



Using Digital I/O

A tutorial covering programming and general topics when using Digital I/O on the E-Nodes.

Contents

Companion Documentation.....	3
Overview.....	3
Using an Output as an Input.....	3
I/O Variables.....	5
'inputs' and 'outputs'.....	5
ints.....	5
'ipf0' to 'ipf9'.....	5
'input0_blank'.....	5
cap_window_hi.....	5
cap_window_ho.....	5
I/O Commands.....	6
output[].....	6
input[].....	6
pulse().....	6
edges().....	7
ints[].....	7
Input Filtering.....	8
ipf0 to ipf9.....	8
input0_blank.....	9
Interrupts.....	10
Interrupt Control.....	10
Selecting interrupt input polarity.....	10
Enabling Interrupts.....	10
int0h() - int9h().....	10
uarth().....	11
msgh().....	11
Encoder Position Capture Interrupt.....	12
Encoder capture polarity.....	12
Enabling Encoder Capture.....	12
Registration Window.....	12
Input[0] interrupt filtering.....	12
Using I/O on the 'CORE' node.....	13
Questions.....	13

Companion Documentation

Full technical detail on the Digital I/O nodes 'NPN I/O' and 'PNP I/O' can be found in the document, 'E-Nodes Technical.pdf'.

Overview

Digital I/O is provided by several different node types. In general they all conform to the same specification and offer the following features:-

- Each connection is both an Input and an Output (see explanation below).
- All are optically isolated - 2500v for 1 minute.
- All are current limit and over voltage protected.
- All require an external 24v power supply to drive the isolated side of the connection.
- They may be of PNP or NPN format.

When used as an Output:

- Outputs are set immediately the user program executes the 'output[x]' command.
- Outputs can generate timed pulses (period = 0.0005 to 60 sec. resolution = 0.0005 sec).
- Because Inputs and Outputs share a connection pulsing an Output can generate a user software interrupt if required (if enabled).

When used as an Input:

- Inputs are read every 0.5 milliseconds.
- Input filtering is provided to prevent false signalling. This is performed digitally in software by setting variables 'ipf0' to 'ipf1' (see Control-C Language). This feature is optional.
- Edge transitions on Inputs can generate user software interrupts if required.
- The number of edge transitions can be recorded using the 'edges' software command.
- On Motion Nodes Input0 may support fast encoder position capture (approximately 1microsecond).

Using an Output as an Input

► **To use an I/O connection as an Input set the corresponding Output to its inactive state.**

Inputs and Outputs share a common connection. Reading an Input simply records the voltage level on that connection. The output circuit can be set to either an active or inactive state. In the inactive state the connection is not influenced by the Output circuit and it can be used as an Input. In the active state the connection will be driven to a defined voltage level dependent on the type, i.e. PNP or NPN.

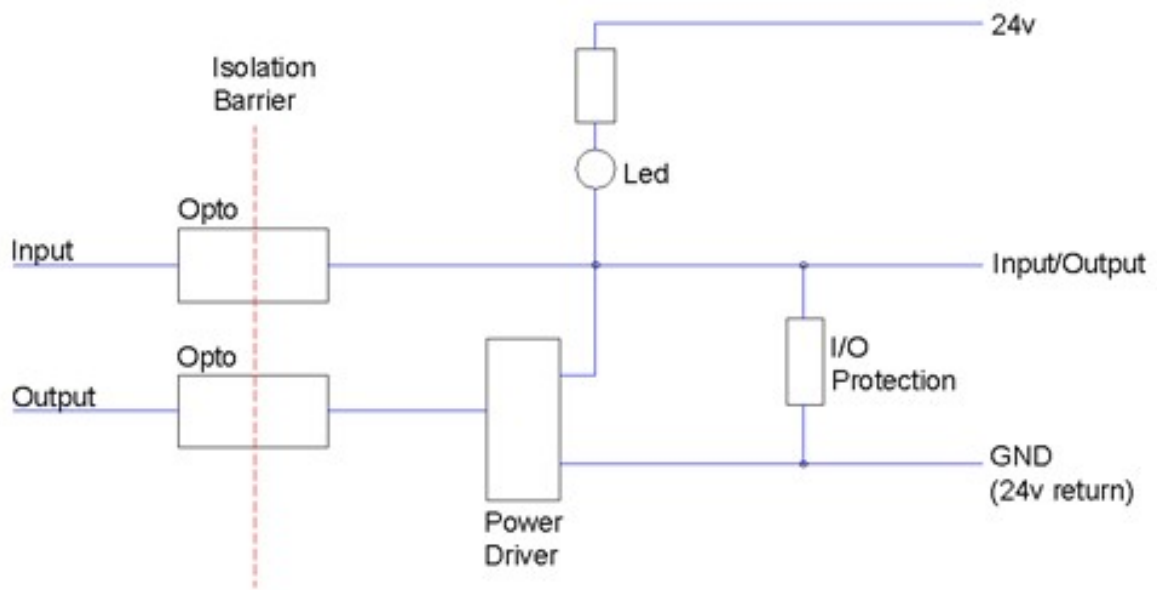
PNP

Programmed value	Output state	Connection voltage
1	Active	driven to 24v
0	Inactive	floating (weak pull down to 0v)

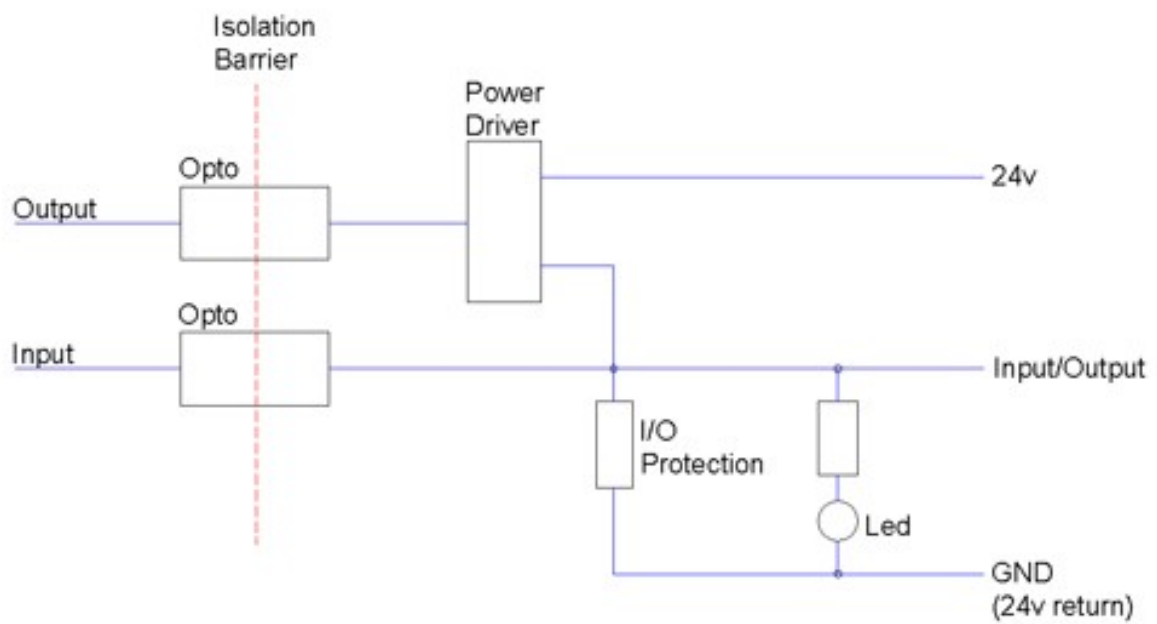
NPN

Programmed value	Output state	Connection voltage
1	Inactive	floating (weak pull up to 24v)
0	Active	driven to 0v

► **After RESET all Outputs are set to their inactive state regardless of the type.**



NPN Digital I/O circuit (simplified)



PNP Digital I/O circuit (simplified)

I/O Variables

There are a number of pre-defined variables in Control-C to support the usage of digital I/O. They are:-

'inputs' and 'outputs'

The state of the inputs and outputs are reported in the pre-defined variables 'outputs' and 'inputs'. The decimal term is ignored in both these variables. The integer term is used as an array of bits with bit 0 reporting the state of the I/O 0 connection and so on up to bit 31 (used by the CORE node). By reading and writing to these variables the programmer can alter their states in one operation. 'inputs' is a read only variable.

Example:-

```
outputs = 0; // float all PNP outputs

outputs = 0x000003FF; // PNP outputs set to 24v

if(inputs & 0x00000001) // test input 0
{
    // do this
};
```

ints

'ints' is used as an array of bits to store the status of user interrupts. It is a read only variable. Changes are effected using the 'ints()' command.

'ipf0' to 'ipf9'

Used for digital filtering of the inputs. See 'Input Filtering' below.

'input0_blank'

Used to blank out periods on input[0] for the prevention of multiple triggering.

cap_window_hi

Used to hold the most +ve value for the encoder capture window generate from interrupts on input[0].

cap_window_ho

Used to hold the most -ve value for the encoder capture window generate from interrupts on input[0].

I/O Commands

Following is a current list of dedicated I/O commands provided by Control-C. Please see 'Control-C Language xxx.pdf' for full details.

output[]

Used to read and write to individual outputs on any node address.

Example:-

```
output[0:2] = 1; //set output 0 on node 2

if(output[0]==1)
{
    // do this
};
```

input[]

Used to read the input state on any node address.

Example:-

```
if(input[0]==1)
{
    // do this
};

// or simpler still

if(input[0])
{
    // do this
}
```

pulse()

Used to generate a pulse on a given output. A pulse is generated by immediately toggling the output, then waiting the required pulse width time period, then toggling the output again. Thus the initial state of the output determines whether the pulse is high going or low going.

Example:

```
pulse(0,10); // 10 millisecond pulse on output 0, default address

pulse(0:1,100); // 100 millisecond pulse on output 0, address 1
```

edges()

Used to count edge transitions on inputs. Rising, falling or both edges can be counted.

When 'edges' is executed the edge count for the given input is cleared and the given edge type is selected. From that time onward the chosen edge types are counted. The current edge count is returned when 'edges' is called as a factor. 'BUILTIN' constants are provided for use with 'edges'.

Example:

```
var inedges;  
edges (1:1, BOTH);           // count both edge types, input[1] node 1  
delay (10);                 // wait 10 seconds  
inedges = edges (1:1);      // read the edge count
```

▶ *The edge count is updated every 500 microseconds. Therefore edge counting can only reliably record edges with a time period greater than 500 microseconds between edge changes.*

ints[]

Used to set and clear the bit array 'ints' that contains setup and status information for interrupts. See interrupts section below.

Input Filtering

All digital inputs are optically isolated providing a degree of filtering for very fast transients signals (below 1 microsecond). To filter out unwanted signals of lower frequency digital inputs are provided with optional digital filtering with the use of pre-defined variables 'ipf0' to 'ipf9'.

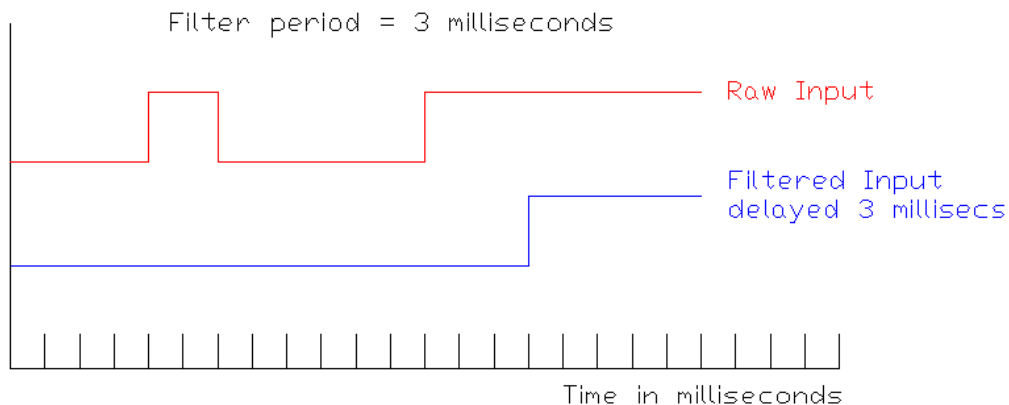
Input 0 also simultaneously performs encoder capture if enabled. This function is processed separately from regular I/O and does not have digital filtering because it needs to react very fast. To help reject unwanted trigger signals this function is provided with a blanking facility by the use of pre-defined variable 'input0_blank'.

ipf0 to ipf9

Function: Stands for 'input filter x', where 'x' is the digital input number.
Digital inputs can have filtering applied to block noise or unwanted signal pulse widths. This variable holds the number of digital I/O loop periods that an input must remain at a stable level before a change in that level will be reported.
Digital I/O is reported every 500 microseconds, therefore an 'ipf' value of 6 will be required to generate a 3 millisecond filter window.

▶ *Filtering is disabled if the value is set to zero.*

Units: digital I/O loop counts
Range: 0 to 1000 (decimal term not used)
Access: read / write
Node type: all nodes with digital inputs



▶ *Digital input filtering delays the input signal by the filter period. Therefore any action taken on changes to inputs including user interrupts will also be delayed by the filter period.*

▶ *Note: Very fast signal glitches less than 1 microsecond are very unlikely to get past the standard opto isolation barrier present on all inputs.*

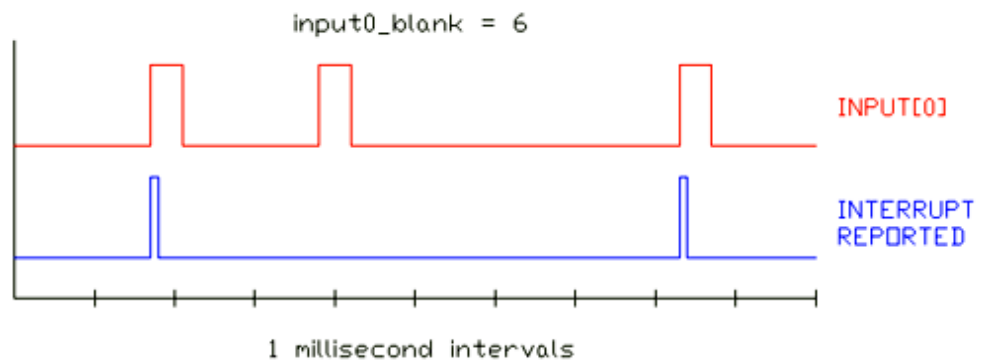
▶ *Note: High speed position capture interrupts on input[0] are processed independently of normal I/O reporting and are NOT effected by digital input filtering. However they do have an input signal blanking feature see 'input0_blank'.*

input0_blank

Function: The number of digital I/O loop periods that 'input[0]' must remain stable for, before a valid change will generate an encoder position capture interrupt. The digital I/O loop period is 500 microseconds therefore a value of 4 will generate a 2 millisecond blank period.

- ▶ *This feature only effects position capture interrupt processing and is completely independent from normal digital I/O processing*
- ▶ *Note: This feature does not delay the reporting as with normal digital I/O filtering.*

Units: milliseconds
Range: 0 to 400 (decimal term not used)
Access: read / write
Node type: all



Interrupts

The E-node range supports 12 different interrupts. 10 activated from digital I/O, 1 activated by character receipt on the uart and 1 activated by a message receipt from another node. Parameters cannot be passed to interrupts and return values are not allowed. The main program is temporarily suspended whilst interrupt handlers are active.

- ▶ *Interrupts are acknowledged according to their priority:*

```
int0h    =    Highest priority
msgH     =    Lowest priority
```

- ▶ *Only a single interrupt can be active at any one time.*
- ▶ *Pending interrupts are acknowledged according to their priority.*
- ▶ *If a particular interrupt is called for a second time whilst that same interrupt is either pending or currently running the second call will be ignored and discarded.*

Interrupt Control

Interrupts are controlled by the setting and clearing of bit fields in the 'ints' variable. A range of 'BUILTIN' constants are provided for this purpose. For input interrupts the edge type (either rising or falling) can be selected. Individual interrupts must be enabled for them to become active and the global enable bit 'GLOBAL_EN' must also be set. Clearing the global enable bit disables all interrupts regardless of the individual setting status.

Selecting interrupt input polarity

Polarity bit = 1 selects RISING EDGE
Polarity bit = 0 selects FALLING EDGE

Example:

```
ints[INT0_POL] = RISING; // select rising edge
```

Enabling Interrupts

Global enabling is achieved using the 'GLOBAL_EN' constant. Setting the bit enables and clearing the bit disables.

Example:

```
ints[GLOBAL_EN] = 1; // enable all individually enabled interrupts
ints[GLOBAL_EN] = 0; // disable all interrupts
```

Individual interrupts must be enabled in addition to the global enable. Use the 'INT0_EN' to 'INT9_EN', 'UART_EN' and 'MSGH_EN' constants. Setting the bit enables and clearing the bit disables.

Example:

```
ints[INT0_EN] = 1; // enable interrupt 0
ints[INT1_EN] = 0; // disable interrupt 1
ints[UART_EN] = 1; // enable uart interrupt
```

int0h() - int9h()

These are the optional pre-defined interrupt handlers for digital inputs 0 through 9. They can be called using either rising or falling edges occurring on the inputs (see Interrupt Control above)

uarth()

This is the optional pre-defined interrupt handler for character receipt on the uart.

Example:

```
uarth()  
{  
  if(charin())  
  {  
    //user code  
  };  
}
```

msgh()

This is the optional pre-defined interrupt handler for message receipt. If enabled this interrupt will be executed when a message is received from another node and placed in the message buffer.

- ▶ *Messages are sent from node to node using the 'msg' statement.*
- ▶ *The format of a single message is the same as a standard variable.*

Encoder Position Capture Interrupt

Control-C supports fast encoder position capture for motion capable nodes. A rising or falling edge on input[0] can be selected to initiate the capture, the result of which is placed in the pre-defined variable 'cappos'. To prevent false triggering or exclude unwanted input[0] triggers a registration window feature is provided.

Encoder capture polarity

To select either rising or falling edge polarity for the encoder capture use the 'status[]' command as follows:-

```
status [CAP_POL] = RISING; // encoder capture edge polarity = rising
status [CAP_POL] = FALLING; // encoder capture edge polarity = falling
```

Enabling Encoder Capture

The encoder position capture function is entirely independent of the other interrupt functions on the E-Nodes. To enable a capture use the 'status[]' command as follows.

```
status [CAPTURED] = 0; // clear captured flag and enable a new capture
```

This clears the captured flag and enables another capture to take place. When a new capture is performed the captured value is placed in the pre-defined variable 'cappos' and the captured flag is set to 1. No further captures will take place until the captured flag is set to 0.

To determine if a capture has taken place test the captured flag as follows:-

```
if (status [CAPTURED])
{
    // do this
};

// or use this

while (!status [CAPTURED]) {};
```

Registration Window

To prevent unwanted triggers a registration window facility is provided. The pre-defined variables 'cap_window_hi' and 'cap_window_lo' can contain values representing the most +ve and most -ve encoder values. If the capture interrupt occurs when the encoder position is within these bounds the position will be captured and reported. If outside these bounds the interrupt will be ignored.

► If cap_window_hi and cap_window_lo are both set to 0 the registration window facility is disabled.

Input[0] interrupt filtering

Encoder capture interrupts generated by input[0] are very fast acting. This interrupt is processed separately from the normal interrupts and is not influenced by the digital filtering facility. However to help prevent noise and false triggers occurring a blanking facility is provided. Please refer to the definition for 'input0_blank' given in the pre-defined variables section.

Using I/O on the 'CORE' node

The 'CORE' node supports up to 32 inputs and 32 outputs. The user program must explicitly enable them as required before use.

Example:

```
status[OP_DI_0_7] = 1; // enable digital inputs 0 to 7
status[OP_DI_8_15] = 1; // enable digital inputs 8 to 15
status[OP_DI_16_23] = 1; // enable digital inputs 16 to 23
status[OP_DI_24_31] = 1; // enable digital inputs 24 to 31

status[OP_DO_0_7] = 1; // enable digital outputs 0 to 7
status[OP_DO_8_15] = 1; // enable digital outputs 8 to 15
status[OP_DO_16_23] = 1; // enable digital outputs 16 to 23
status[OP_DO_24_31] = 1; // enable digital outputs 24 to 31
```

Questions

If you have any questions regarding either this tutorial or any other aspect of using the E-Node range please contact the staff at Etrol Ltd.

Tele: (+44) 01730 816893
email: sales@etrol.co.uk