i³

...Display, Control, Communicate

IMO i³

i³ display
control
communicate

i³ IMO
Operator
Control
Station

HMI | Logic | I/O | Net | DeviceNet

Tutorial

HSC & PWM Tutorial

IMO

Table of Contents

Introduction

The i^3 has a two channel PWM output that can be configured into 3 different modes. Possible configurations are: 2x High Speed Counter (HSC) function, 2x Pulse Width Modulation (PWM) function (or a mix of the two) and 1x Stepper function.

This output function allows the i^3 to be used in motion control applications by operating a stepper motor (through its driver) to a high level of accuracy. If the PWM / HSC output function is used then the i^3 can control two axis or, a single Stepper motor axis.

The purpose of this tutorial is to demonstrate the Pulse Width Modulation output function of the i^3. Only the models with transistor outputs support the PWM function. In this tutorial we will demonstrate all the functions related to the PWM function and demonstrate the i^3 controlling a Stepper Motor following a pattern.

Stepper and Servo Motors

A stepper motor rotates in defined steps depending up on the resolution of the motor. For example an 8 bit digital controller will give a resolution of $360°/256$ giving a resolution of 1.40625. Given this level of accuracy stepper motors are widely used in motion control and positioning applications.

Attached to the shaft of the motor is a series of permanent magnets and around the body there is a series of coils that creates a magnetic field when a charge is applied. To make the shaft rotate the coils must be pulsed on and off constantly, the sequence in which coils are switched on determines the direction of the motor.

Stepper motors could be used with an encoder to determine the exact real position of the shaft however they are generally used in an "open-loop" control system.

A servo motor is similar to standard electric motor in that it doesn't have predifined steps and is less complicated in terms of magnets to coils. The accuracy of a servo motor depends on the feedback system and unlike Stepper motors, a Servo motor must be implemented in a closed loop system. With the motor being less complicated the Servo Drive controller circuitry will be a lot more complex than that of a Stepper motor.

PWM Register Map

All three PWM output functions use the same registers but their use depends on the motor type connected.

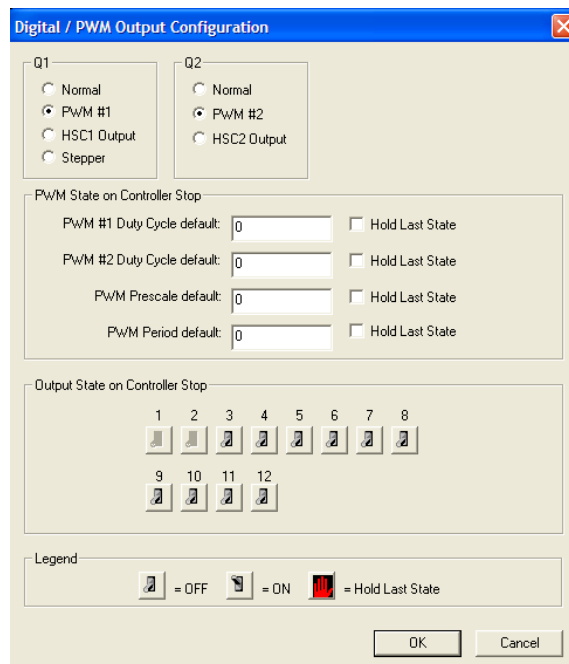| Register	PWM	HSC	Stepper
%AQ1	PWM1 Duty Cycle (32-bit)	HSC1 Preset Value	Start Frequency
%AQ2			Run Frequency
%AQ3	PWM2 Duty Cycle (32-bit)	HSC2 Preset Value	Accel Count (32-bit)
%AQ4			
%AQ5	PWM Prescale (32-bit)		Run Count (32-bit)
%AQ6			
%AQ7	PWM Period (32-bit)		Decel Count (32-bit)
%AQ8			
%Q1			Run
%I30			Ready/Done
%I31			Error

Configuring the PWM Functions

The PWM outputs are configured in the Config I/O menu and then the registers shown are manipulated.

Configuring the PWM Outputs

To set up the PWM function click on the Config I/O icon or select the option from the drop down menu.

To enable the two PWM outputs select the appropriate function

It is possible to set up the PWM function in the I/O config menu, but in this tutorial we will use registers.

It is only possible to select Stepper on Q1 as it uses Q2 registers. Q2 will only then be a digital direction bit.

Function

Having selected Q1 or Q2 as a PWM output we now are required to use the registers as shown below.

Register	PWM
%AQ1	PWM1 Duty Cycle (32-bit)
%AQ2	
%AQ3	PWM2 Duty Cycle (32-bit)
%AQ4	
%AQ5	PWM Prescale
%AQ6	(32-bit)
%AQ7	PWM Period
%AQ8	(32-bit)

Both outputs are configured to the same frequency, while the pulse width can be adjusted on each output independently. The PWM functions require three parameters (%AQ registers) to be set for operation. These parameters may be set whilst in run-time, that is the user can enter a new value through the HMI.

Prescale – This sets the resolution of the PWM output. Fore most applications this can set this to 15, which will give a resolution of 1 microsecond in terms of period and duty cycle.

The frequency of the PWM output is calculated using the following formula:


$$Frequency =$$

Period – This value (%AQ7-8) sets the period of the output signal by specifying the number of internal PWM counter *counts* before the cycle is reset (larger count results in a smaller frequency).   The duration of each *count* is determined by the pre-scale value. This parameter affects the Period of both PWM outputs

The formula above shows how the pre-scale and period create an output frequency. For example if the PWM were set for 1 microsecond resolution, a value of 20,000 would result in a 50 Hz output.
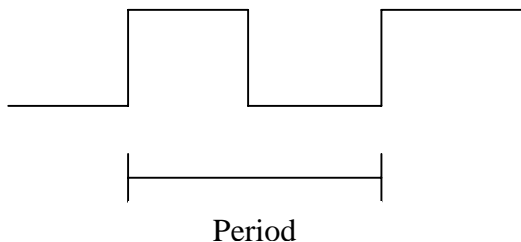
Period

This value sets the period of the output signal. This value determines the width of the output wave. A longer period results in a smaller frequency. The numeric value entered here is the number of counts for the pulse width. The duration of each count is set by the value of the pre-scaler.

.

Duty Cycle Count - This value (PWM1: %AQ1-2, PWM2: %AQ3-4) sets the width of the output signal by specifying the number of internal PWM counter *counts* that the output is maintained high. The duration of each *count* is determined by the pre-scaler value. Each PWM channel has its own duty cycle count parameter.
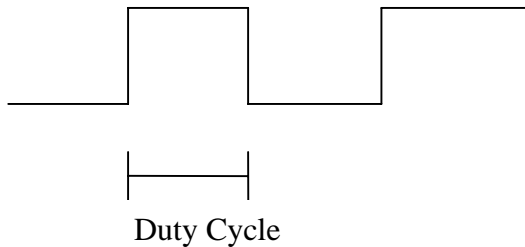
Duty Cycle

> The duty cycle determines the amount of time the output wave spends high. If the period is set to 1000 and the duty cycle is set to 500 it would result in a duty cycle of 50 percent. A duty cycle value of 250 would result in a duty cycle of 25 percent

At controller power-up or during a download, the PWM output is maintained at zero until both the Period (count) and the Duty cycle (count) are loaded with non-zero values. When the controller is placed in stop mode, the state of the PWM outputs is dependent on the *PWM State on Controller Stop* configuration. This configuration allows for either hold-last-state or specific pre-scale, period and duty cycle counts. Specifying zero for either the period or duty causes the PWM output to remain low during stop mode.

Note that the nominal output driver turn-on-time delay (to reach 50% output) is 25 microseconds. Therefore, this limitation should be considered when determining both the minimum pulse width and the duty cycle accuracy of the application.

HSC Output Function

HSC (High Speed Counter)

When either Q1 or Q2 is configured for HSC operation, HSC1 or HSC2 totalize functions are extended to allow respective direct output control based on a comparison of the current count and a preset value (PV). See totalize in the HSC section above for more information.

Totalize

In totalize mode, the accumulator is simply incremented each time the input transitions in a specific direction. Totalize mode is configurable to specify the edge (rising or falling) on which the accumulator is incremented.

Rising Edge Signal   Falling Edge Signal
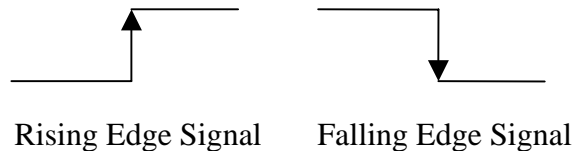
Three different options are available to reset the current count. They are:

• Configured reset value
  When configuring the Totalize function, a value may be specified under the *Counts per Rev* column. When the Totalizer accumulator reaches this value - 1, the accumulator will reset to zero on the next count. Specifying zero for this value allows the Totalizer to count through the full 32-bit range before resetting.

• Ladder control
  Setting registers %Q17-20 reset HSC1-4 (respectively) with no additional configuration. When these registers are asserted, the associated Totalizer accumulator is reset and held at zero (level sensitive). *See also Section 10.6.*

• Direct digital input control (HSC1 and HSC2 only)
  HSC3 (%I11) and HSC4 (%I12) may be configured as hardware digital reset signals for HSC1 and HSC2 respectively. To enable these inputs as reset signals, specify the type as *Totalize Reset* (note that the corresponding Totalize HSC must be previously configured before this option is available). The direct digital reset controls are edge sensitive with the edge polarity configurable.

Maximum direct digital reset latency is 100 μs.

The totalize function also supports an option which compares the current accumulator value with a supplied Preset Value (PV), which is provided through a %AQ, and drives a physical digital output based on the that comparison.

- This option (available for HSC1 and HSC2 only) drives Q1 or Q2 output point (respectively) once the associated Totalizer accumulator reaches (or exceeds) the PV value. To enable this function, the corresponding PWM function output (Q1 or Q2) must be configured for *HSCn Output*.

Note: **Q1 and Q2 are PWM function outputs that may be configured independently as one of the following: standard digital output, PWM, HSCn or stepper output.**

Stepper Function

When the output Q1 is configured for Stepper, the stepper function is enabled at the Q1 output. Only one stepper function and output is available. Q2 is then limited to only a direct digital output, which could be used to signify direction.

The Stepper function requires all five parameters (%AQ registers) to be set for operation. These parameters may be set at whilst the $\boldsymbol{i}^3$ is in run mode but are 'latched' when the stepper function is in operation.

Register	Stepper
%AQ1	Start Frequency
%AQ2	Run Frequency
%AQ3	Accel Count
%AQ4	(32-bit)
%AQ5	Run Count
%AQ6	(32-bit)
%AQ7	Decel Count
%AQ8	(32-bit)
%Q1	Run
%I30	Ready/Done
%I31	Error

- Start Frequency (cycles per second) (%AQ1)
  This value sets the frequency for the first cycle during the acceleration phase and the frequency of the last cycle during the deceleration phase. When an acceleration or deceleration count is specified, the Start Frequency must be greater than 0 and must not exceed the run frequency or an error is generated.

- Run Frequency (cycles per second) (%AQ2)
  This value sets the frequency for the last cycle during the acceleration phase, the consistent frequency during the run phase, and the frequency of the first cycle during the deceleration mode. The Run Frequency must be greater than 0 and must not exceed 5000 cycles/sec. or an error is generated.

- Acceleration Count (%AQ3-4)
  This value sets the number of cycles to occur within the acceleration phase. The frequency of the cycles within this mode will vary linearly between the specified Start and Run frequency. The acceleration count must not equal 1 or an error is generated. Setting this value to zero disables this phase.

- Run Count (%AQ5-6)
  This value sets the number of cycles to occur within the run phase. The frequency of the cycles within this mode is constant at the specified Run frequency. The Run count may be any value. Setting this value to zero disables this phase.

- Deceleration Count (%AQ7-8)
  This value sets the number of cycles to occur within the deceleration phase. The frequency of the cycles within this phase will vary linearly between the specified Run and Stop frequency. The Deceleration count <u>must not equal 1</u> or an error is generated. Setting this value to zero disables this phase.

The stepper function provides two Boolean registers to provide stepper status

- Ready/Done (%I30)
  A high indication on this register indicates the stepper sequence can be started (i.e. not currently busy).

- Error (%I31)
  A high indication on this register indicates that one of the analogue parameters specified above is invalid or the stepper action was aborted before the operation was complete. This register is cleared on the next start command if the cause of error was rectified.

To start the stepper function, one discrete register is required (%Q1). This register must remain set to logic '1' in order to complete the entire cycle. Clearing this register before the cycle is complete aborts the step sequence and sets the error bit.

Note that setting the PLC mode to Stop while the stepper is in operation causes the stepper output to immediately drop to zero and the current stepper count to be lost.

Please note that stepper output level may cause damage or be incompatible with some motor driver inputs. Please always consult the drive documentation to determine if output level of the i^3 and type is compatible.

Programming Examples

For each PWM output function we will program a small example to demonstrate. Except the HSC function as that is an extension of the HSC input function and covered in the HSC Tutorial.

PWM Function Program

In this example we will control a small stepper motor by varying the frequency and the ratio of on time to off time. The user will be able to step through a sequence and enter variables to alter the speed of the stepper.

Please take care when matching the output of the i^3 to the input of the stepper driver. Always read the specifications.

I/O Configuration

First we need to connect the i^3 and configure the PWM output.

Click on the icon [I/O icon] and auto configure the i^3 connected. Then click on the config button next to the orange I/O and setup the Digital Out/PWM.

We are only going to configure the output Q1.

We could enter initial values for the Duty cycle, Pre-scale and Period, however we will leave these at zero and use the associated registers in a ladder program.

%AQ1-2 PWM1 Duty Cycle
%AQ5-6 PWM Pre-scale
%AQ7-8 PWM Period

Ladder Logic Programming

First we need to enter the logic to step through a small sequence of three steps. Step 1 will be initialising/resetting the registers to zero. Step 2 will be moving the first set of PWM parameters in to the registers and Step 3 moves another set of parameters in the registers.

Insert a Normally Open (NO) contact in the 1st rung and assign to %T1. On the same rung assign a positive transition coil, assign to %T2. On the rung below insert a NO contact and assign to %T2, following that insert the function Shift Left (SHL) from the Bitwise operations.
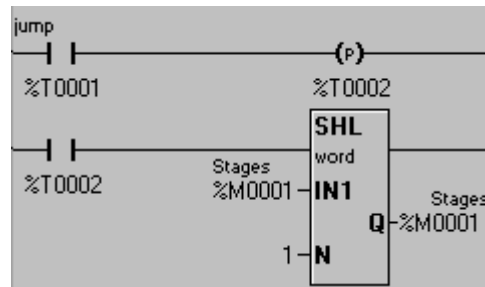
The SHL function logically shifts the data left by the number set in N, contained within a word. Set the functions IN1 and Q to be %M1, with the N set to 1.

0000 0000 0000 0001 shift this register left by one value and we get:
0000 0000 0000 0010, if we shift it again: 0000 0000 0000 0100 and so on.

As we are only going to have 3 steps we need to check that we don't overrun the data by shifting too many times.
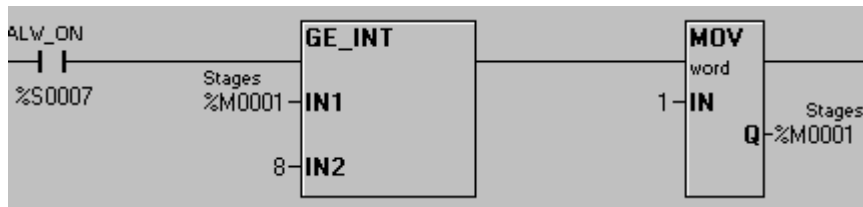
To qualify that we have the correct amount of steps in our sequence we are going to use a Greater than or Equal to (GE) function operating a Move function. If the value is greater than or equal to 8 (binary 1000 = 8) then the move function will move 1 back into %M1.

Now we need to set up the values we want to move into the PWM registers on the given steps. We are going to use the registers %R1 to %R11 for steps 2 and 3, while step 1 will be an initialisation stage.
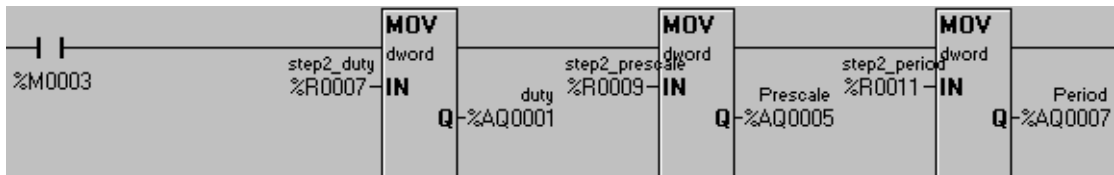
Step 1. Initialise the PWM registers to zero.
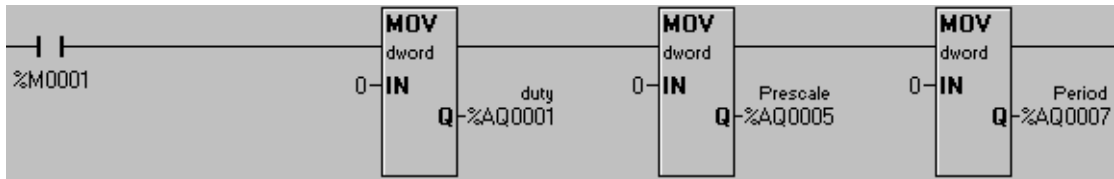
Step 2. Move the values stored in registers %R1, %R3 and %R5 into the PWM registers
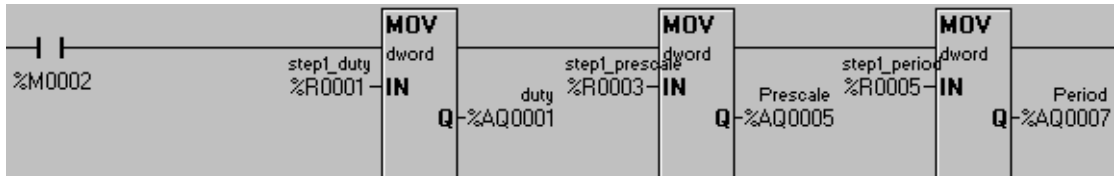
Step 2. Move the values stored in registers %R7, %R9 and %R11 into the PWM registers

Now that the ladder logic has been created we can program some screens for the user interaction.

Screen Editor Programming

Open the screen editor programming by clicking on the icon [T].

On the first screen we need to assign a soft key as a button to step through the sequence and have three numeric displays to show the values of the PWM registers %AQ1, %AQ5 and %AQ7 (all double words). Finally we need to assign a screen change button to move to the second screen where the user can enter values for steps 2 and 3.
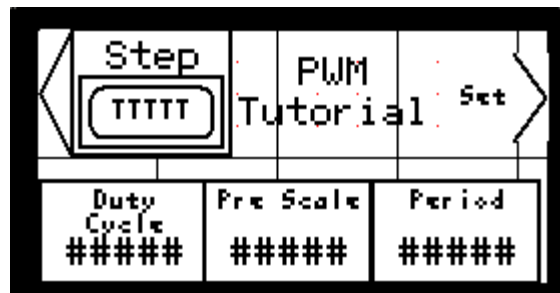
On screen 2 we require six numeric data entry fields all with the Editable feature enabled.

Step 1.
Duty Cycle: %R1   Pre-scale: %R3   Period: %R5
Step 2.
Duty Cycle: %R7   Pre-scale: %R9   Period: %R11
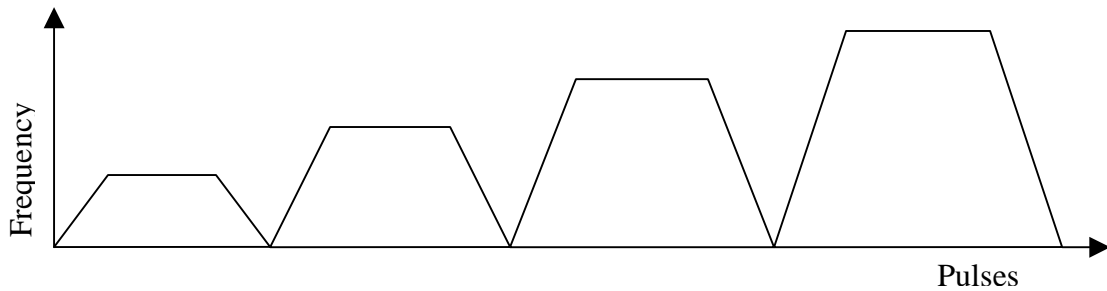

Finally we need to add a screen jump button to go back to the first screen.

Now that the programming is complete we can download this to the i^3. Please see program example: i3_tut_PWM_PWM.csp

Stepper Function

In this example we will run a stepper motor through a sequence of motion by varying the data in the associated registers.

The desired run operation is as follows.

I/O Configuration

As with the previous example we need to configure the I/O. You will notice that when you select Q1 to be in the Stepper function, Q2's options are greyed out.

Ladder Logic Programming

In this program we will have an automatic cycling through the sequence with one start/stop button.

First we have to program the logic to enable the automatic cycling process.
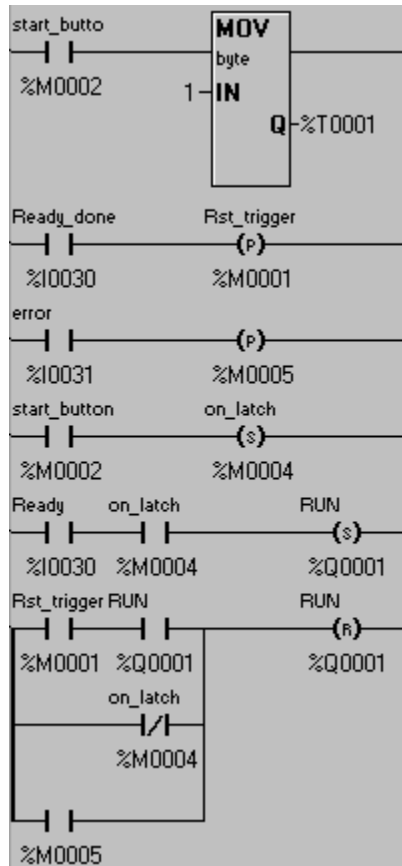

We will configure %M2 as a push button on the screen, which will start the sequence. Once it has been started it cannot be stopped until the sequence has finished. %M2 will set on the bit %M4 to hold the operation on; it will be reset by the output of a counter.

On operating the start button we will move '1' into the register %T1. We will use the %T1 register to rotate a value to cycle through the sequence. The reset pulse (%M1) will be the input to a Rotate function that will rotate the %T registers through %T1, T2, T3 to T4.

Using the ready/done signal from the stepper function (%I30) we will operate a NO positive edge coil (%M1). We will use this coil

To start the cycle (%Q1) the Ready signal (%I30) and the start latch (%M4) need to be on. To retrigger the Run signal and start the stepper on the next step of the cycle we will AND together RUN signal (%Q1) with the reset pulse (%M1).

In case of an error the error signal (%I31) will reset the RUN signal (%I30) and the start latch (%M4).
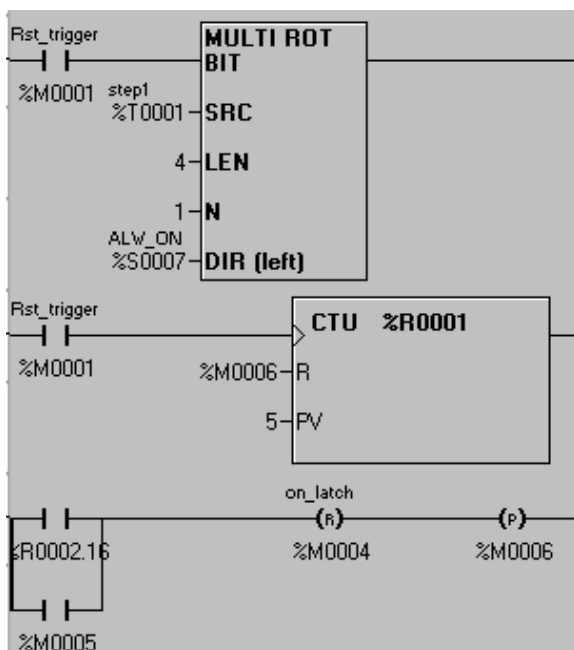
A counter will be used to count the steps and end the cycle.

The output of the counter will reset the on latch (%M4) hence stopping the sequence from restarting and reset the counter.

Now that the sequencing has been programmed we need to move data into the Stepper registers on the given step. On a given step i.e. %T1 we will move five sets of data into the corresponding registers; %AQ1 Start frequency, %AQ2 Run Frequency, %AQ3 & 4 Acceleration count, %AQ5 & 6 Run count and %AQ7 & 8 Deceleration count.
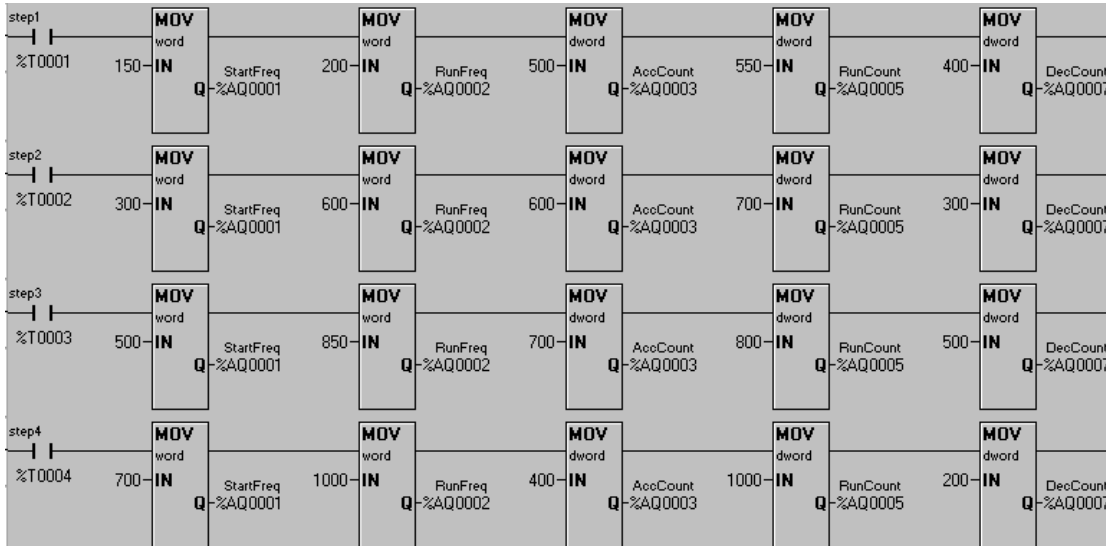

Screen Editor Programming

With the ladder logic programmed we can now edit some user screens. Open the screen editor by clicking on the icon .

We are going to have two screens, the first where the user will start the sequence and be able to see what step they are on and the second that will show a graph of the Acceleration, Run and Deceleration counts against time.

First assign a soft key to a Switch function to %M2 and assign the text indicator properties to %M4. Set the text to be "Running" and "Stopped"
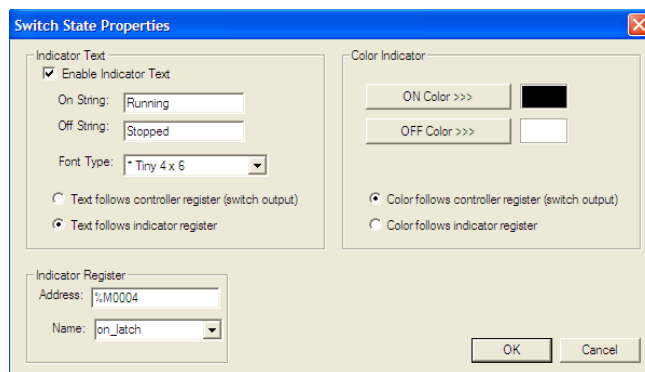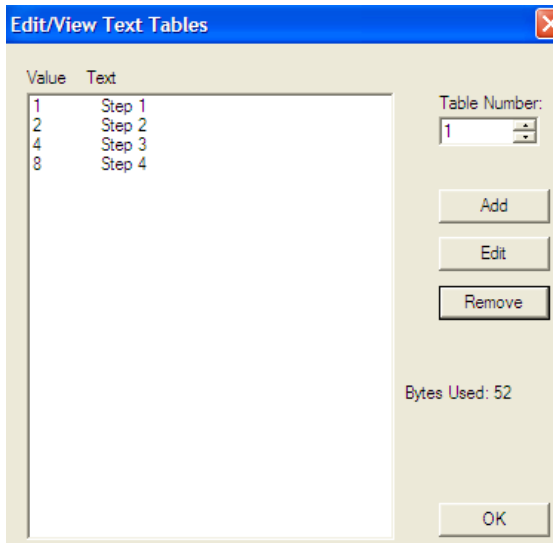
Next enter a Text Table function for the sequence and set up to display the current step for the correct value of the %T1

Enter some static text to tell the user about the system and finally assign a soft key to jump to screen 2, and enable the feature that allows the ESC button to return. The screen should now look similar to the one shown below.
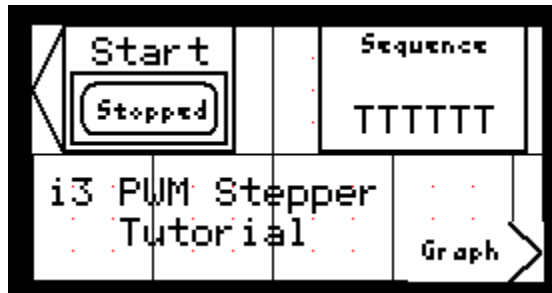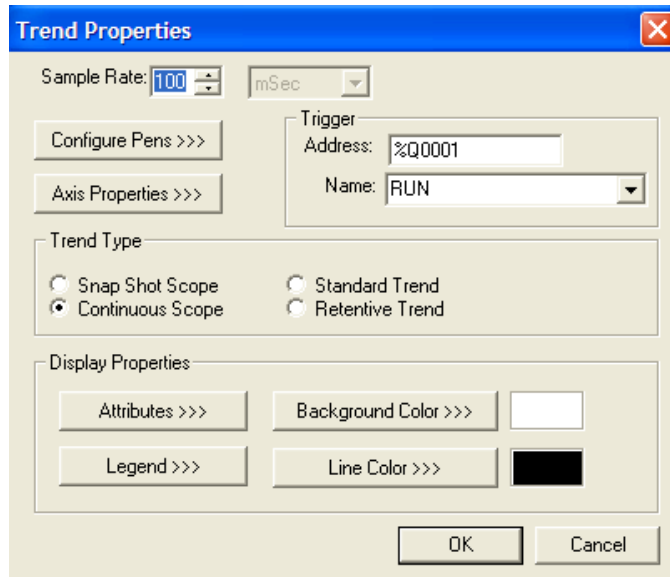
On screen two we are going to have the full screen as a graph. We do not need a screen jump button to return to screen 1 as we have configured this in the previous screen.
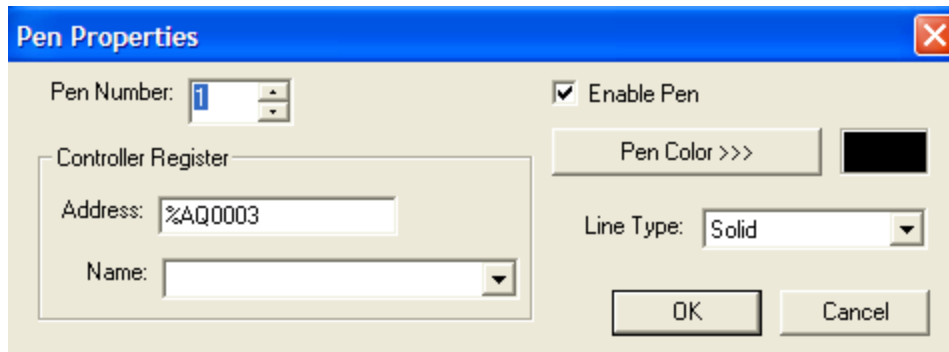
Enter a Trend function and drag it the whole size of the screen.

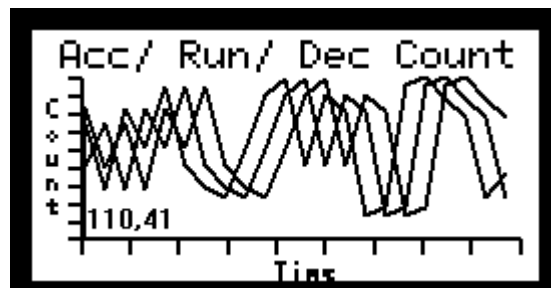Configure the trigger to be the Run signal %Q1.

Set the trend type as continuous.

Configure three pens, one for each count: Acceleration %AQ3, Run %AQ5 and Deceleration %AQ7.

Finally set the legends to display appropriate information relating to the graph. Screen 2 should now look similar to the screen below.

Now that the programming is complete we can download this to the i^3. Please see program example: i3_tut_PWM_stepper.csp

Please take care in the interfacing of the stepper motor. Always read both the i^3 and stepper motor configuration documents.

IMO Precision Controls Limited
1000 North Circular Road
Staples Corner, London
NW2 7JP United Kingdom
Tel: +44 (0)20 8452 6444
Fax: +44 (0)20 8450 2274
Email: imo@imopc.com
Web: www.imopc.com

IMO Jeambrun Automation SAS
165 Rue Jean Jaures,
94700 Maisons Alfort
Paris, France
Tel: +33 (0)1 45 13 47 05
Fax: +33 (0)1 45 13 47 37
Email: info@imopc.fr
Web: www.imopc.fr

IMO Deutschland
Für weitere Einzelheiten
zu IMO Agenten und
Distributoren in Ihrer Nähe
schreiben. Sie bitte ein E-mail
an folgende Adresse:
imo@imopc.com

IMO Italia
Viale A. Volta 127/a
50131 Firenze, Italia
Tel: +39 800 783281
Fax: +39 800 783282
Email: info@imopc.it
Web: www.imopc.it

IMO Canada
Unit 10, Whitmore Road
Woodbridge, Ontario.
L4L 8G4 Cananda
Tel: +1 905 265 9844
Fax: +1 905 265 1749
Email: imocanada@imopc.com

PANEL

Cam Switches
Din Terminals
Drives
Enclosures
Fieldbus remote I/O
Isolators & Switch Fuses
MCB & RCD
Motor Circuit Breakers
Motor Control Gear
Panel Meters
Relays
Signal Conditioning
Sockets
Timers
Transformers & Power Supplies

MACHINE

Drives
Intelligent Terminals/HMI
Limit Switches
Photoelectric Switches
PLCs
Proximity Switches
Temperature Controls

PROCESS

Data Acquisition & Control
Drives
Intelligent Terminals/HMI
Limit Switches
Photoelectric Switches
Proximity Switches
PLCs
Signal Conditioning
Temperature Controls

SAFETY

Lightguards
Safety Limit Switches
Safety Relays

DRIVES

Jaguar VXM 0.37-500kW
Jaguar VXSM 0.37-7.5kW
Jaguar CUB 0.37-2.2kW

All IMO products are tried, tested and approved to relevant international quality standards

PCB EMECH

Audible devices
Chip-on-Board
Device programmers
LEDs & 7 seg. displays
PCB Terminal blocks
Relays - automotive
Relays - power
Relays - signal
Switches