

AKI Sample Project: Motion Tasking

Overview

This HMI program is set up to run ten different motion tasks in the AKD drive. The operator can edit homing, edit each of the motion tasks, start a homing move, and run each of the ten tasks.

Features:

- Edit Homing
- Select and Edit motion tasks 1 - 10
- See homing status on the run screen
- Start homing
- Select and Run motion tasks 1 - 10

The HMI program has the motion tasks listed as follows:

1. Return to Zero
2. Long Feed
3. Bypass Loader Positions - Move to End
4. Move to Loader Position 1
5. Move to Loader Position 2
6. Move to Loader Position 3
7. Move to Loader Position 4
8. Alternate - Move to Loader Position 1
9. Alternate - Move to Loader Position 2
10. Alternate - Move to Loader Position 3

Requirements:

- HMI program is set up for the AKD to have IP Address 192.168.0.2.
- Must have Motion Tasks 1 - 10 saved in the drive
- Set MODBUS.SCALING = 0 to use drive internal user units

Application Note

KOLLMORGEN

Because Motion Matters™

HMI Screenshots:

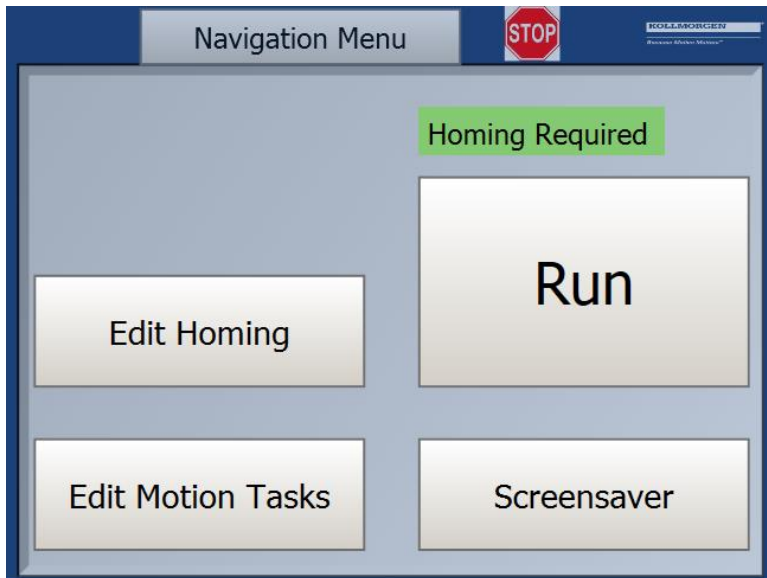


Figure 1: Navigation Menu

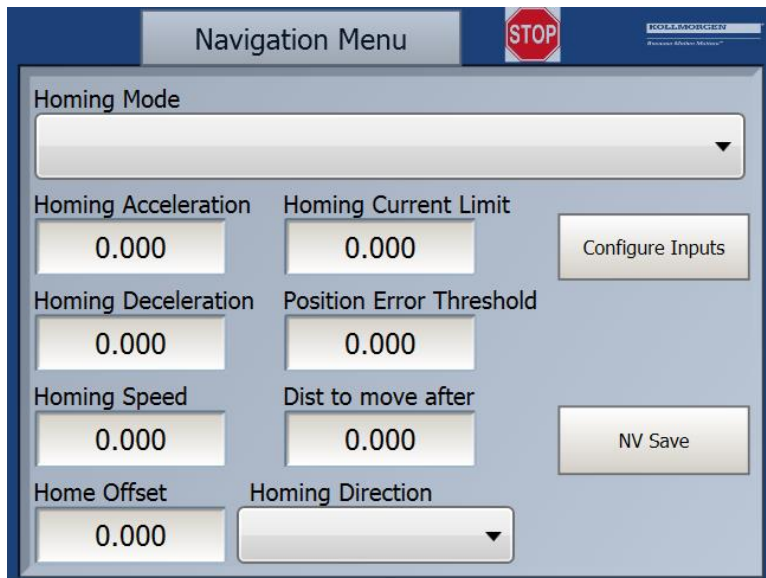


Figure 2: Edit Homing

Application Note

KOLLMORGEN

Because Motion Matters™

Navigation Menu

STOP

KOLLMORGEN
Motion Manager™

▼

Motion Task Type ▼

Target Position 0.000

Target Speed 0.000

Acceleration 0.000

Deceleration 0.000

Set

NV Save

Figure 3: Edit Motion Tasks

Navigation Menu

STOP

KOLLMORGEN
Motion Manager™

▼

Start Homing

✗ Active ✗ Finished

✗ Found ● Error

Start Move

Select Motion Task ▼

✗ Motion Task Moving

✗ MT Target Vel Reached

✗ Target Position Reached

● Invalid Motion Task

● MT Exception

Position 0.000

Velocity 0.000

Current 0.000

Figure 4: Run

Application Note

KOLLMORGEN[®]

Because Motion Matters™

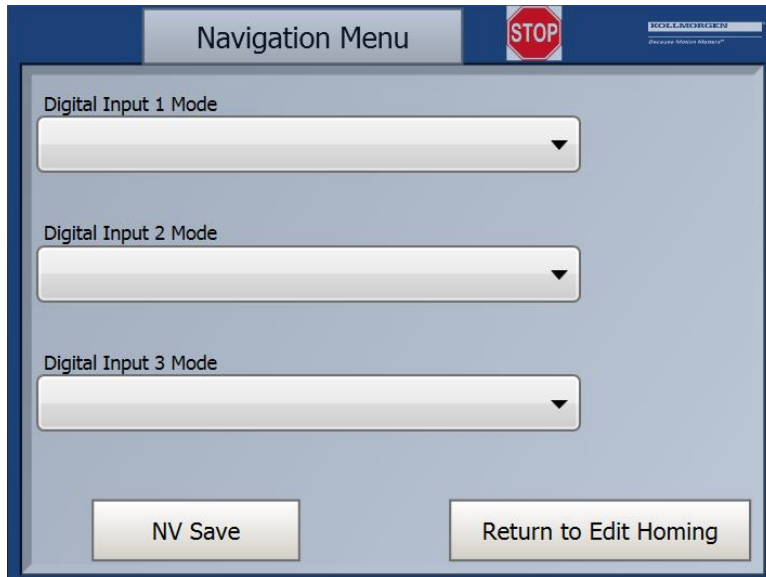


Figure 5: Configure Inputs

Script Programming in C#:

Tag screen:

using System.Threading;

```
public partial class Tags
{
```

```
    //Handles setting the motion task type after a move type is selected
```

```
    //Only supports Absolute, Relative CMD, and Relative Actual
```

```
    void MoveType_ValueChange(System.Object sender,
```

```
Neo.ApplicationFramework.Interfaces.Events.ValueChangedEventArgs e)
```

```
    {
```

```
        if(Globals.Tags.MoveType.Value==0) //if MoveType=0 is "Absolute"
```

```
        {
```

```
            Globals.Tags.MT_CNTL_0.Value=0; //MT.CNTL bit0=0
```

```
            Globals.Tags.MT_CNTL_1.Value=0; //MT.CNTL bit1=0
```

```
            Globals.Tags.MT_CNTL_2.Value=0; //MT.CNTL bit2=0, Absolute
```

```
        }
```

```
        if(Globals.Tags.MoveType.Value==1) //if MoveType=1 is "Relative CMD"
```

```
        {
```

```
            Globals.Tags.MT_CNTL_0.Value=1; //MT.CNTL bit0=1
```

```
            Globals.Tags.MT_CNTL_1.Value=0; //MT.CNTL bit1=0
```

```
            Globals.Tags.MT_CNTL_2.Value=0; //MT.CNTL bit2=0, Relative to
```

Command Position

```
        }
```

```
        if(Globals.Tags.MoveType.Value==2) //if MoveType=0 is "Relative Actual"
```

```
        {
```

```
            Globals.Tags.MT_CNTL_0.Value=1; //MT.CNTL bit0=1
```

```
            Globals.Tags.MT_CNTL_1.Value=1; //MT.CNTL bit1=1
```

```
            Globals.Tags.MT_CNTL_2.Value=1; //MT.CNTL bit2=1, Relative to
```

Feedback Position

```
        }
```

```
    }
```

```
    void MT_LOAD_ValueChange(System.Object sender,
```

```
Neo.ApplicationFramework.Interfaces.Events.ValueChangedEventArgs e)
```

```
    {
```

```
        Thread.Sleep(50);
```

```
        Globals.Tags.MT_LOAD.Value=1;
```

```
        Thread.Sleep(100);
```

```
//if MT.CNTL bit0=0, bit1=0, and bit2=0, then
if((Globals.Tags.MT_CNTL_0.Value==0) &&
(Globals.Tags.MT_CNTL_1.Value==0) && (Globals.Tags.MT_CNTL_2.Value==0))
{
    Globals.Tags.MoveType.Value=0;//set MoveType=0 (Absolute)
}
//if MT.CNTL bit0=1, bit1=0, and bit2=0, then
if((Globals.Tags.MT_CNTL_0.Value==1) &&
(Globals.Tags.MT_CNTL_1.Value==0) && (Globals.Tags.MT_CNTL_2.Value==0))
{
    Globals.Tags.MoveType.Value=1;//set MoveType=1 (Relative CMD)
}
//if MT.CNTL bit0=1, bit1=1, and bit2=1, then
if((Globals.Tags.MT_CNTL_0.Value==1) &&
(Globals.Tags.MT_CNTL_1.Value==1) && (Globals.Tags.MT_CNTL_2.Value==1))
{
    Globals.Tags.MoveType.Value=2;//set MoveType=2 (Relative Actual)
}
}
}
```

Motion Task Screen:

```
public partial class Edit_MT
{
    //Upon making a motion task selection, set MT.NUM and load MT values
    void ComboBox1_SelectionChanged(System.Object sender, System.EventArgs e)
    {
        Globals.Tags.MT_NUM.Value=Globals.Tags.MT_Select.Value;
    }
}
```

Application Note

KOLLMORGEN®

Because Motion Matters™

Run Screen:

```
public partial class Run
{
    //When start button is clicked, set MT.MOVE to value of Movetomake
    void Button2_Click(System.Object sender, System.EventArgs e)
    {
        Globals.Tags.MT_MOVE.Value = Globals.Tags.Movetomake.Value;
    }
}
```