



Tutorial

Designing and Setting up an E-Node Control System

This tutorial goes through the complete process of analysing and defining a control system for a simple machine that cuts paper to length using registration with two motors, a touch screen and a limited number of sensors.

Contents

Introduction.....	3
The Three Laws.....	3
Law 1. Keep It Simple (KIS).....	3
Law 2. Break It Down (BID).....	3
Law 3. Make It Modular (MIM).....	3
Analysing the system.....	4
Building and Commissioning the System.....	6
Step 1. Assembly.....	6
Step 2. E-Net termination.....	6
Step 3. Node Configuration.....	6
Step 4. Commissioning.....	7
Step 5. Programming.....	8
Making Additions.....	11
Questions.....	11

Introduction

This tutorial assumes the reader is familiar with the basics of the E-Node range and Control-C. It also assumes that Control-C is installed on your computer and that you are familiar with its use. If not please refer to the Etrol web site downloads page where all the documents and software are freely available.

The E-Nodes are a range of small control nodes each with its own facilities. They are all connected together via the built in E-Net network. They are all independently programmable and they share their facilities over the E-Net.

At the heart of the E-Node design philosophy is the concept of 'Modularity'. The node themselves are modules and the software they are programmed with can be created in modules. Control systems using the E-Node are intended to be built in a modular fashion.

The E-Nodes themselves are relatively simple and so is Control-C the programming language. However they can be combined into systems to perform very powerful and complex tasks. The method used to design and build these systems is very important. Imagining the E-Node modules are like house bricks. If you follow good building practice a long lasting and beautiful house may result. However if you ignore good building practice an ugly unsafe monster of a house will probably be the case.

This leads us to the three laws.

The Three Laws

To help you build good quality control systems that are easy to understand, reliable and easy to maintain we offer the following laws

Law 1. Keep It Simple (KIS)

Whatever you are doing, keep it simple. If it is not simple then use Law 2. and break it down until it becomes clear and understandable.

Law 2. Break It Down (BID)

Break all parts of the control system down into modules with each module having a clear defined function. Here we are referring to both the physical system and the software used to program it.

Law 3. Make It Modular (MIM)

By studying the needs of your proposed control system and applying 'KIS' and 'BID' you should be able to build the entire system from interconnected modules.

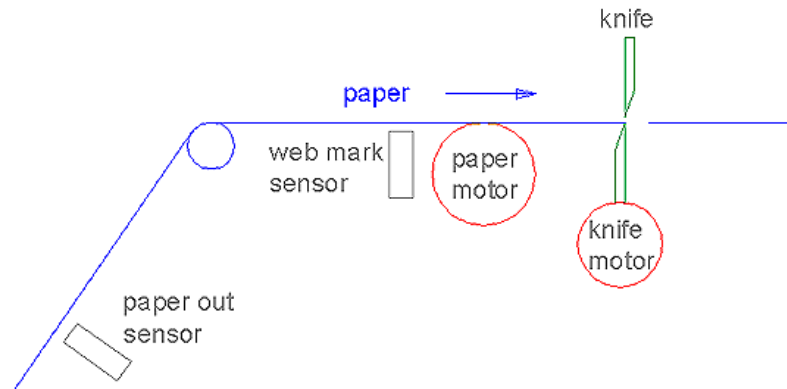
If you follow the three laws your control system should be a hierarchy of modules much like a family tree in which the top level is broken down into lower level modules and so on until the bottom level is reached. Ideally the bottom level modules will all be very simple and easy to understand and the module tree will describe the function of the whole control system.

If your control system is simple to start with then the three laws may not be needed however its always good to keep them in mind because control system have a habit of growing. When the boss sees what you've created he will always ask what else it can do, how long it will take and how much it will cost.

Analysing the system

The first step in designing your control system is to analyse its needs. From this you can create a list of the components required to build the system.

For this tutorial we shall analyse a simplified example of a paper cutting machine as shown in the following diagram:-



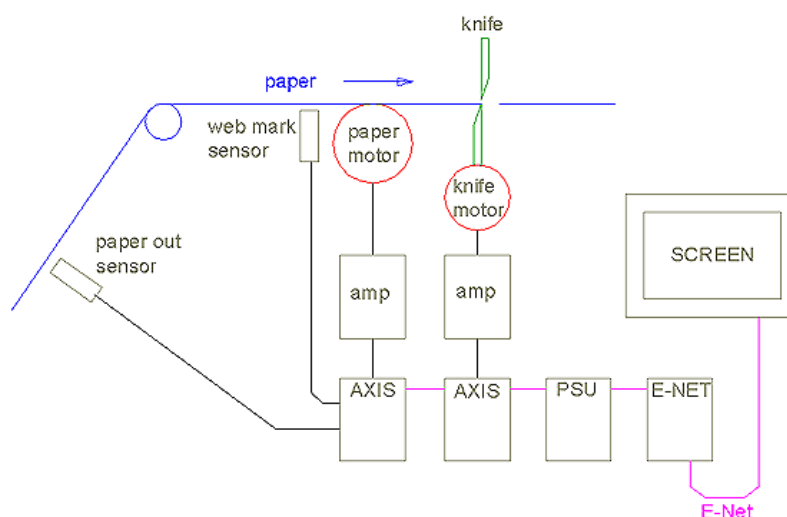
This machine cuts continuous paper web to length. The length is determined by regular print marks on the underside of the web. A 'paper motor' advances the paper web and a 'knife motor' activates the cross cutting knife. A 'web mark sensor' records the position of the web mark and a 'paper out sensor' senses if paper has run out.

In operation we want the machine to advance the paper until the web mark is seen and then stop a given distance from that point. The paper should then be cut by the knife. The whole process then repeats continuously until the paper runs out or its commanded to stop.

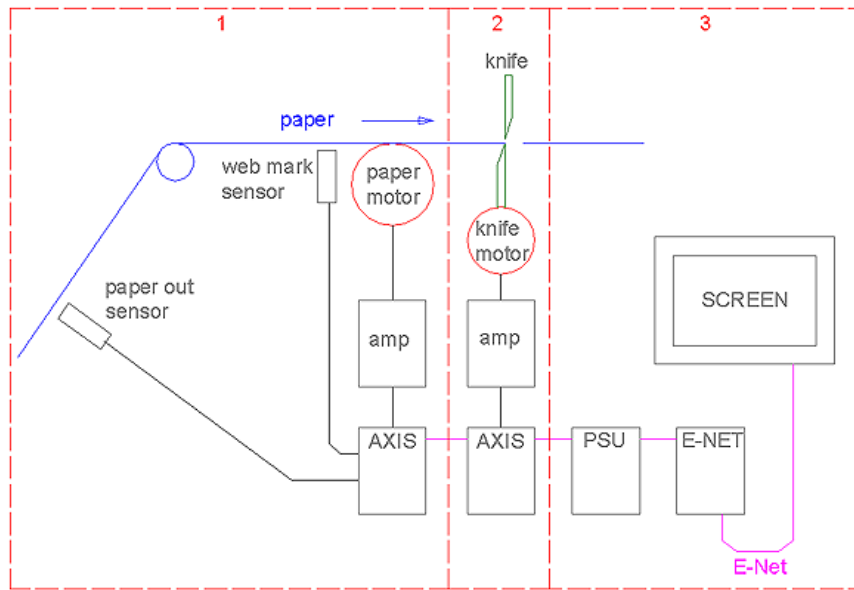
Analysing this machine the following list of requirements can be compiled:-

- Amplifiers for the motors..
- An AXIS node to control the paper motor.
- An AXIS node to control the knife motor.
- A means of inputting data and control commands - we will use a SCREEN node.
- A PSU-24V node to power the AXIS and SCREEN nodes.
- An E-NET node to interface the two AXIS nodes to the SCREEN using a LINK CABLE. We can then site the SCREEN remotely on the end of the long LINK CABLE.

Adding these components to the schematic we have the new diagram:-



This system is rather simple and might not require breaking into modules but to illustrate the method we shall use three modules split in terms of function.



- Module 1. Paper motor control.
- Module 2. Knife motor control.
- Module 3. Command and control.

Having split the system into these modules you can now see what the programs in the different modules have to do:

Program functions are:-

- Module 1.
 - ◊ Observe go and stop commands from the SCREEN.
 - ◊ Advance paper.
 - ◊ Observe the web mark and stop the paper a given distance from the mark.
 - ◊ Inform the knife module when the paper is stopped and ready to be cut.
 - ◊ Observe the paper out sensor and act accordingly.
- Module 2.
 - ◊ Activate the knife motor on command.
- Module 3.
 - ◊ Accept command entries from the operator
 - ◊ Communicate commands to the other modules over the E-Net.

That is the process of analysis complete. You can see that by breaking the system down into simple modules it becomes easy to see which functions are best undertaken by each module. You have also generated a parts list and defined the program structure for each module. This is a very simple system but later on in the tutorial we will see how this simple system can easily be enhanced without problem now that the modular approach is in place.

► *Please note: To maintain clarity we have deliberately omitted items such as control of amplifier power at this stage.*

Building and Commissioning the System

Here we will only deal with the building and commissioning aspects associated with the E-Nodes and we will not go into detailed wiring schemes. Please refer to the E-Node Technical documentation for full details.

Step 1. Assembly

Assemble the four E-Nodes units on a length of standard DIN rail. Mount to PSU at one end followed by the two AXIS nodes followed by the E-Net Node. Connect all four with length of E-Net ribbon cable (under the top hinged covers).

Wire up the rest of the system.

The web mark sensor must use 'input[0]' to allow position registration to be used.

External 24v needs to be supplied to the isolated side of the AXIS for the digital I/O to function.

Use output[2] to enable the associated amplifier on each AXIS node.

Connect the amplifier and encoder feedback to the AXIS nodes in accordance with the data sheets.

Mount the SCREEN node in a suitable position for the operator to use. Connect an E-Net LINK cable between the SCREEN (left most connector) and the E-NET node.

Place a wire link between pins 1 and 2 of E-NET node 'Con B' to pass power out on the 'LINK Cable' to the SCREEN. Move the link on the SCREEN 'J1' to position 2-3 to accept power from the 'LINK Cable' to power the SCREEN.

Step 2. E-Net termination

The 'CAN' based part of the E-Net must have 120 ohm terminations at the extreme ends of the network. In this case the whole network is 'CAN' based. The SCREEN is at one end and the farthest AXIS is at the other. On the rear of the SCREEN jumper J3 must be placed in link position 4-5. On the farthest AXIS the link under the top hinged cover must be placed in the position closest to the ribbon cable connector (please see the E-Nodes technical document for more details). The link on the second AXIS must be in the 'park' position.

Step 3. Node Configuration

We must now configure the nodes. Before programming or commissioning can take place every node in the system must be programmed with a unique network address and 'power on run' program status. You need to choose the addresses you wish to use. They are a vital part of the programming task and should remain fixed once they have been set.

We will make the addresses the same as the module numbers.

Power up the system. Start Control-C on your P.C. Connect the 'Adaptor Cable' between the P.C. and the paper AXIS node.

► *Note: The Adaptor cable must be plugged directly into the node you wish to configure.*

Open the 'Configure Node' window (Ctrl+Alt+C). Set the 'Full Node Address' to 1. and check 'Run Program on Node Power Up'. Click the 'Program' button. Do the same for the knife AXIS node and the SCREEN node using addresses 2 and 3 respectively. Make certain the cable is moved to the node in question each time.

You are now ready to start commissioning and programming the system.

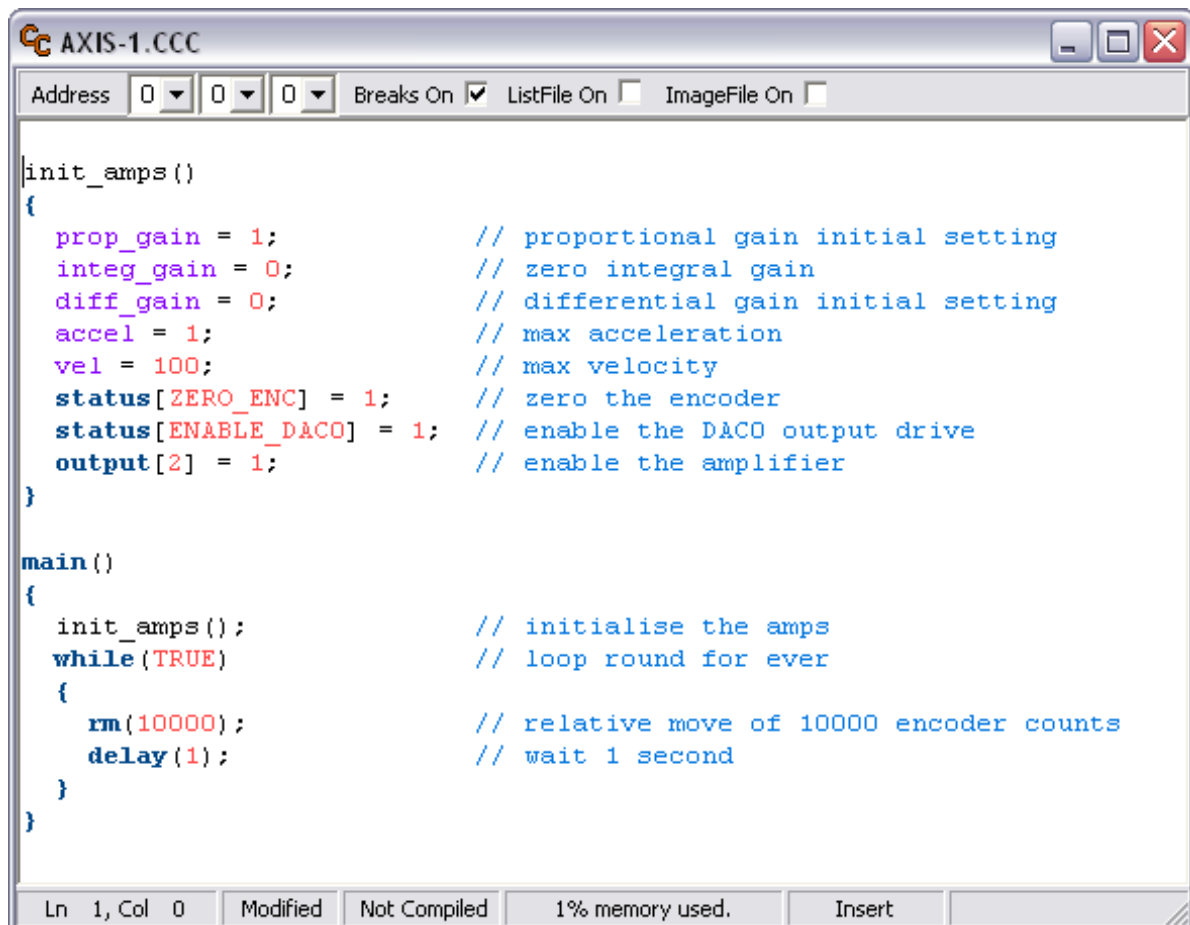
► *Note: After the nodes have been configured normal programming and communications to any node can use any E-Port connection.*

Step 4. Commissioning

First we will test the web sensors. With the system powered up and the 'Adaptor Cable' connected to any node open the I/O window (Ctrl+F2) in Control-C. Select address = 1. The status of the paper AXIS should now be displayed. Exercise the sensors and check the action is reported in the I/O window.

Next we will commission the amplifiers/motors. The optimum configuration is to set up the amplifiers in velocity control mode and use the AXIS nodes to perform the position control loop. Please refer to the amplifier data sheet to achieve this set up. Our aim is to set optimum values for 'Proportional gain' and 'Differential gain' within the AXIS to give fast stable position moves of the motors.

Open a new file window and create the following program:-



```
AXIS-1.CCC
Address 0 0 0 Breaks On [x] ListFile On [ ] ImageFile On [ ]

init_amps()
{
    prop_gain = 1;           // proportional gain initial setting
    integ_gain = 0;         // zero integral gain
    diff_gain = 0;         // differential gain initial setting
    accel = 1;             // max acceleration
    vel = 100;             // max velocity
    status[ZERO_ENC] = 1;   // zero the encoder
    status[ENABLE_DAC0] = 1; // enable the DAC0 output drive
    output[2] = 1;         // enable the amplifier
}

main()
{
    init_amps();           // initialise the amps
    while(TRUE)           // loop round for ever
    {
        rm(10000);        // relative move of 10000 encoder counts
        delay(1);         // wait 1 second
    }
}

Ln 1, Col 0 Modified Not Compiled 1% memory used. Insert
```

In this program we are assuming that the motor has an encoder giving 10000 counts/rev. Compile (F5), Download (F6) and Run (F7) the program. You may need to select different values for 'accel' and 'vel' and also a different value for the relative move.

When running the motor should advance 10000 counts (1 rev) every second. Now open the variables window (Ctrl+F3) and set the address to 1, then scroll down until 'prop_gain' and 'diff_gain' are visible. Adjust prop-gain by increasing the value until a crisp movement is made every second and it stops sharply at the end of the move. If you go too large the motor will start to oscillate or ring at the end of each move. This may be cured by adding a small amount of 'diff_gain'.

Having optimised the values save the program and then repeat the exercise for the 'knife AXIS'.

This is a simplified procedure. You will probably find there are other small tasks that require completion in addition to those detailed here.

You should now be in a position to start the programming.

Step 5. Programming

We now need to create a program for each of the three modules but before doing this we must decide on one important issue, 'How will we communicate between the programs'? There are several possible techniques we can choose from. These are detailed in another tutorial document. We will choose the method of writing directly from one node to another into declared global variables. No additional wiring is required, the E-Net performs the task automatically.

Now lets take a look at the three programs in more detail starting with module 1. Here is the list of operations to be performed.

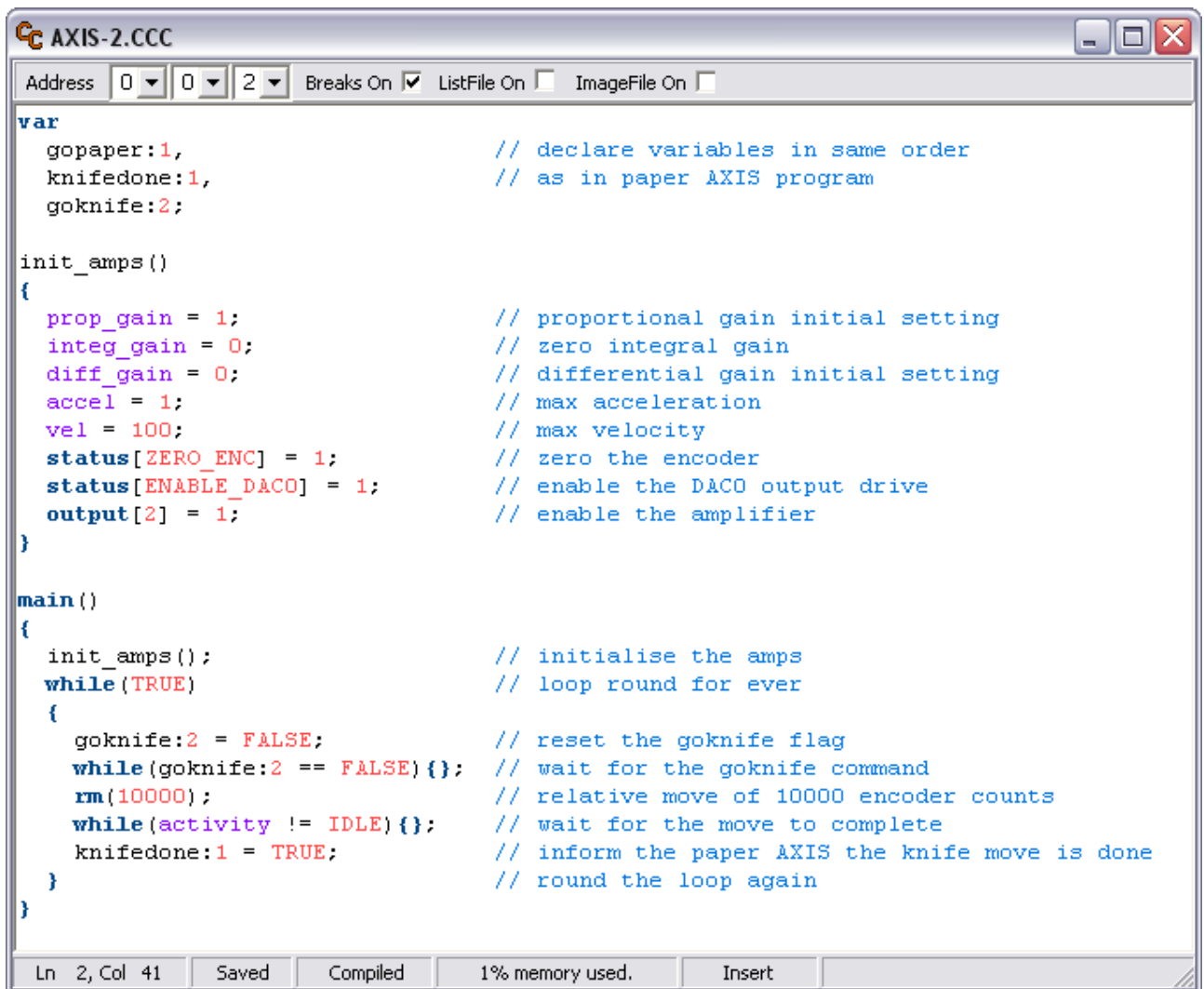
- ◇ Observe go and stop commands from the SCREEN.
- ◇ Advance paper.
- ◇ Observe the web mark and stop the paper a given distance from the mark.
- ◇ Inform the knife module when the paper is stopped and ready to be cut.
- ◇ Observe the paper out sensor and act accordingly.

And here is the program required to perform them:-



```
AXIS-1.CCC
Address 0 0 1 Breaks On ListFile On ImageFile On
define
  OFFSET 1000; // declare the constant OFFSET
var
  gopaper:1, // declare variables
  knifedone:1,
  goknife:2;
init_amps()
{
  prop_gain = 1; // proportional gain initial setting
  integ_gain = 0; // zero integral gain
  diff_gain = 0; // differential gain initial setting
  accel = 1; // max acceleration
  vel = 100; // max velocity
  status[ZERO_ENC] = 1; // zero the encoder
  status[ENABLE_DAC0] = 1; // enable the DAC0 output drive
  output[2] = 1; // enable the amplifier
}
main()
{
  status[CAP_POL] = 0; // select falling edge for encoder capture polarity
  gopaper:1 = FALSE; // initialise gopaper
  init_amps(); // initialise the amps
  while(TRUE) // loop round for ever
  {
    while(gopaper:1 == FALSE) {}; // wait for the gopaper command
    rm(100000); // relative move of 100000 encoder counts
    status[CAPTURED] = 0; // clear captured flag
    while(status[CAPTURED]==0) {}; // wait for new web mark capture
    demand = cappos + OFFSET; // adjust stopping position
    while(activity != IDLE) {}; // wait for the move to complete
    knifedone:1 = FALSE; // reset the knifedone flag
    goknife:2 = TRUE; // inform the knife AXIS to go
    while(knifedone:1 == FALSE) {}; // wait for the knife move to complete
  } // round the loop again
}
```

Module 2, the knife AXIS only has to move the motor one rev on command and then report that is has completed the task. Here is the program:-



```
CC AXIS-2.CCC
Address 0 0 2 Breaks On [x] ListFile On [ ] ImageFile On [ ]

var
gopaper:1, // declare variables in same order
knifedone:1, // as in paper AXIS program
goknife:2;

init_amps()
{
prop_gain = 1; // proportional gain initial setting
integ_gain = 0; // zero integral gain
diff_gain = 0; // differential gain initial setting
accel = 1; // max acceleration
vel = 100; // max velocity
status[ZERO_ENC] = 1; // zero the encoder
status[ENABLE_DAC0] = 1; // enable the DAC0 output drive
output[2] = 1; // enable the amplifier
}

main()
{
init_amps(); // initialise the amps
while(TRUE) // loop round for ever
{
goknife:2 = FALSE; // reset the goknife flag
while(goknife:2 == FALSE) {}; // wait for the goknife command
rm(10000); // relative move of 10000 encoder counts
while(activity != IDLE) {}; // wait for the move to complete
knifedone:1 = TRUE; // inform the paper AXIS the knife move is done
}
}

Ln 2, Col 41 Saved Compiled 1% memory used. Insert
```

► Note: It is important to declare the shared global variables in the same order in all programs that use them. This ensures that the compiler is correctly informed of the location in memory that these variables occupy.

Module 3, the SCREEN program simply has to command the paper AXIS to go or stop. So this simple program consists of two keys on a blank background. Here is the program:-

```
cutter SCREEN.CCC
Address 0 0 3 Breaks On ListFile On ImageFile On
define
  GO 0, // define key value constants
  STOP 1;

var
  gopaper:1, // declare variables in same order
  knifedone:1, // as in paper AXIS program
  goknife:2;

main()
{
  var k;
  bcolour = WHITE; // set background colour
  clear(); // clear the whole screen to the bcolour
  fcolour = BLACK; // set foreground colour
  font = NARROW20; // select a nice large font
  key(GO, 50, 60, 150, 160, "GO"); // place the GO key
  key(STOP, 160, 60, 260, 160, "STOP"); // place the STOP key
  while(TRUE) // loop round for ever
  {
    if(keypressed()) //if a key is pressed
    {
      k = getKey(); // get the key value
      if(k == GO) gopaper:1 = TRUE; // test the key value and act
      if(k == STOP) gopaper:1 = FALSE; //
    }
  } // round the loop again
}
```

Ln 26, Col 63 Saved Compiled 1% memory used. Insert

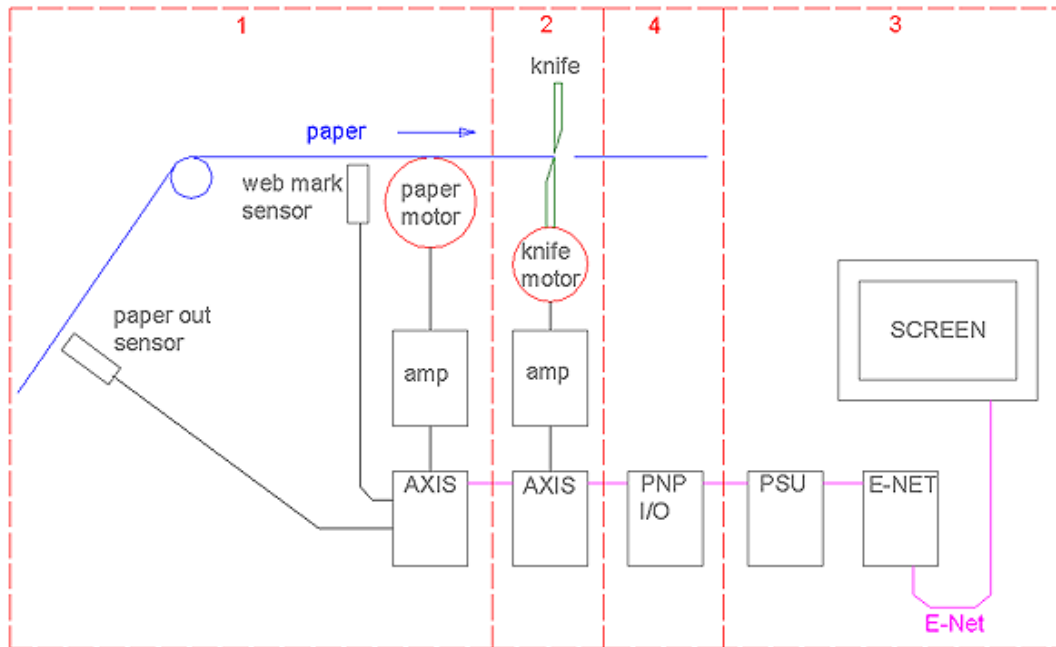
And here is the screen image it creates:-



That concludes the commissioning.

Making Additions

Now let us assume that a selection of additional I/O is required. It is a simple matter to add a 'PNP I/O' node as shown below:-



The new module is designated number 4 and the PNP I/O node is programmed with the address 4. Because it automatically becomes a network member it is very simple to use. It does not need a dedicated program but can be used from any of the other programs by using standard commands.

Example:

```
output[0:4] =1; // output 0 address 4 = 0
if(input[0:4]) {} // if input 0 address 4 == 1 then etc
```

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