using the default address.

3.3. I2C

The I2C interface is the standard arrangement, somewhere along the bus a pull up resistor is required for the SCL and SDA lines.

The default device address is 0x68(8 bit) or 0x34 for Rpi and Arduino

3.4. Serial

The serial interface is a standard 1 start bit 8 data bits and 1 stop bit and is initially set to 9600 Baud. This is user changeable from 2400 to 115200 in 8 steps.

The interface can be connected to a UART or USB to serial device and expected the voltage to be TTL levels (0 and 5V or 0 and 3.3V). By default and when connecting directly to one of the above devices the output (TX) is correct for a single device.

There is an opportunity to have more than one device share the TX line and this is achieved with an open collector circuit. The circuit diagram of the PO17 PCB shows how this can be achieved. When using this circuit the output (TX) requires inverting and this is done via a setting in the EEPROM.

By default the serial interface is set at 9600 Baud and the serial device address is '**h**'.

4. ADC

The ADC is running continuously and saving values to its internal RAM. The speed of the samples can be adjusted to suit the physical conditions.

The MAXIMUM input voltage to any ADC pin must not exceed +V. The range of the ADC is 10 bits so this will translate to a value of 0 to 1023.

There is an accurate reference voltage that can be selected via the interface, see the command table for the values.

Poor results will be obtained if the input impedance exceeds 10k

5. Digital Output

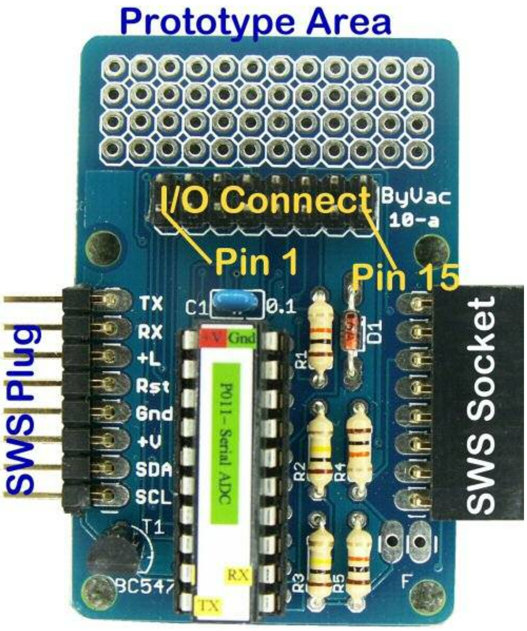
These will provide an output fro 0 to +V and can be accessed via the serial or I2C interface.

6. Using the PCB

The device can be used with or without the PCB, it is however a very convenient way of using the device.

Full instructions for assembling the PCB are given at www.pichips.co.uk

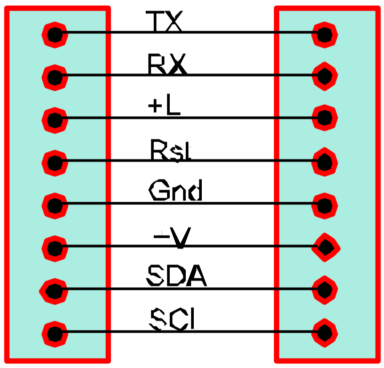
The PCB is designed for various IC's and so this data sheet will cover the options specifically for the PO11/2



6.1. Sideways Stackable Connectors

The PCB provides a pin (left hand side) and socket (right hand side) arrangement. This will allow many PCB's to be stacked together to form a system.

The pins and socket are connected through on a 1:1 arrangement.



The +L pin goes to pin 20 of the IC and is therefore the main supply. The +V pin does not go to any part of the IC.

6.2. I/O Connector

2	4	6	8	10	12	14	16
1	3	5	7	9	11	13	15

Pin Numbering Layout

Pin	Connected to
1	AN1
2	D1

3	AN2
4	D2
5	D3
6	AN3
7	AN4
8	AN5
9	AN6
10	AN7
11	Vref+
12	AN8
13	+L
14	+V
15	GND
16	GND

I/O Connector Table

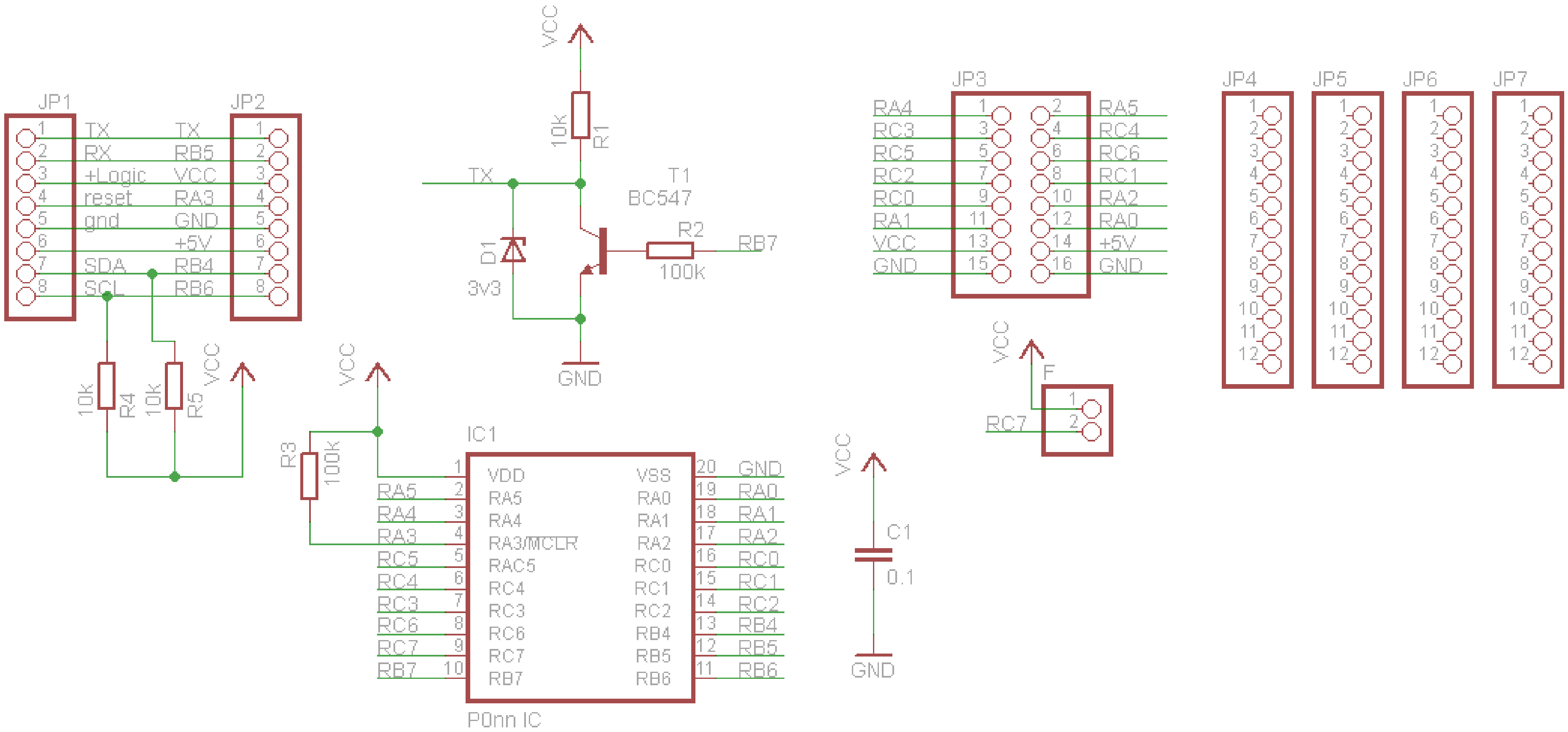
6.3. Serial Circuit

This circuit has two features:

1) It has been specially designed for use with 5V and 3V3 hosts. The device can be operated at 5V but still be connected to a 3V3 host without fear that the TX pin will be outputting 5V as this is clamped to 3v3 by the diode.

2) The output transistor is effectively an open collector device, this makes it possible to use many devices on the same serial bus without a clash of the TX lines.

7. Circuit Diagram of the POnn PCB



8. Command Set

Default address, serial ('h') I2C (0x34, 7bit)

The commands are sent to the serial interface byte by byte, however for convenience the byte values chosen coincide with ASCII characters. This makes debugging on a terminal very easy.

All commands will be referred to by their ASCII value but remember on a microcontroller host system, sending 'a' on a terminal is just the same as sending the value 97.

Where appropriate all of the serial commands listed in the summary table below have an I2C equivalent.

The full command details are listed later on in the text.

Command Set Summary	
a (1)	Get ADC channel
d (2)	Digital outputs
V (3)	Voltage reference
W (0x91)	Write to EEPROM
R (0x90)	Read from EEPROM
I	Invert TX
C (0x95)	Reset
D (0xa1)	Device ID number
V (0xa0)	Firmware Version
H (0x96)	Hello

Table 1 Command Set summary (I2C)

All of the above commands require a device address to be specified before sending the command and also **every serial command sequence must be terminated with CR** ("\r") (13) (0xd).

Serial:

The device will return ACK (6) on all successful commands and NACK (21) on unsuccessful commands.

Any command beginning with an address that does not match the devices address is ignored.

I2C:

Follows the normal I2C rules.

9. EEPROM values

The EEPROM contains important values that control the way the device behaves. All of the values can be changed by the user using the 'W' command.

The EEPROM consists of 255 bytes and in general the first 16 bytes are used by the system, the second 16 byte are used by the device and the rest of the bytes can normally be used by the user.

Adr	Default Value	Description
0	0	System Use
1	112	Device address
2	6	ACK value
3	21	NACK value
4	3	Baud rate
5	1	Error reporting 1 = on
6	13	End of line
7	1	Invert TX 1=invert
14	112	Device address copy
16	8	ADC scan rate
17	2	Voltage reference code
250	112	Device address copy

Table 2 EEPROM use

The user is free to use any locations that are not occupied by the system but for future use it is best to avoid locations below 32.

EEPROM values are only read on start up so when changing values they will not normally take effect until the device is reset.

9.1. Address

These EEPROM locations contains the device address. By convention the address is set to values between the values 97 to 122, no checking is made by the device so setting values outside this range may or may not work.

For security the address is stored in three places and to change the address of the device at least two of the locations need to be set otherwise the device will detect the anomaly at start up and revert to the majority value.

Normally to change the address of a device locations 1 and 14 are both changed. The device will detect this at start up and change the address in location 250 to match.

9.2. ACK character

By default this is 6 but can be changed using the EERPOM Write command. The effect will not be implemented until the device is reset.

9.3. NACK character

By default this is 21 but can be changed using the EERPOM Write command. The effect will not be implemented until the device is reset.

9.4. Baud Rate

The Baud rate has the following values:

0. no valid
1. Baud rate is fixed at 2400
2. Baud rate is fixed at 4800
3. Baud rate is fixed at 9600 (default*)
4. Baud rate is fixed at 14400
5. Baud rate is fixed at 19200
6. Baud rate is fixed at 38400
7. Baud rate is fixed at 57600
8. Baud rate is fixed at 115200

9.5. Turn off Error reporting

By default error reporting is enabled and this will be reported and an output prefixed by Error, for example '**Error 2**'. This may get in the way of the program trying to control the device and so it can be disabled with this command. The effect will not be implemented until the device is reset. This does not apply to I2C if available.

9.6. CR Character

By default this is 13 which is the standard ASCII CR and the whole protocol relies on this being at the end of every command. It may be that this is unsuitable in some systems and so this can be changed.

9.7. ADC San Rate

The ADC scan rate can be adjusted from 128uS to 32mS in increments of 128uS. A value of 0 should not be used. As an example to set the scan rate to 5mS (5000/128) = 39.

This should be set in the EEPROM location. As the EEPROM is only read at start up it will not take effect until reset.

9.8. Voltage Reference Code

The voltage reference has the following values:

0. Uses the Vref+ pin
1. 1.024V
2. 2.048V
3. 4.096V
4. Uses +V

Any other values will default to code 2. Obviously to use code 4 the voltage to the chip needs to be +5V.

The code set here will be the default code, it can be changed dynamically using the 'v'(3) command.

10. Commands

All serial commands are proceeded by an address and terminate with CR (0xd). In the examples given below the address is 'h' or 0x68 (0x34)

When a command is accepted by the device it always returns ACK which by default is the value 6. If the device rejects the command then it will return NACK, value 21

Serial	I2C	range	Default Value	EEPROM Location	Description
a,n	1,n	1-9			**Get ADC Value** the channel is specified after the command and the value returned will be between 0 and 1023. Channel 9 is an indication of the internal temperature of the IC, the higher the value the hotter the IC is. **Examples**: Get value of ADC channel 3 **ha3** **I2C** Two bytes need fetching for this, the high byte will be sent first **s 1 3 r g-2 p** or **bus.i2c(0x34,[1,3],2)** See www.pichips.co.uk and 'notsmb' for an explanation of the above nomenclature.
dm,n	2 m n	0-3 (m) 0-1 (n)			**Digital Output** Sends a digital value either 0 or 1 to the selected channel. The channel is the first number **Examples**: make channel 1 high **hd1,1** **I2C** **s 2 1 p** or **bus.i2c(0x34,[2,1],0)** See www.pichips.co.uk and 'notsmb' for an explanation of the above nomenclature.
vn	3 n	0 - 4			**Set Reference Voltage Code** Sets the ADC voltage reference. **Examples**: Set the reference voltage to 1.024V **hv1** **I2C** **s 3 1 p**

					or **bus.i2c(0x34,[3,1],0)** See www.pichips.co.uk and 'notsmb' for an explanation of the above nomenclature.
pn	4 n	0 - 1			**Set Weak Pull up** By default there are weak pull up resistors internally attached to the ADC lines. The feature can be switched off with this command. 0 switches off an 1 switches on. **Examples**: Switch off the weak pull ups **hp0** **I2C** **s 4 0 p** or **bus.i2c(0x34,[4,0],0)** See www.pichips.co.uk and 'notsmb' for an explanation of the above nomenclature.
SYSTEM					
Wn,m	0x91	n=0-255 m=0-255			**Write to EEPROM** This will write a single byte to an EEPROM location, the command format is: <adr>W<EEPROM address>,<value> Care should be taken when using this command for two reasons: 1) The EEPROM can only be written to a certain number of times all be it a large number. 2) The EEPROM contains system information that is used at start up a wrong value could mean loss of communication with the device that would require a factory reset. **I2C Example write 23 to location 7** **s 0x91 7 23 p** or **bus.i2c(0x34,[0x91,7,23],0)** See www.pichips.co.uk and 'notsmb' for an explanation of the above nomenclature.
Rn,m	0x90	n=0-255 m=0-255			**Read from EEPROM** The EEPROM values can be read with this command given a starting address and the number of bytes to read. <adr>R<"EEPROM adr><#bytes> This example will output the first 16 bytes of EEPROM **fR0,16** The output from the device will commence after receiving CR and will consist of a string of data terminated with ACK.

					The sting will be in the form of text delimited by ‘,’ and all of the values will be decimal. An example of output for the first 5 bytes of EERPOM would be: “0,97,6,21,0”<ACK> **I2C** I2C will read only single values at a time, to read from location 3: **s 0x90 3 r g-1 p** or **bus.i2c(0x34,[0x90,3]1)** See www.pichips.co.uk and ‘notsmb’ for an explanation of the above nomenclature.
D	0xa1				**Device ID** Returns a number representing the device product number as a string **fD** Output from the above would be: “4601”<ACK> **I2C** returns two bytes representing a 16 bit number, high byte first **s 0xa1 r g-2 p** or **bus.i2c(0x34,[0xa1],2)** See www.pichips.co.uk and ‘notsmb’ for an explanation of the above nomenclature.
I	n/a		1	7	**Toggle Inverted** This command will invert the output of the TX pin and store the value to EEPROM. A value of 1 is inverted and 0 is not inverted. The command will toggle from one to the other. If the TX pin requires changing, instead of ACK being returned by the device a value of 0x3e (‘>’) is returned. This is easily detected and this command can be issued to correct it. Example **fI** This is only needed as a **one time** operation as the change is automatically written to EEPROM
C	0x95				**Reset** Resets an individual device. This is a soft reset. A soft reset will normally be the same as a reset at start-up but this may not always be the case. Obviously no ACK will be returned by this command. Example

					fC **I2C** **s 0x95 p** or **bus.i2c(0x34,[0x95],0)** See www.pichips.co.uk and ‘notsmb’ for an explanation of the above nomenclature.
V	0xa0				**Version** Returns the firmware version as a string in the format “H.L” Example **fV** **I2C** Sends two bytes, major revision first so 2.7 world be 2 and 7
H	n/a				**Hello** This command is used to check what devices are on the bus. It simply returns ACK but where there is more then one device on the bus the following sudo code will list them: for j = 97 to 122 Send(chr$(j)+”H\r”) if ack received then print device j found If a device is found then the other attributes such as device ID can be obtained. Also user information could be stored in the devices EEPROM and retrieved.