Fuji Electric
e-Front runners

| APPLICATION NOTE | AN-General-0008v101EN |
|---|---|

# Positioning with FRENIC-MEGA Servo and V8 HMI series

| | |
|---|---|
| **Inverter type** | FRENIC-MEGA Servo |
| **Software version** | 1700 and later |
| **Required options** | None |
| **HMI type** | V series (MONITOUCH) |
| **Related documentation** | Macro Reference (1056NE1)<br>FRENIC-MEGA Servo IM (INR-SI47-1545-E) |
| **Author** | Jaume Alonso |
| **Use** | Public, Web |
| **Date** | 25/10/2011 |
| **Version** | 1.0.1 |
| **Languages** | English |

## 1. Introduction.

FRENIC-MEGA Servo is a specific product for servo applications. Normally, in case of servo applications, a positioning is required.

FRENIC-MEGA has two positions selectable by digital input and a third position available by communications. It can be a limitation when the application has more than two positions. In this case, an external device like PLC is needed.

Additionally, in positioning applications, an HMI may be needed for a friendly machine-human interface. HMI gives the possibility to give feedback about variables and to introduce set points.

## 2. Proposed solution.

HMI V series from Fuji Electric have included a Macro function. This function allows the user to implement a code in order to have a basic sequence. In V series the cycle time (period) between two executions of a Macro block can be up to 100 ms. This cycle time makes V series Macros suitable for low reaction process. This cycle time is fast enough for some positioning application, where objective position has to be sent between each positioning and handling. It is understood that positioning and handling takes more than 100 ms.

## 3. Application example.

In order to give a clear idea, an example is proposed and main parts are analyzed. In this case, a simple case of a system with three cyclic positions is used.

### 3.1 Hardware architecture.

Depending on the application, different architectures can be made. In this application note, an example with three target positions will be studied. Figure 1 shows the architecture of a system using three target positions.

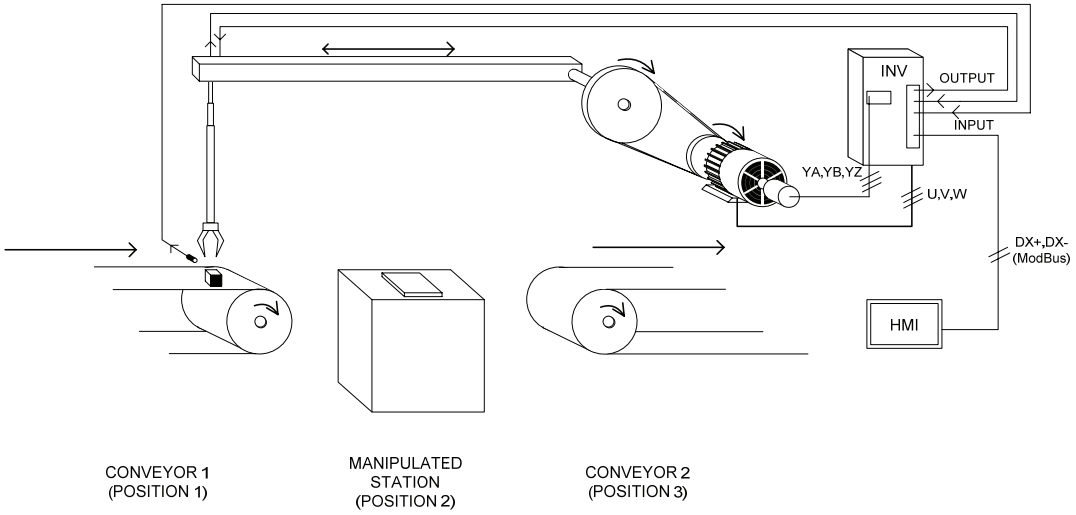



Figure 1. Hardware architecture of a system with three target positions.

NOTE: All process related to servo application like homing (LS sensor) and so on, are not explained in this application note. For additional details refer to IM of FRENIC-MEGA servo.

In this architecture, tasks are divided as follows:

- Inverter
  - Servo control.
  - Hardware for inputs and outputs (U-DI, U-DO functions).
- HMI
  - Position and other variables (speed, ramps, etc) configuration.
  - Feedback of system variables.
  - Sequence controller.

3.2 Wiring (communications).
Communication between HMI and inverter can be done by means of serial communication using Modbus RTU protocol. This protocol is included as standard in both devices. The connection between HMI and the inverter is shown in figure 2.

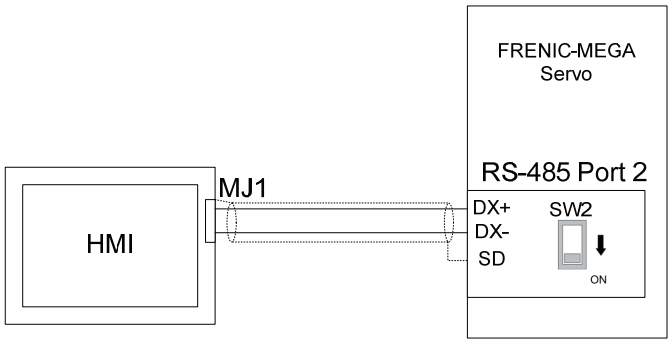
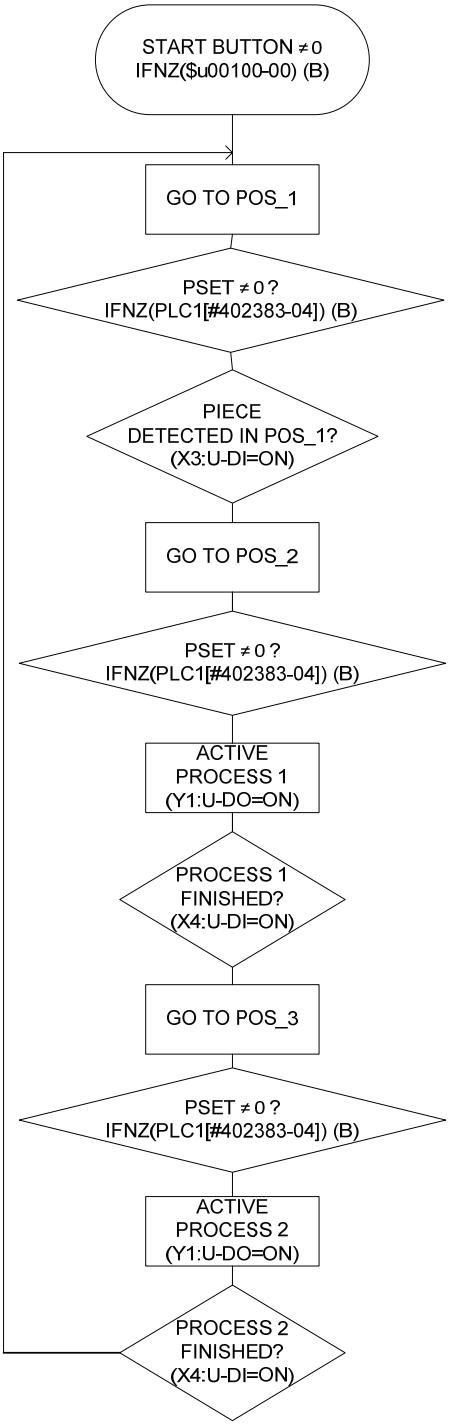



Figure 2. Connection between V8 series and FRENIC-MEGA servo.

In case of FRENIC MEGA inverter, it is recommended to use port 2 for serial communications, thereby port 1 is free to connect the keypad or loader software.

3.3 Software (Macro) flow diagram.
Here, the basic sequence of software is explained as a flow diagram. Minor processes like visualize process variables are not shown.




As a “PROCESS 1” it is understood that robotic arm places a piece in manipulation station, piece is manipulated and it takes again the piece.

As a “PROCESS 2” it is understood that robotic arm places a piece to conveyor 2 (position 3).

3.4 Software code (Macro).

Main software (the one explained in flow diagram) runs in a “Cycle Macro”. The reason is that this macro does not need an event to run. It is periodically running (minim cycle time is 100 ms). A “Cycle Macro” is running while screen associated is shown. Cycle Macro for screen 3 is shown in Figure 3 as an example.

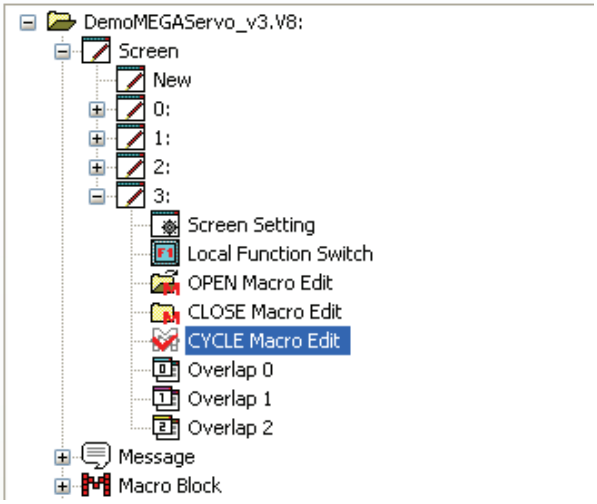



Figure 3. Cycle Macro for screen 3

In this subchapter, main parts of the code will be explained.

1. Feedback current position in a DOUBLE WORD

   Feedback current position is given by two parameters (L72 and L73). L72 are upper four digits, and L73 are lower four digits. In order to work with these two parameters, and in order to monitor this information, it is more convenient to group it in a DOUBLE WORD variable. This double word will be an internal variable (i.v.) of the Macro. In the following piece of code, it is shown how to do it. Comments can be found in the lines of code after a semicolon.

```
$u00200 = PLC1[#402377] (W)    ;save L72 to i.v.200
$u00202 = PLC1[#402378] (W)    ;save L73 to i.v.202
IFNZ($u00200-15) (B)           ;Check if L72 sign is negative
$u00205 = $u00200 & 32767 (W)      ;Save L72 w/o sign to i.v.205
$u00300 = $u00202 (W)          ;Save L73 to i.v.300
$u00207 = $u00205 X 10000 (D) ;Save L72 in i.v.207 upper position
$u00300 = $u00300 + $u00207 (D) ;Save L72 to i.v.300 lower position
$u00300 = $u00300 X -1 (D)     ;Subtract one to feedback position
```

```
ELSE
$u00300 = $u00202 (W)          ;Save L73 to i.v.300
$u00207 = $u00200 X 10000 (D) ;Save L72 in i.v.207 upper position
$u00300 = $u00300 + $u00207 (D) ;Save L72 to i.v.300 lower pos
ENDIF
```

2. Go to POS_1 (STEP 0)

Movement from actual position to POS_1.

```
IF($u00000 == 0) (W)           ;Access to STEP_0
$u04004 = 0 (W)                ;Led in MAIN_POS OFF
$u04005 = 0 (W)                ;Led in POS_1 OFF
$u04006 = 0 (W)                ;Led in POS_2 OFF
$u04007 = 0 (W)                ;Led in POS_3 OFF
PLC1[#402334] = $u01001 (W)    ;Send target position
PLC1[#402306] = $u02001 (W)    ;Send target speed
PLC1[#400008] = $u03001 (W)    ;Send target acceleration
PLC1[#400009] = $u03002 (W)    ;Send target deceleration
PLC1[#401799] = 4H (W)         ;Enable X1 (S-ON)
PLC1[#401799] = CH (W)         ;Enable X2 (START)
$u00000 = 1 (W)                ;STEP 0 finished, go to STEP 1
ENDIF
```

3. PSET≠0 (STEP 1)

Movement set in STEP 0 is finished as soon as PSET is detected. In other words, movement to POS_1 is finished.

```
IF($u00000 == 1) (W)           ;Access to STEP_1
IFNZ(PLC1[#402383-04]) (B)     ;Check if L97(bit) ≠ 0 (PSET=1)
PLC1[#401799] = 4H (W)         ;Disable X2 (START)
$u04005 = 1 (W)                ;Led in POS_1 ON
$u00000 = 2 (W)                ;STEP 1 finished, go to STEP 2
ENDIF
ENDIF
```

4. Active PROCESS 1 (STEP 5)

When position 2 is reached, PROCESS 1 is activated through an inverter output (Output Y5). To finalize PROCESS 1 a signal is received through an inverter input (Input X4).

```
IF($u00000 == 5) (W)                ;Access to STEP_5
PLC1[#401800] = 10H (W)             ;Enable Y5 (UD-O)
IFNZ(PLC1[#403881-05]) (B)          ;Check if X4 ≠ 0 (UD-I)
PLC1[#401800] = 0H (W)              ;Disable Y5 (UD-O)
$u00000 = 6 (W)                     ;STEP 5 finished, go to STEP 6
ENDIF
ENDIF
```

3.5 Inverter settings.

In order to be able to communicate HMI with inverter, some parameters related to Modbus RTU protocol have to be set. Additionally, parameters related to servo application have to be also set according. Specific parameters related to motor control (like motor parameters and gains) are not detailed in this application note. Table 1 shows the inverter basic settings.

Table 1. Inverter basic settings

| Parameter | Description | Setting | Comments |
|---|---|---|---|
| E01 | Terminal [X1] function | 60:S-ON | |
| E02 | Terminal [X2] function | 61: START | |
| E03 | Terminal [X3] function | 25: U-DI | For piece detection in position 1 |
| E04 | Terminal [X4] function | 25: U-DI | For detection of process status |
| E24 | Terminal [Y5] function | 27: U-DO | For process activation |
| H30 | Communications link function (Mode selection) | 8 : RS-485-2 (both) | |
| y97 | Communication Data Storage Selection | 1 | To save parameters on RAM memory |
| L28 | Positioning Data Select (COM) | 1 | To give position via COMM |
| L97(bit0) | PSET output signal ON/OFF | 0 | Depends on the position deviation only |

These settings can be programmed directly on the inverter at power up (of both components) using also Macro Block function.

```
PLC1[#400258] = 60 (W)   ;E01=60 (S-ON)
PLC1[#400259] = 61 (W)   ;E02=61 (START)
PLC1[#400260] = 25 (W)   ;E03=25 (U-DI)
PLC1[#400261] = 25 (W)   ;E04=25 (U-DI)
PLC1[#400281] = 27 (W)   ;E24=27 (U-DO)
PLC1[#401055] = 8 (W)    ;H30=8 (Frequency/Run command by COMM)
```

PLC1[#403682] = 1 (W) ;y97=1 (To save parameters on RAM memory)

PLC1[#402333] = 1 (W) ;L28=1 (To give position via COMM)

PLC1[#402402] = 32 (W) ;L97=100000 (PSET output depends on the position deviation only)

3.6 Screen configuration.
Screen can be configured in different ways in order to give different information and/or possibilities.

For this example two screens have been created. One screen is used to input variables like position, speed and ramps. This screen is shown in figure 4.

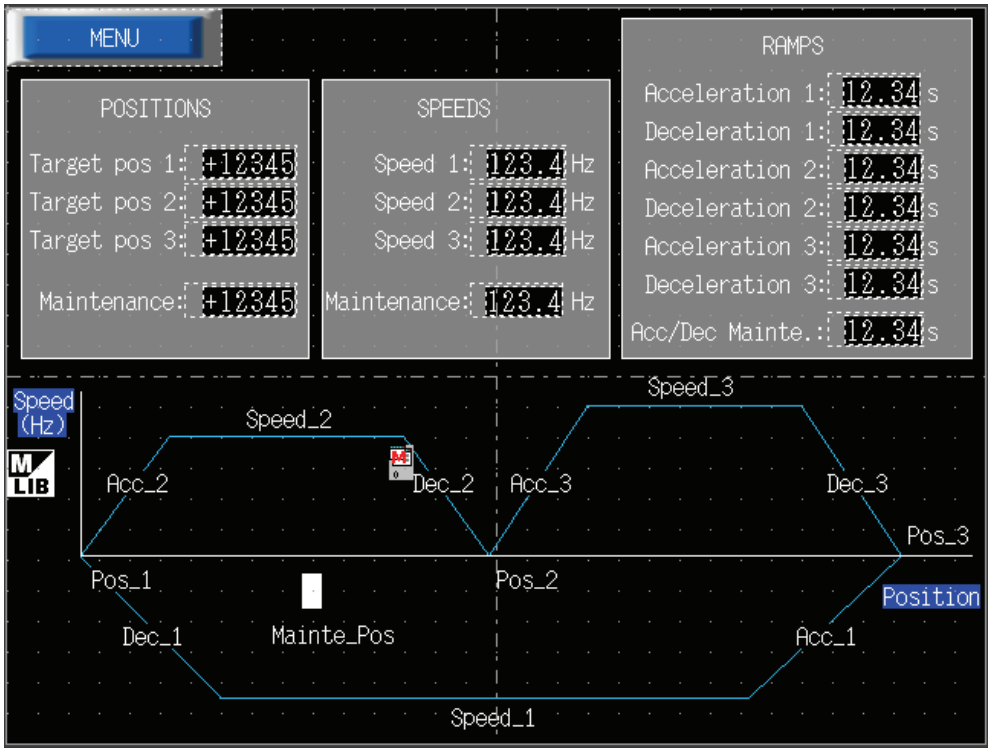


Figure 4. Variables screen

As it can be observed in the figure, process starts in POS_1. Then, actuator it is moved to POS_2 by means of Speed_2 and with Acc_2 and Dec_2 (Acceleration and Deceleration).

The second screen is used to monitor/control the process. This screen is shown in figure 5.

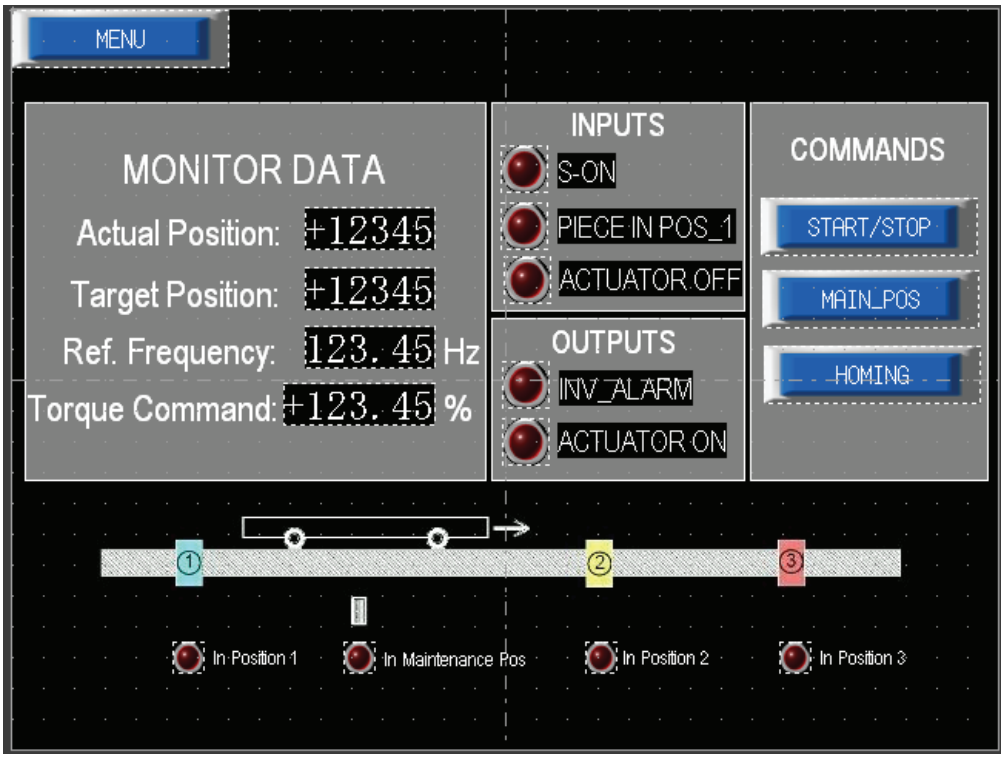



Figure 5. Process monitoring/control screen

## 3. Conclusion

An easy positioning application can be done by means of FRENIC-MEGA Servo and V series HMI. HMI is used as a positioning control; Macro function gives position via serial communications to the inverter. In this case, there is no limitation regarding position indexes. Inverter input/outputs are used to interact between the machine and HMI (positioning control).

How to program a basic positioning sequence with Macros is explained. Positioning sequence is programmed in a cyclic Macro, seeing that code has to run any time that positioning is done.

A very interesting advantage when using a V series HMI with FRENIC series is that, by just informing the type of inverter to be connected, communications parameters are automatically programmed.

## 4. Document history.

| Version | Changes applied | Date | Written | Checked | Approved |
|---|---|---|---|---|---|
| 1.0.0 | First version | 18/10/2011 | J. Alonso | | |
| 1.0.1 | Small text corrections | 25/10/2011 | J. Alonso | D. Bedford | D. Bedford |
| | | | | | |