Welcome to Yaskawa America's Training Café Express

- To make this Café enjoyable for all, please follow these tips.
  - Please do not put us on hold. Others will hear the hold music.
  - Do not use a speaker phone. Background noise can be heard.
  - We welcome comments and questions.
    You can type questions into the "Chat" window. Please send to 'All Panelists'
  - Questions not answered during the Café can be e-mailed to training@yaskawa.com or can be entered into the survey sent to you at the end of the class.

YASKAWA

Driving value

MP2000iec: CSV File Transfer to Create XY Interpolated Motion Paths

Greg Findlay
Kevin Hull
Nishant Unnikrishnan

Yaskawa America Incorporated
9-6-2011

YASKAWA

Driving value

File transfer to MP2000iec

Greg Findlay

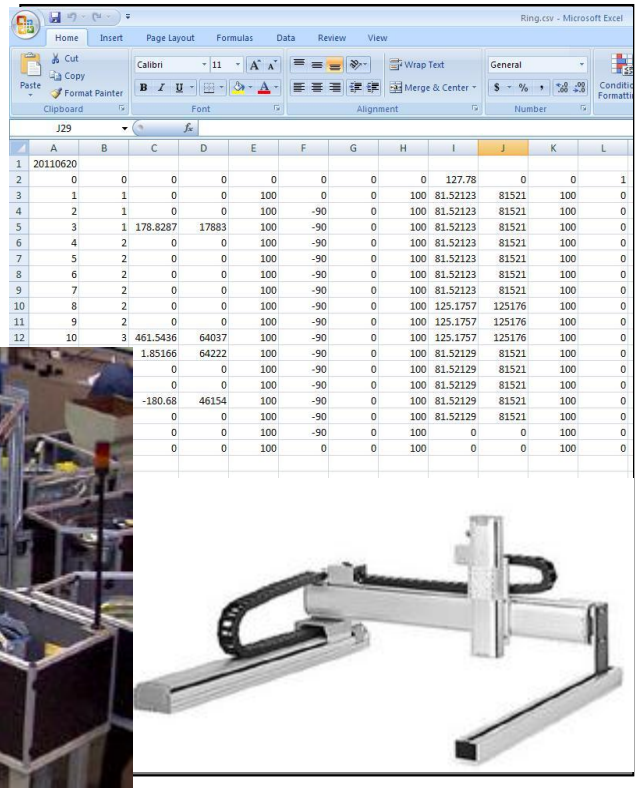
Yaskawa America Incorporated

YASKAWA

Use PC to transfer files to controller

Application Uses

- *Recipes*
- *Part Files*
- *Paths (Gantry Robots)*
- *Machine Setup*

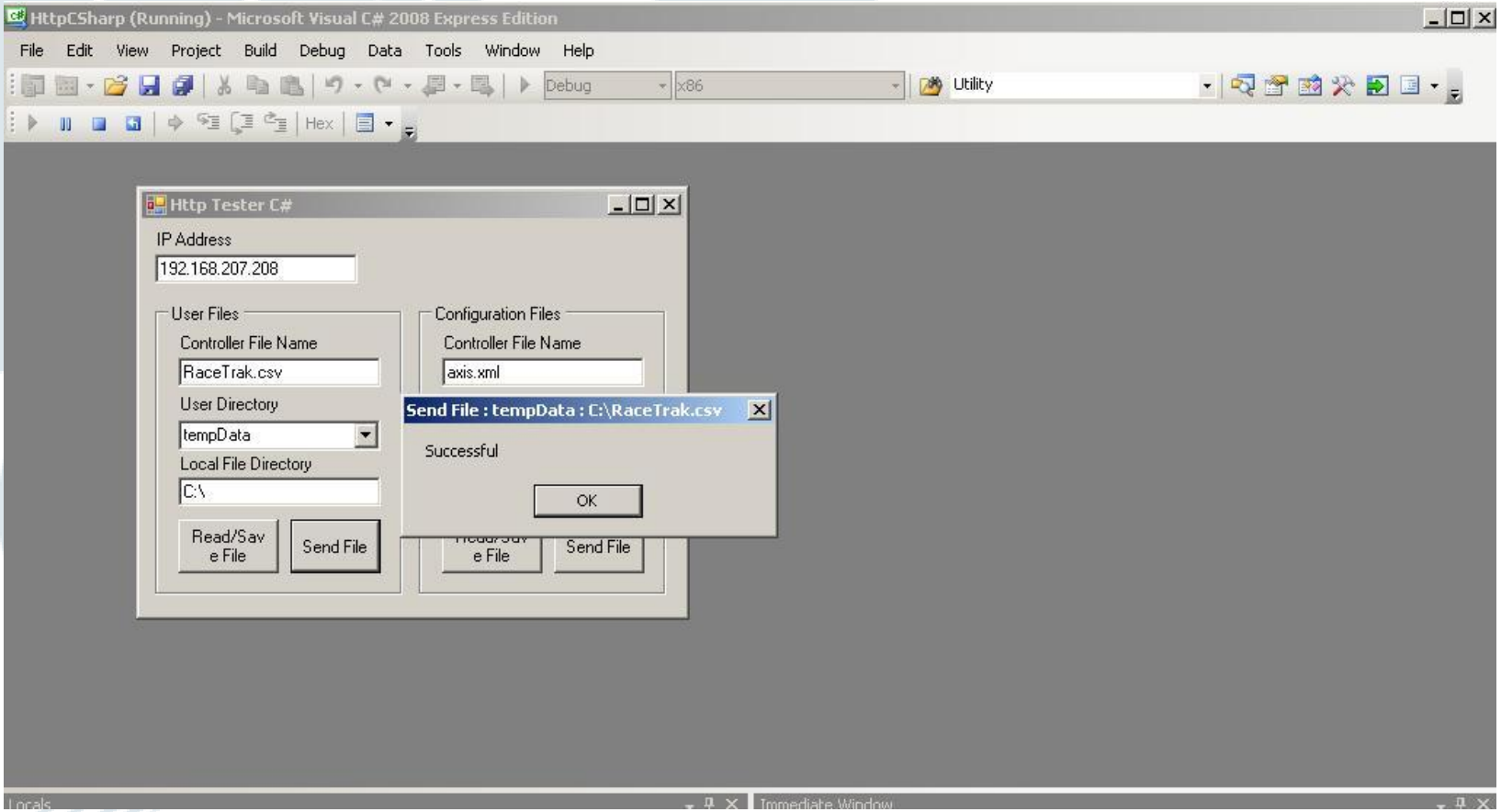


	A	B	C	D	E	F	G	H	I	J	K	L
1	20110620											
2	0	0	0	0	0	0	0	0	127.78	0	0	1
3	1	1	0	0	100	0	0	100	81.52123	81521	100	0
4	2	1	0	0	100	-90	0	100	81.52123	81521	100	0
5	3	1	178.8287	17883	100	-90	0	100	81.52123	81521	100	0
6	4	2	0	0	100	-90	0	100	81.52123	81521	100	0
7	5	2	0	0	100	-90	0	100	81.52123	81521	100	0
8	6	2	0	0	100	-90	0	100	81.52123	81521	100	0
9	7	2	0	0	100	-90	0	100	81.52123	81521	100	0
10	8	2	0	0	100	-90	0	100	125.1757	125176	100	0
11	9	2	0	0	100	-90	0	100	125.1757	125176	100	0
12	10	3	461.5436	64037	100	-90	0	100	125.1757	125176	100	0
			1.85166	64222	100	-90	0	100	81.52129	81521	100	0
			0	0	100	-90	0	100	81.52129	81521	100	0
			0	0	100	-90	0	100	81.52129	81521	100	0
			-180.68	46154	100	-90	0	100	81.52129	81521	100	0
			0	0	100	-90	0	100	81.52129	81521	100	0
			0	0	100	-90	0	100	0	0	100	0
			0	0	100	0	0	100	0	0	100	0

Summary of Procedure

- *Use Ethernet communications to transfer information to the controller*
- *MP2000iec family built in server makes this possible*
- *Use the http protocol, block of data sent with the request*
- *Use .NET HttpWebrequest Class*
- *Use the POST method of HttpWebrequest class*
- *Put your files to ram or flash of the controller*
  - */ramdisk/user/data*
  - */flash/user/data*

Example: C# using HttpWebrequest Class



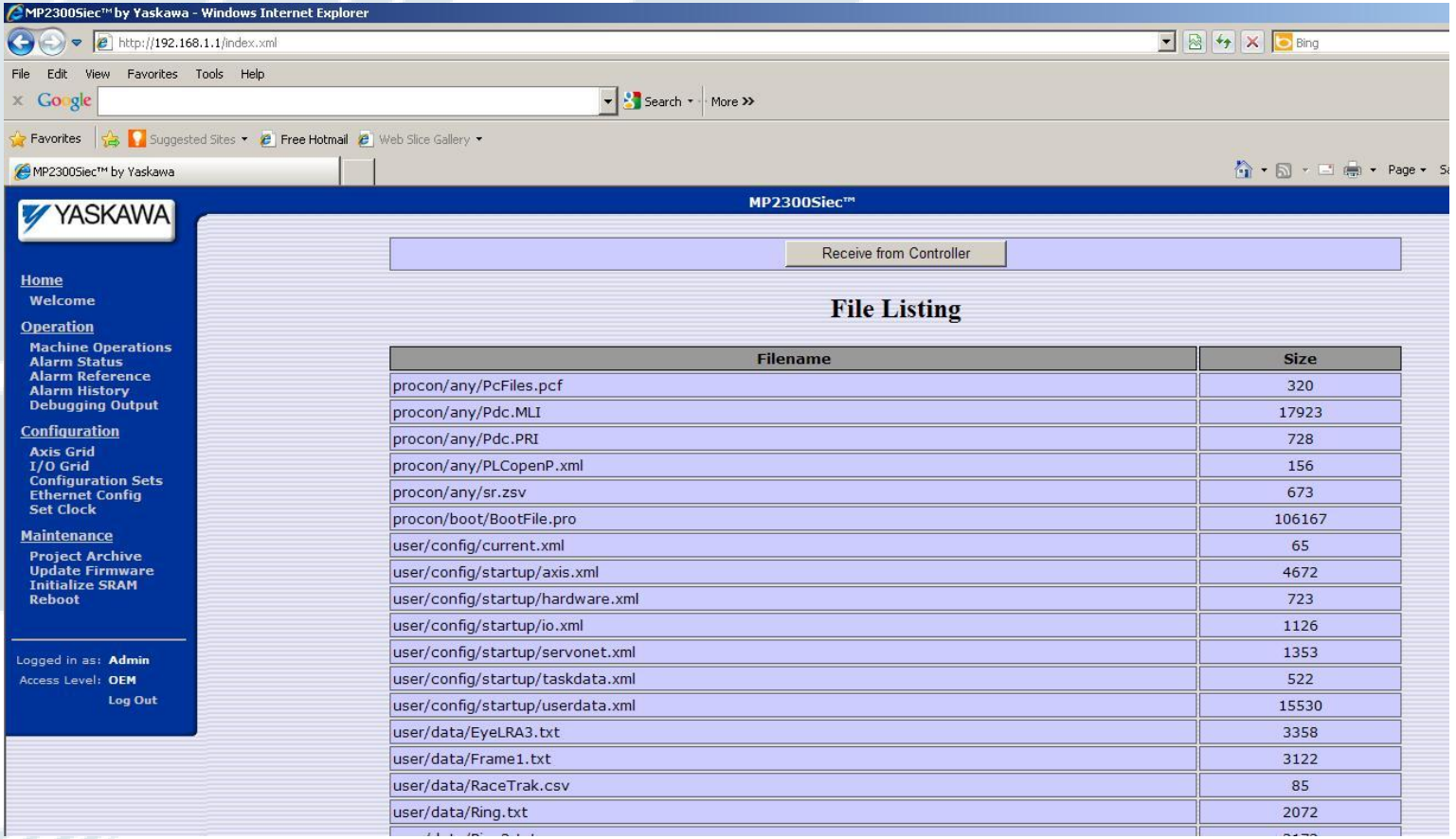
HttpWebRequest CreatePostRequest method

```
public class HttpUtility
{
    public HttpUtility()
    {
        // set connection limit
        ServicePointManager.DefaultConnectionLimit = 10;
        // signal finished port after 5 seconds idle time
        ServicePointManager.MaxServicePointIdleTime = 5000;
        // expect: not default "100 continue"
        ServicePointManager.Expect100Continue = false;
    }

    protected HttpWebRequest CreatePostRequest(string uri)
    {
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(uri);
        request.Method = "POST";
        // remove first 2 dashes of boundary for content type definition
        request.ContentType = "multipart/form-data; boundary=" + boundary_.Substring(2);
        request.Headers.Add("Cache-Control", "no-cache");
        return request;
    }

    protected void UpdateBoundary()
    {
        // create new unique boundary id
        boundary_ = DASHES_ + string.Format("{0:X}", DateTime.Now.Ticks);
    }

    protected string CreateBoundaryPostDataString()
```


HTTP Request summary to transfer file

1. *Initial Line*
    1. *Method: POST*
    2. *Local Path of requested resource /upload/tempdata*
    3. *Version HTTP / 1.1*
2. *Header Lines*
    1. *Content Type: multipart/ form-data; boundary*
    2. *Cache Control; no-cache*
    3. *Host: {MP2000iec IP address}*
    4. *Content Length: in bytes*
    5. *Connection: Keep Alive*
    6. *MIME: multipart/form-data, repeat boundary*
3. *Optional Message*
    1. *Contains byte stream of file you want to send*

View of Project Archieve of file transfer to flash



 |

Driving value

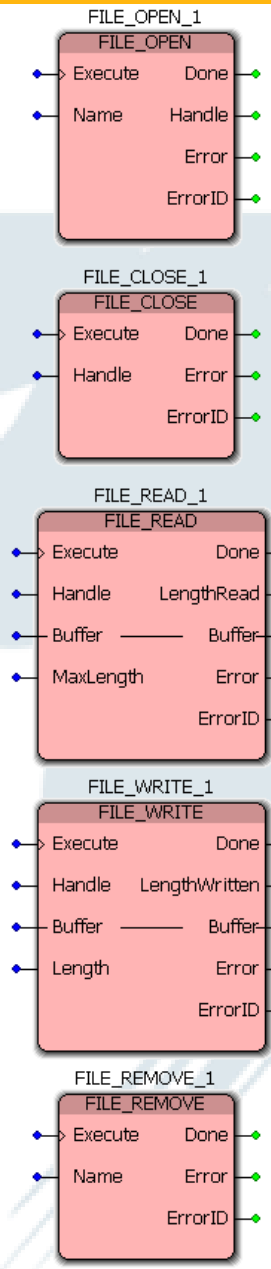
File Read/Write Template

Kevin Hull

Yaskawa America Incorporated

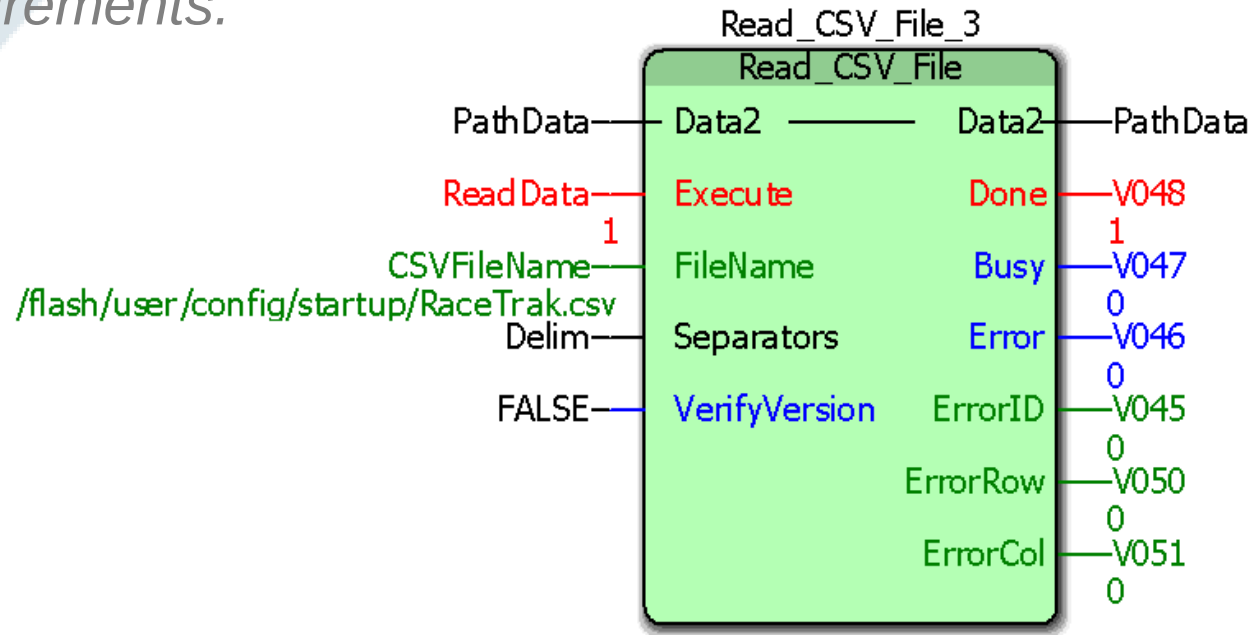
YASKAWA

File Read / Write Template



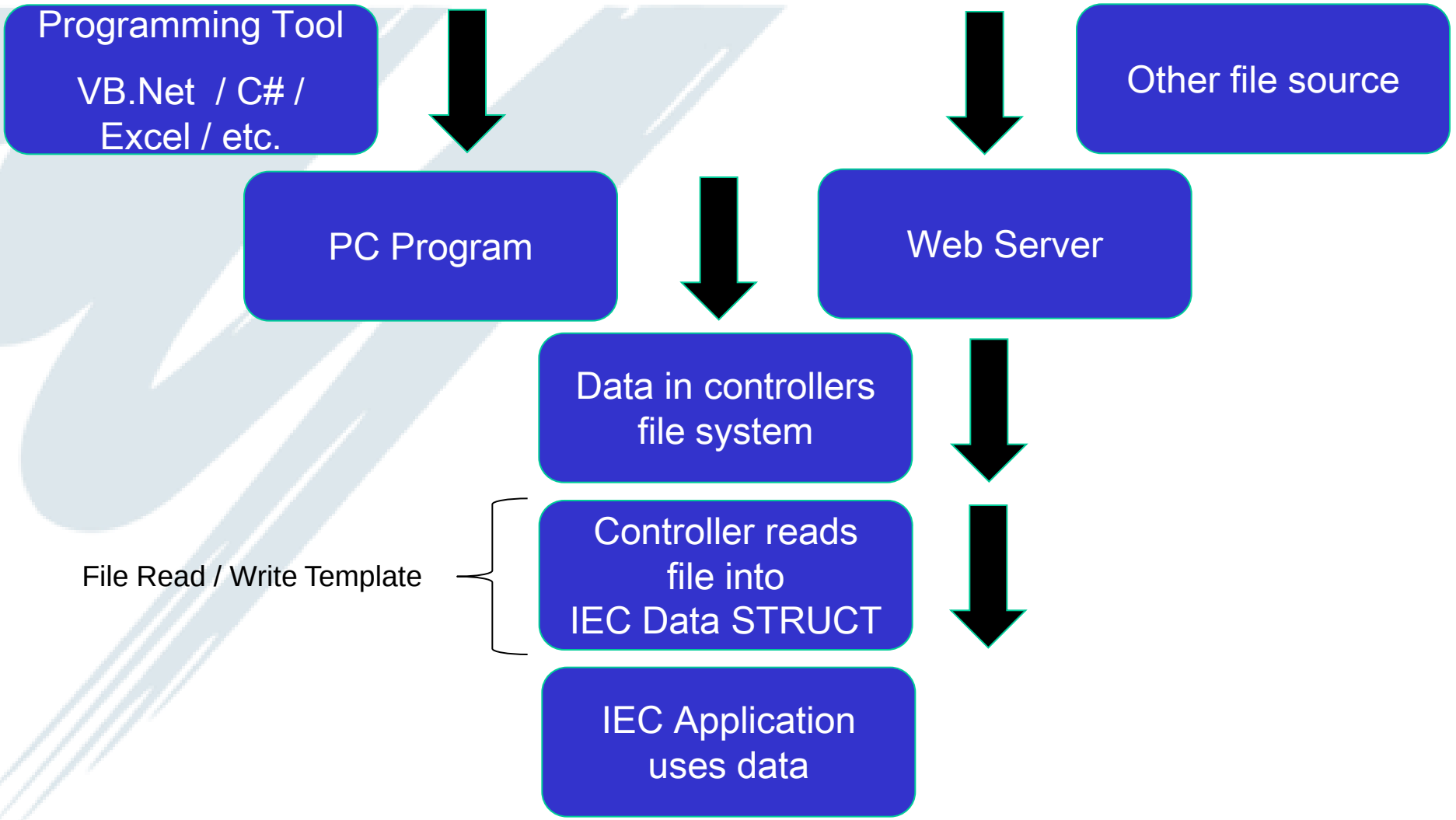
Introducing the File Read / Write Template!

- *Builds upon the 7 functions available in the ProConOS firmware library*
- *A "Template," NOT a "Toolbox" because every application will require customization of the datatypes and file processing based on the specific application requirements.*



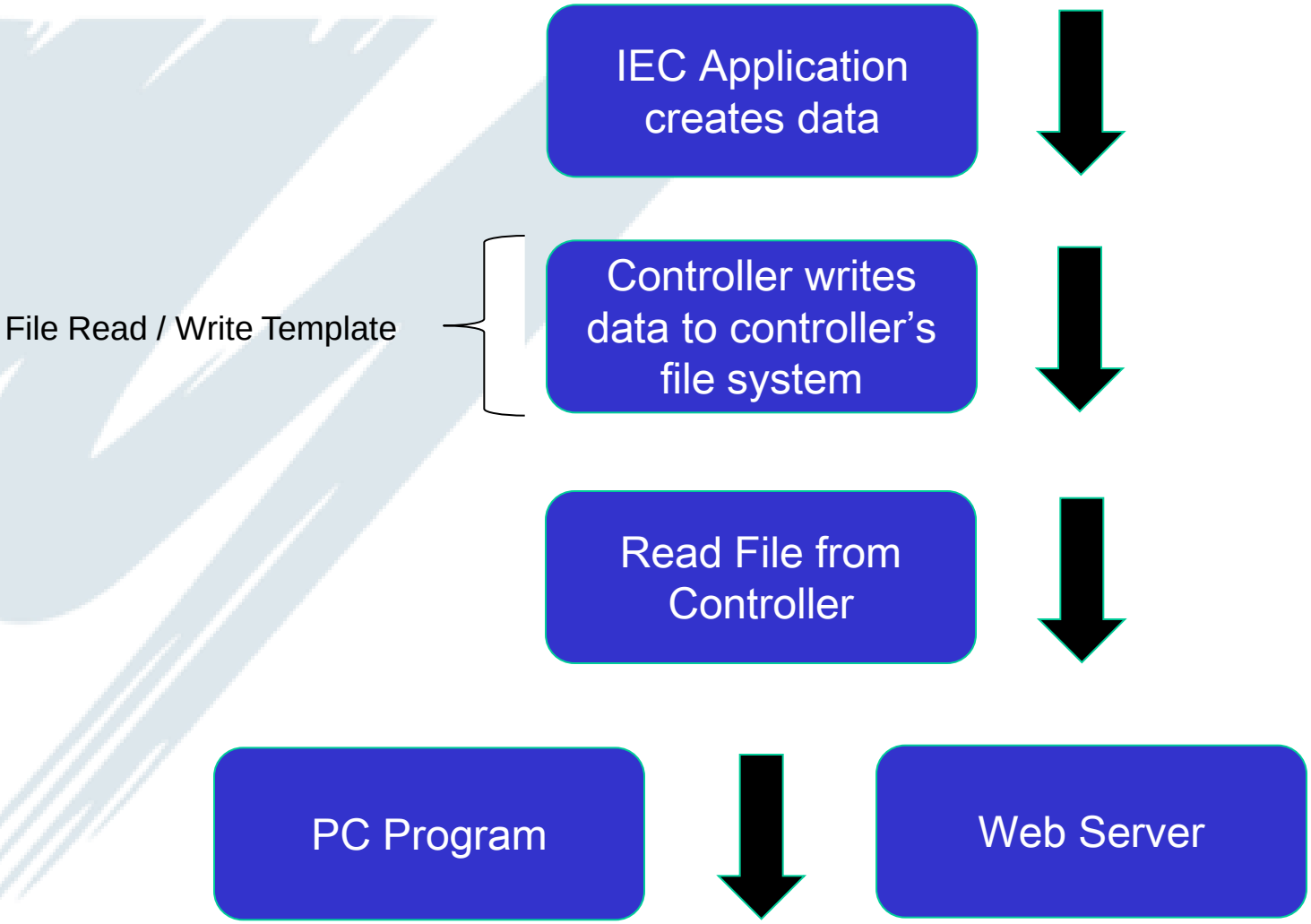
File Read / Write Template

Writing files to the controller



File Read / Write Template

Reading files from the controller



 |

File Read / Write Template

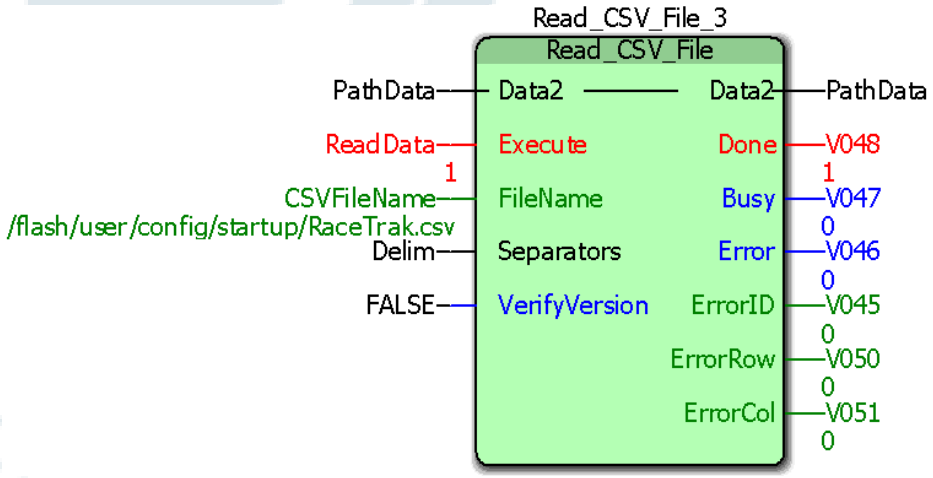
Purpose and Benefits

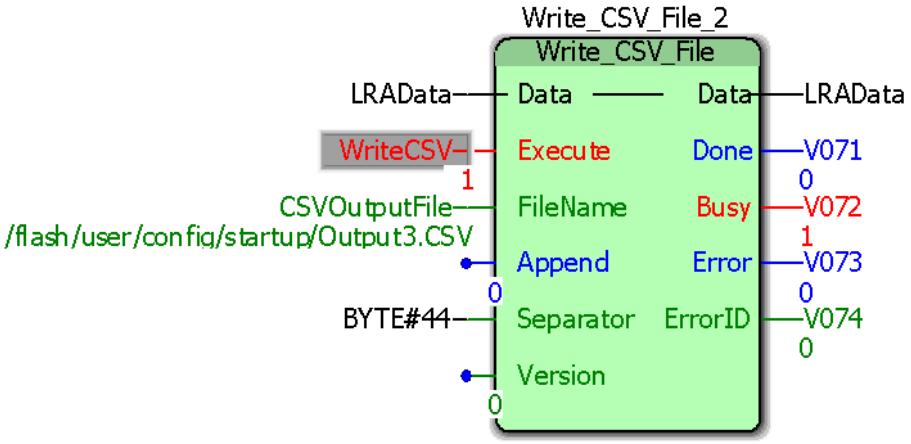
- *To send and retrieve a wide variety of complex recipe data to and from the controller*
- *Alternative to Modbus, Ethernet/IP, and OPC*
  - *Each of these protocols is designed for cyclic data refresh rates, which is usually not required for larger part or recipe definition.*
  - *No need to worry about registers or addresses*
- *Choose between FLASH and RAM for either long term or temporary data requirements.*

File Read / Write Template

Capabilities

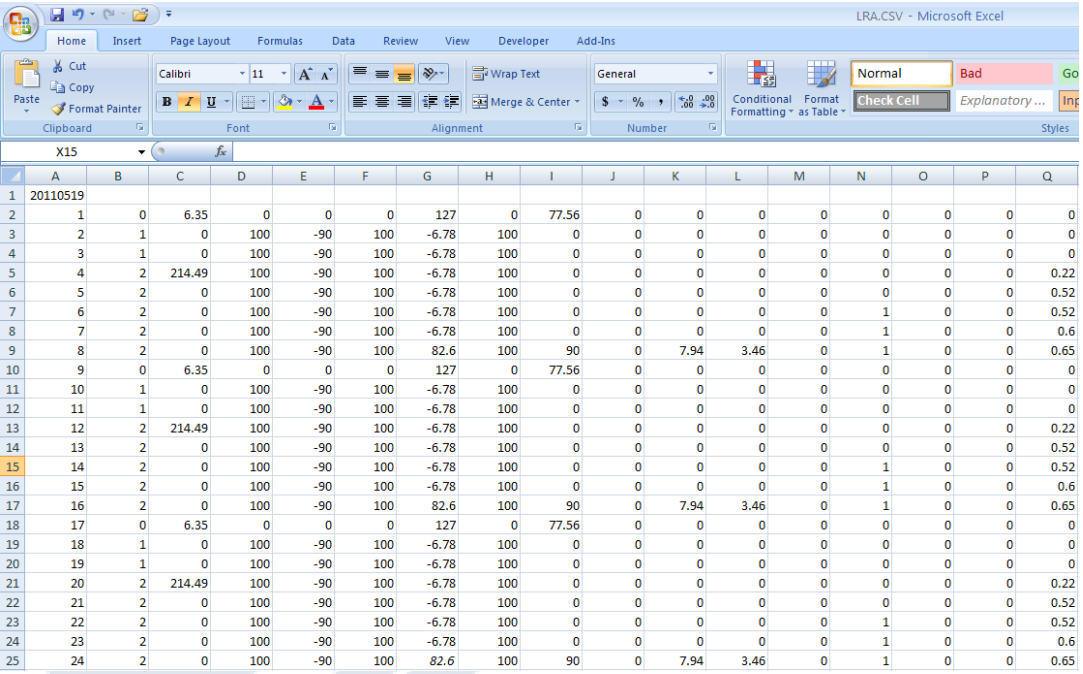
- Binary file reading and writing
- CSV file reading and writing (ASCII characters)
- FLASH or RAM location on controller
- Optional data file version control
- Choice of delimiters for CSV files





File Read / Write Template

CSV - Examples of customizations required for use



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	20110519																
2	1	0	6.35	0	0	0	127	0	77.56	0	0	0	0	0	0	0	0
3	2	1	0	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0
4	3	1	0	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0
5	4	2	214.49	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0.22
6	5	2	0	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0.52
7	6	2	0	100	-90	100	-6.78	100	0	0	0	0	0	1	0	0	0.52
8	7	2	0	100	-90	100	-6.78	100	0	0	0	0	0	1	0	0	0.6
9	8	2	0	100	-90	100	82.6	100	90	0	7.94	3.46	0	1	0	0	0.65
10	9	0	6.35	0	0	0	127	0	77.56	0	0	0	0	0	0	0	0
11	10	1	0	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0
12	11	1	0	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0
13	12	2	214.49	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0.22
14	13	2	0	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0.52
15	14	2	0	100	-90	100	-6.78	100	0	0	0	0	0	1	0	0	0.52
16	15	2	0	100	-90	100	-6.78	100	0	0	0	0	0	1	0	0	0.6
17	16	2	0	100	-90	100	82.6	100	90	0	7.94	3.46	0	1	0	0	0.65
18	17	0	6.35	0	0	0	127	0	77.56	0	0	0	0	0	0	0	0
19	18	1	0	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0
20	19	1	0	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0
21	20	2	214.49	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0.22
22	21	2	0	100	-90	100	-6.78	100	0	0	0	0	0	0	0	0	0.52
23	22	2	0	100	-90	100	-6.78	100	0	0	0	0	0	1	0	0	0.52
24	23	2	0	100	-90	100	-6.78	100	0	0	0	0	0	1	0	0	0.6
25	24	2	0	100	-90	100	82.6	100	90	0	7.94	3.46	0	1	0	0	0.65

```
80                (*********      MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS         **********)
81                (*********      MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS         **********)
82                (*********      MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS         **********)
83
84                (*  Verify that the file version matches one of the formats supported by this function  (ADD MORE COMPARISONS AS NEEDED)    *)
85                IF EQ_STRING(Data.Version, v20110519) THEN
86              0     VersionCode:=INT#2;
87                ELSIF EQ_STRING(Data.Version, v20110620) THEN
88              0     VersionCode:=INT#3;
89                ELSIF EQ_STRING(Data.Version, v20110623) THEN
90              0     VersionCode:=INT#4;
91                END_IF;
92
93                (*********      MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS         **********)
94                (*********      MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS         **********)
95                (*********      MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS         **********)
```

 |

YASKAWA™

CSV - Examples of customizations required for use

```
           3:  (********************        FIle format defined on 20110620          ***************************)

     1         CASE Value OF
                   1:Data.Record[row].Step:=STRING_TO_DINT(ReadCSVLine.Value);         Value:=Value + INT#1;
                   2:Data.Record[row].SEStep:=STRING_TO_DINT(ReadCSVLine.Value);       Value:=Value + INT#1;
                   3:Data.Record[row].Length:=STRING_TO_LREAL(ReadCSVLine.Value);      Value:=Value + INT#1;
                   4:Data.Record[row].LengthABS:=STRING_TO_DINT(ReadCSVLine.Value);    Value:=Value + INT#1;
                   5:Data.Record[row].LSpeed:=STRING_TO_INT(ReadCSVLine.Value);        Value:=Value + INT#1;
                   6:Data.Record[row].Rotation:=STRING_TO_LREAL(ReadCSVLine.Value);    Value:=Value + INT#1;
                   7:Data.Record[row].RotationPOS:=STRING_TO_DINT(ReadCSVLine.Value);  Value:=Value + INT#1;
                   8:Data.Record[row].RSpeed:=STRING_TO_INT(ReadCSVLine.Value);        Value:=Value + INT#1;
                   9:Data.Record[row].BenderAngle:=STRING_TO_LREAL(ReadCSVLine.Value); Value:=Value + INT#1;
                   10:Data.Record[row].BenderPos:=STRING_TO_DINT(ReadCSVLine.Value);   Value:=Value + INT#1;
                   11:Data.Record[row].BSpeed:=STRING_TO_INT(ReadCSVLine.Value);       Value:=Value + INT#1;
                   12:Data.Record[row].MandrelAngle:=STRING_TO_LREAL(ReadCSVLine.Value); Value:=Value + INT#1;
                   13:Data.Record[row].MandrelPos:=STRING_TO_DINT(ReadCSVLine.Value);  Value:=Value + INT#1;
                   14:Data.Record[row].WireAngle:=STRING_TO_LREAL(ReadCSVLine.Value);  Value:=Value + INT#1;
                   15:Data.Record[row].WireRadius:=STRING_TO_LREAL(ReadCSVLine.Value); Value:=Value + INT#1;
                   16:Data.Record[row].WireRadTmp:=STRING_TO_LREAL(ReadCSVLine.Value); Value:=Value + INT#1;
                   17:Data.Record[row].BendRadius:=STRING_TO_LREAL(ReadCSVLine.Value); Value:=Value + INT#1;
                   18:Data.Record[row].SpringBack:=STRING_TO_LREAL(ReadCSVLine.Value); Value:=Value + INT#1;
                   19:Data.Record[row].Flag:=STRING_TO_DINT(ReadCSVLine.Value);        Value:=Value + INT#1;
                   20:Data.Record[row].Pin:=STRING_TO_INT(ReadCSVLine.Value);          Value:=Value + INT#1;
                   21:Data.Record[row].Tool:=STRING_TO_INT(ReadCSVLine.Value);         Value:=Value + INT#1;
                   22:Data.Record[row].ToolUx:=STRING_TO_LREAL(ReadCSVLine.Value);     Value:=Value + INT#1;
                   23:Data.Record[row].EstTime:=STRING_TO_REAL(ReadCSVLine.Value);     Value:=Value + INT#1;
                   24:Data.Record[row].IOOutWord1:=STRING_TO_DINT(ReadCSVLine.Value);  Value:=Value + INT#1;
                   25:Data.Record[row].IOOutWord2:=STRING_TO_DINT(ReadCSVLine.Value);  Value:=Value + INT#1;
                   26:Data.Record[row].IOOutWord3:=STRING_TO_DINT(ReadCSVLine.Value);  Value:=Value + INT#1;
                   27:Data.Record[row].IOOutWord4:=STRING_TO_DINT(ReadCSVLine.Value);  Value:=Value + INT#1;
                   28:Data.Record[row].IOOutWord5:=STRING_TO_DINT(ReadCSVLine.Value);  Value:=Value + INT#1;
                   29:Data.Record[row].IOOutWord6:=STRING_TO_DINT(ReadCSVLine.Value);  Value:=Value + INT#1;
                   30:Data.Record[row].IOOutWord7:=STRING_TO_DINT(ReadCSVLine.Value);  Value:=Value + INT#1;
                   31:Data.Record[row].IOOutWord8:=STRING_TO_DINT(ReadCSVLine.Value);  Value:=Value + INT#1;
                   32:Data.Record[row].IOINWord1:=STRING_TO_DINT(ReadCSVLine.Value);   Value:=Value + INT#1;
                   33:Data.Record[row].IOINWord2:=STRING_TO_DINT(ReadCSVLine.Value);   Value:=Value + INT#1;
                   34:Data.Record[row].IOINWord3:=STRING_TO_DINT(ReadCSVLine.Value);   Value:=Value + INT#1;
                   35:Data.Record[row].IOINWord4:=STRING_TO_DINT(ReadCSVLine.Value);   Value:=Value + INT#1;
                   36:Data.Record[row].IOINWord5:=STRING_TO_DINT(ReadCSVLine.Value);   Value:=Value + INT#1;
                   37:Data.Record[row].IOINWord6:=STRING_TO_DINT(ReadCSVLine.Value);   Value:=Value + INT#1;
                   38:Data.Record[row].IOINWord7:=STRING_TO_DINT(ReadCSVLine.Value);   Value:=Value + INT#1;
                   39:Data.Record[row].IOINWord8:=STRING_TO_DINT(ReadCSVLine.Value);   Value:=Value + INT#1;
                   40:Data.Record[row].Dwell:=STRING_TO_INT(ReadCSVLine.Value);        Value:=Value + INT#1;
                   41:Data.Record[row].ProgNum:=STRING_TO_INT(ReadCSVLine.Value);      Value:=Value + INT#1;
                   42:Data.Record[row].LoopStart:=STRING_TO_INT(ReadCSVLine.Value);    Value:=Value + INT#1;
                   43:Data.Record[row].LoopEnd:=STRING_TO_INT(ReadCSVLine.Value);      Value:=Value + INT#1;
                   44:Data.Record[row].ProgCall:=STRING_TO_INT(ReadCSVLine.Value);     (*  last one handled below  *)
               END_CASE;

 FALSE         IF ReadCSVLine.EOL THEN      (*  The end of the row was detected  *)
  TRUE             ReadValueEnable:=FALSE;  (*  To allow the ReadLine function to run once more with Enable=FALSE  *)
     1             IF Value=INT#44 THEN     (*  Should be EOL event only for last column if data is formatted properly  *)
     4                 Row:=Row + INT#1;
     0                 Data.Records:=INT_TO_UINT(Row);
     1                 Value:=INT#1;
                   ELSE
 FALSE                 RowError:=TRUE;
                   END_IF;
               END_IF;
```

 |

File Read / Write Template

YASKAWA

Example of binary file output from controller

IEC Side

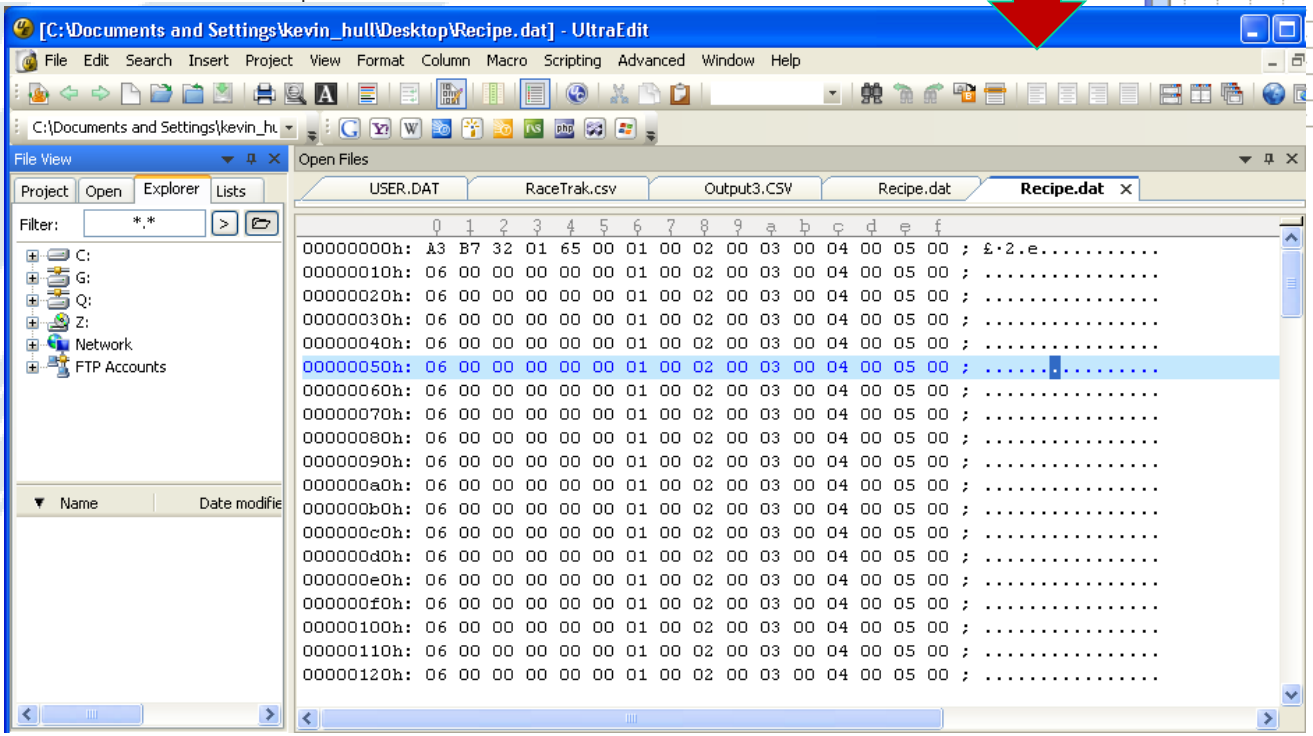
PC Side

```
1
2
3       /flash/user/config/startup/recipe.dat
4               /ramdisk/user/data/recipe.dat
5
6
7                                       20101027
8
9
10                                          101
11
12
13
14
15
16
17
18
19                                          101
```

```
(*********      Examples for binary data files      ***********)
    BinaryFileName:='/flash/user/config/startup/recipe.dat';
    BinaryFileName2:='/ramdisk/user/data/recipe.dat';


    MyData.Version:=DataSpec;

    (*  Test data  *)
    FOR x:=UINT#0 TO UINT#100 BY UINT#1 DO
        MyData.Record[x].Item1:=UINT#16#0001;
        MyData.Record[x].Item2:=UINT#16#0002;
        MyData.Record[x].Item3:=UINT#16#0003;
        MyData.Record[x].Item4:=UINT#16#0004;
        MyData.Record[x].Item5:=UINT#16#0005;
        MyData.Record[x].Item6:=UINT#16#0006;
    END_FOR;

    MyData.Records:=x;
```

Watch Window

Variable	Value
MyData	
Version	20101027
Record	
[0]	
Item1	1
Item2	2
Item3	3
Item4	4
Item5	5
Item6	6
Item7	0
Item8	0
[1]	
Item1	1
Item2	2
Item3	3
Item4	4
Item5	5
Item6	6
Item7	0
Item8	0



File Read / Write Template

Summary

- *In development and experimentation stage since December 2010*
- *File_RW Template used in one application to date*
  - *ProConOS file handling FBs used in two other applications before template was created.*
- *Looking for applications to further test the capabilities and get feedback*

Driving value

PathGenerator, MovePath & CalculateAngles with MP2000iec

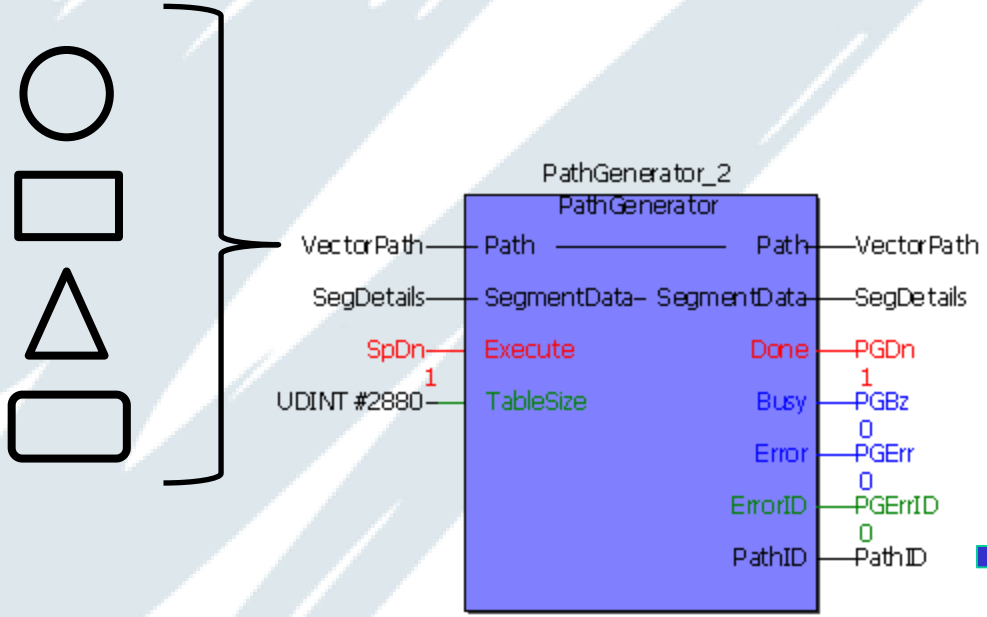
Gantry_Toolbox_v008 & v200

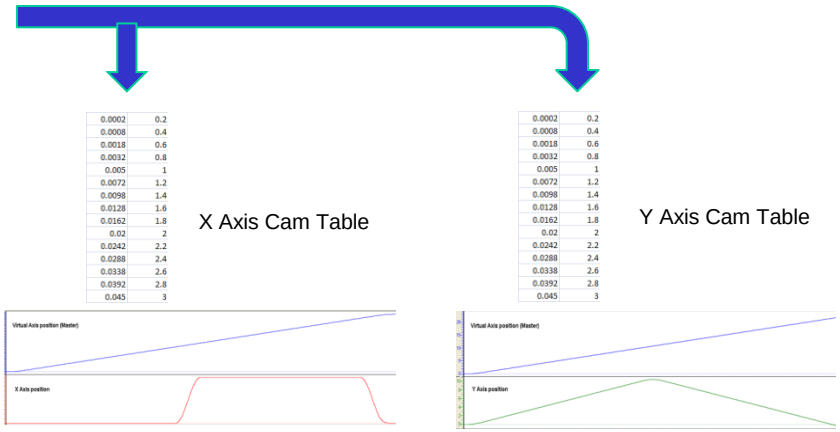
MotionWorks IEC Pro

YASKAWA

PathGenerator

Converts user defined straight line and arc segment data into cam files which can be used to produce XY coordinated motion. The cam files are loaded into the motion engine ready for use

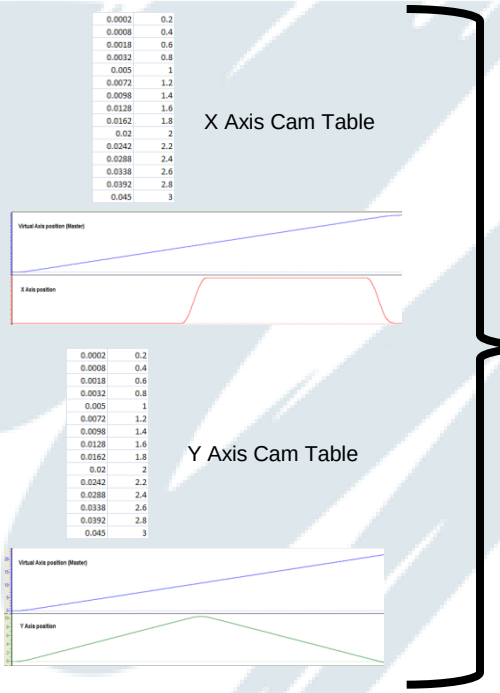


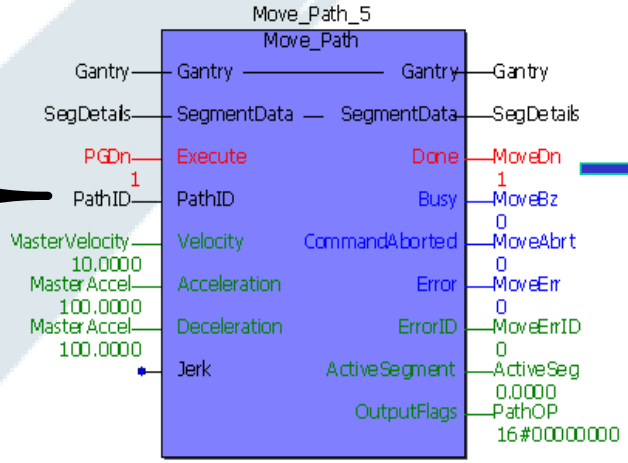


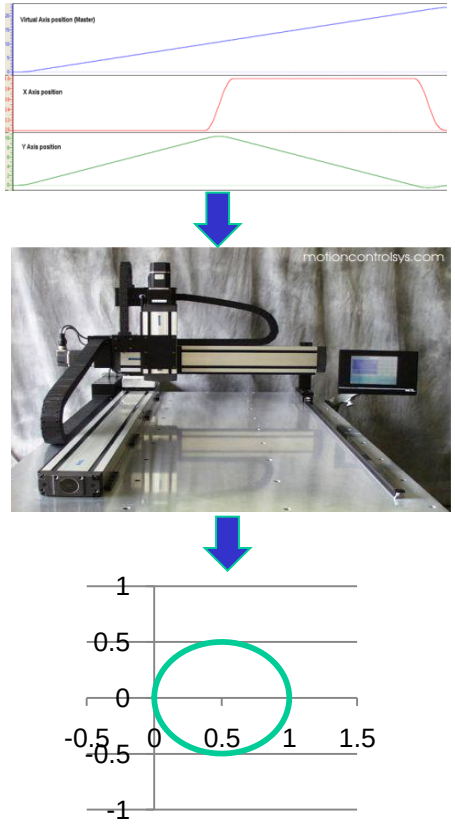
 |

MovePath

Uses the profile described by the 'PathStruct' data type and commands motion to the X, Y axes using a virtual axis as the master

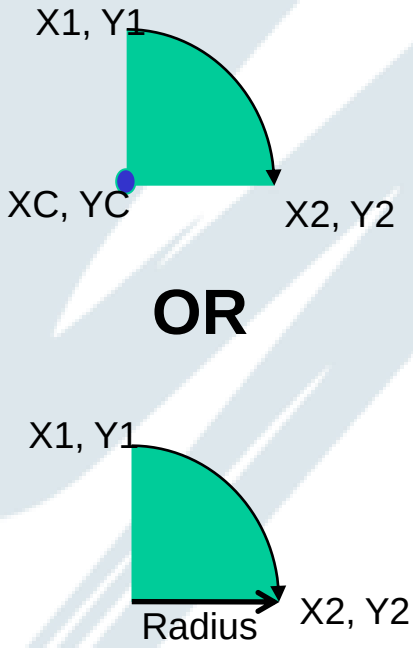


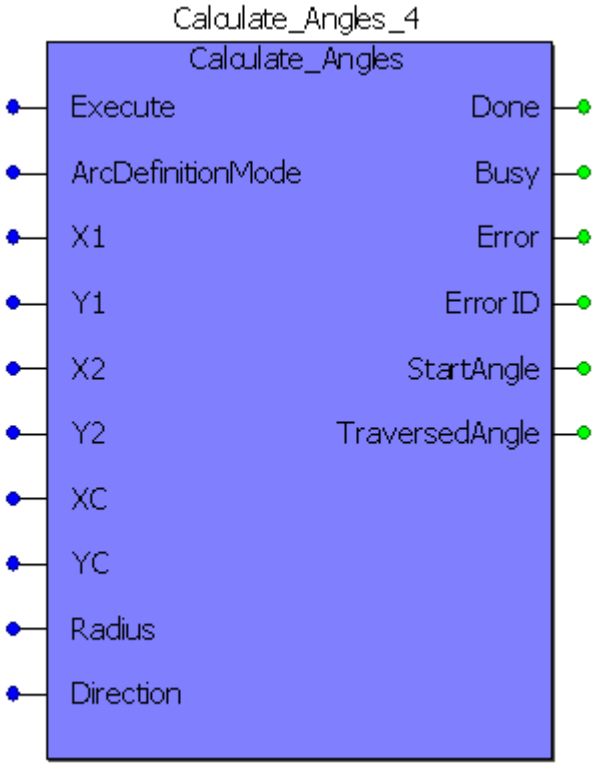


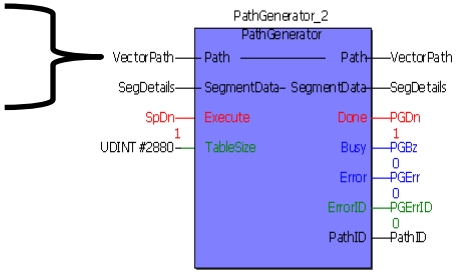


CalculateAngles

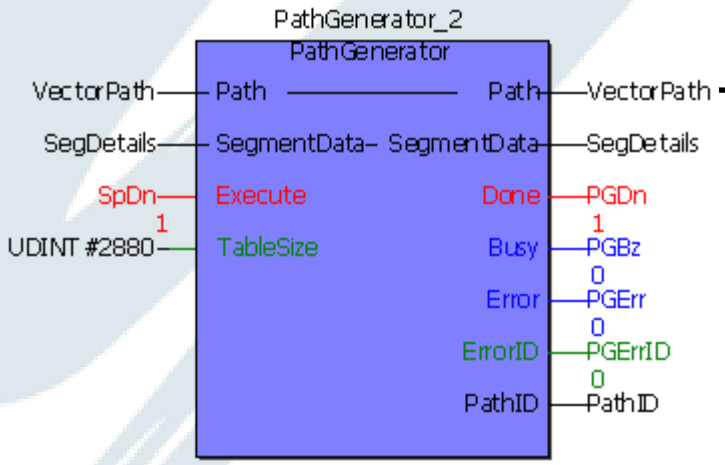
Uses either a) two co-ordinates and center point of an arc or b) two co-ordinates and radius of an arc to calculate start and traversed angles required for PathStruct data type in the PathGenerator function block







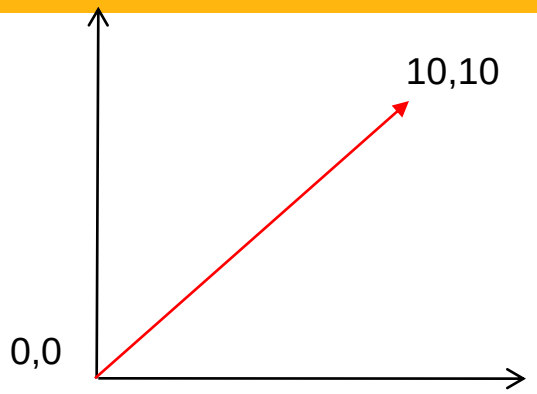
Path Generator Inputs




```
PathDetail:STRUCT
    SegmentType:INT;
    XCoord:LREAL;
    YCoord:LREAL;
    Radius:LREAL;
    StartAngle:LREAL;
    TraversedAngle:LREAL;
    Resolution:REAL;
    OutputFlags:DWORD;
    MasterEnd:LREAL;
END_STRUCT;

PathPointArray: ARRAY[0..100] OF PathDetail;

PathStruct: STRUCT
    Data:PathPointArray;
    Segments:INT;
END_STRUCT;


(*  ENUM Type for PathDetail's SegmentType  *)
TB_PatternType:
(
    na,
    StraightLine,
    Arc
```

 |

Straight Line Path Example



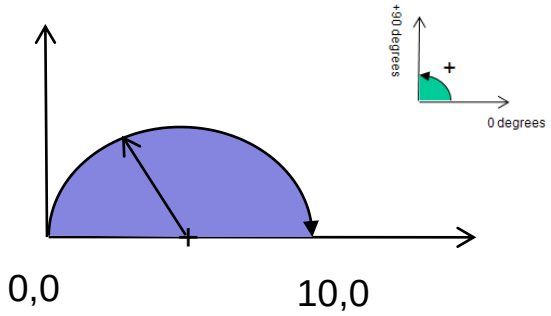

```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#10.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
```

Gantry Datatypes

```
PathDetail:STRUCT
    SegmentType:INT;
    XCoord:LREAL;
    YCoord:LREAL;
    Radius:LREAL;
    StartAngle:LREAL;
    TraversedAngle:LREAL;
    Resolution:REAL;
    OutputFlags:DWORD;
    MasterEnd:LREAL;
END_STRUCT;

PathPointArray: ARRAY[0..100] OF PathDetail;

PathStruct: STRUCT
    Data:PathPointArray;
    Segments:INT;
END_STRUCT;


(*  ENUM Type for PathDetail's SegmentType  *)
TB_PatternType:
(
    na,
    StraightLine,
    Arc
```

 |

Arc Path Example

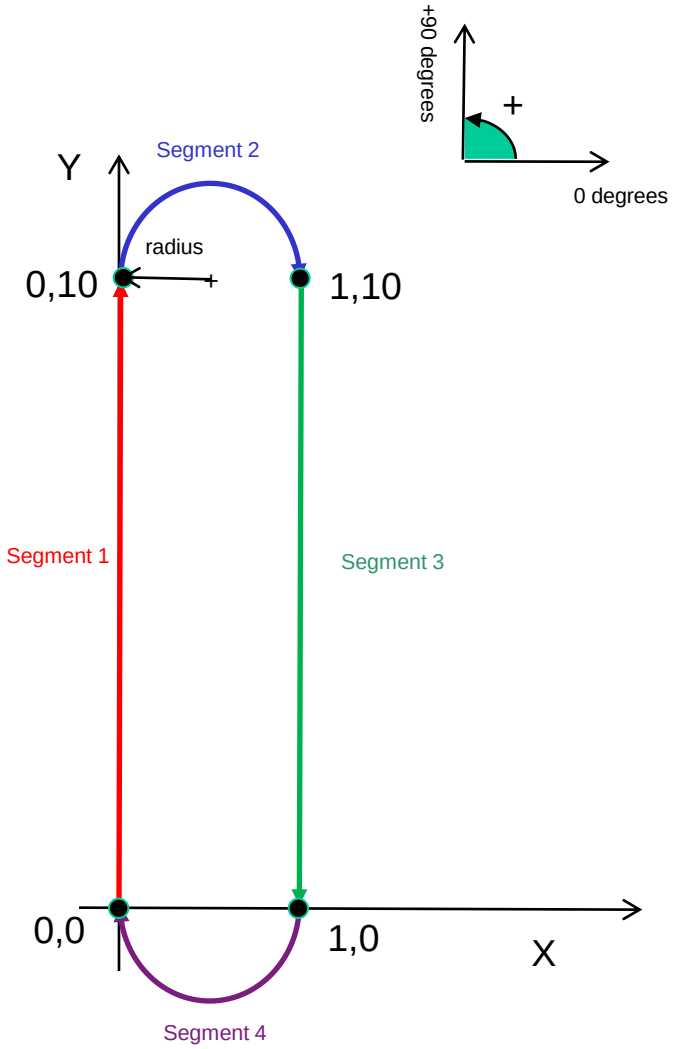



```
VectorPath.Data[2].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[2].Radius:=LREAL#5.0;
VectorPath.Data[2].StartAngle:=LREAL#180.0;
VectorPath.Data[2].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[2].Resolution:=REAL#0.05;
VectorPath.Data[2].OutputFlags:= DWORD#2
```

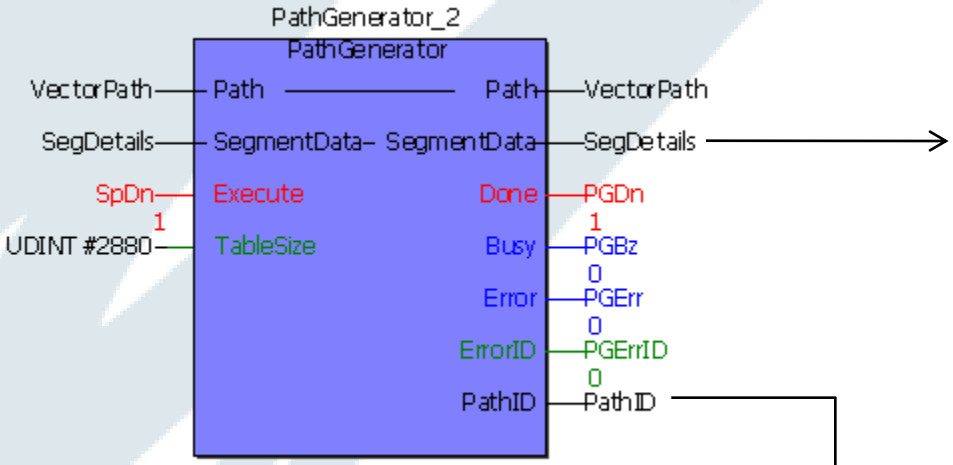
```
PathDetail:STRUCT
    SegmentType:INT;
    XCoord:LREAL;
    YCoord:LREAL;
    Radius:LREAL;
    StartAngle:LREAL;
    TraversedAngle:LREAL;
    Resolution:REAL;
    OutputFlags:DWORD;
    MasterEnd:LREAL;
END_STRUCT;

PathPointArray: ARRAY[0..100] OF PathDetail;

PathStruct: STRUCT
    Data:PathPointArray;
    Segments:INT;
END_STRUCT;


(*  ENUM Type for PathDetail's SegmentType  *)
TB_PatternType:
(
    na,
    StraightLine,
    Arc
```

Complex Path Example

```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#0.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
VectorPath.Data[1].OutputFlags:=DWORD#1;

VectorPath.Data[2].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[2].Radius:=LREAL#0.5;
VectorPath.Data[2].StartAngle:=LREAL#180.0;
VectorPath.Data[2].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[2].Resolution:=REAL#0.05;

VectorPath.Data[3].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[3].XCoord:=LREAL#1.0;
VectorPath.Data[3].YCoord:=LREAL#0.0;
VectorPath.Data[3].OutputFlags:=DWORD#2;

VectorPath.Data[4].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[4].Radius:=LREAL#0.5;
VectorPath.Data[4].StartAngle:=LREAL#0.0;
VectorPath.Data[4].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[4].Resolution:=REAL#0.05;

VectorPath.Segments := INT#4;
```



Path Generator Outputs



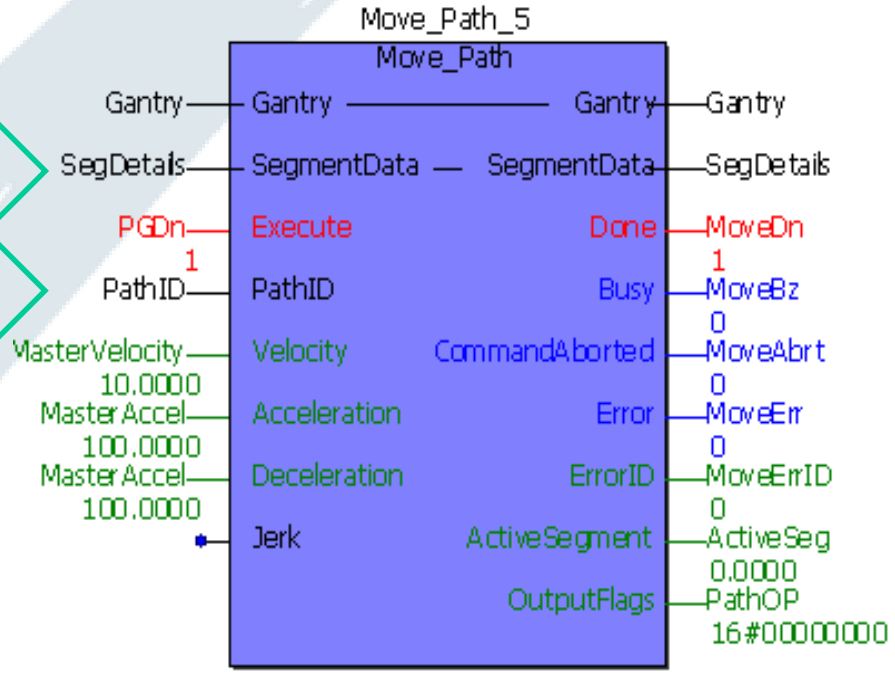
SegDetails	
Segment	
[0]	
Segment	0
OutputFlags	16#00000000
VectorDistance	0.0000000
[1]	
Segment	1
OutputFlags	16#00000001
VectorDistance	10.0000000
[2]	
Segment	2
OutputFlags	16#00000000
VectorDistance	11.5707963
[3]	
Segment	3
OutputFlags	16#00000002
VectorDistance	21.5707963
[4]	
Segment	4
OutputFlags	16#00000000
VectorDistance	23.1415927

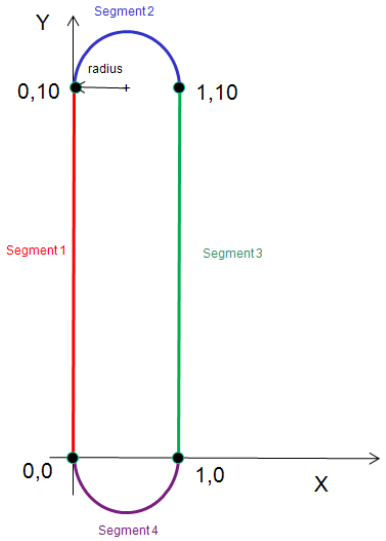
PathID	
XAxisTable	1
YAxisTable	2
PathLength	23.1416
Position	0.0000

Example 1

From Path Generator

From Path Generator

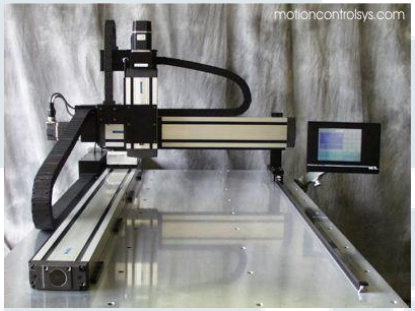


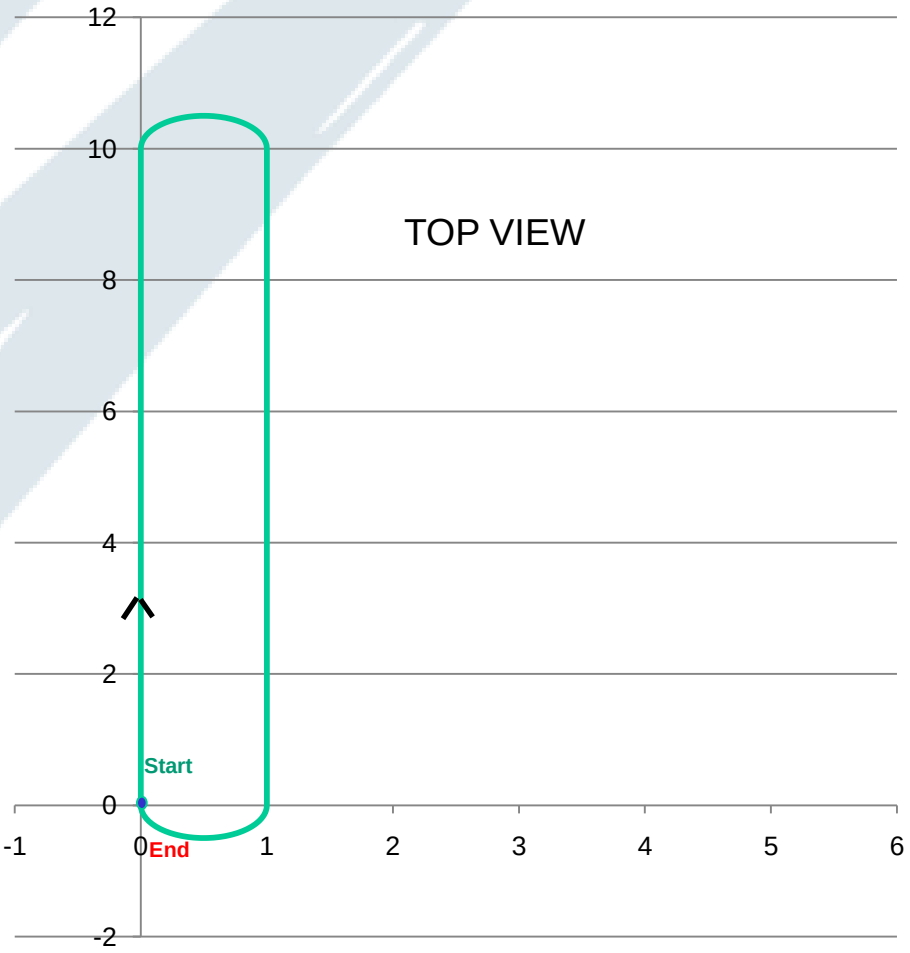


 |

Commanded Profiles

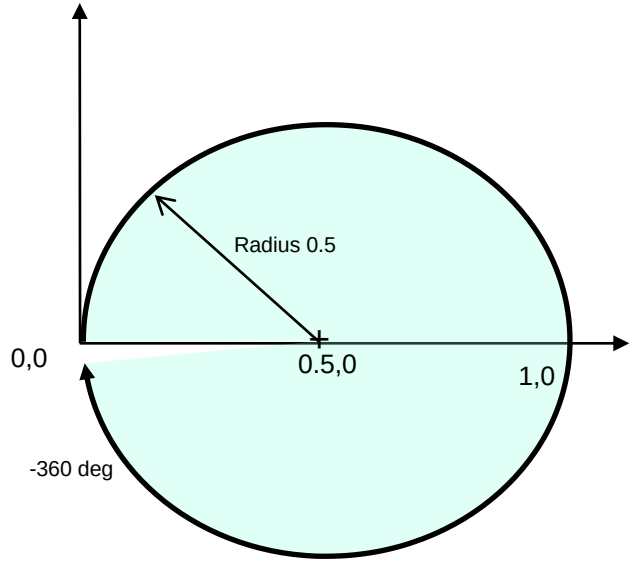
Actual Profile

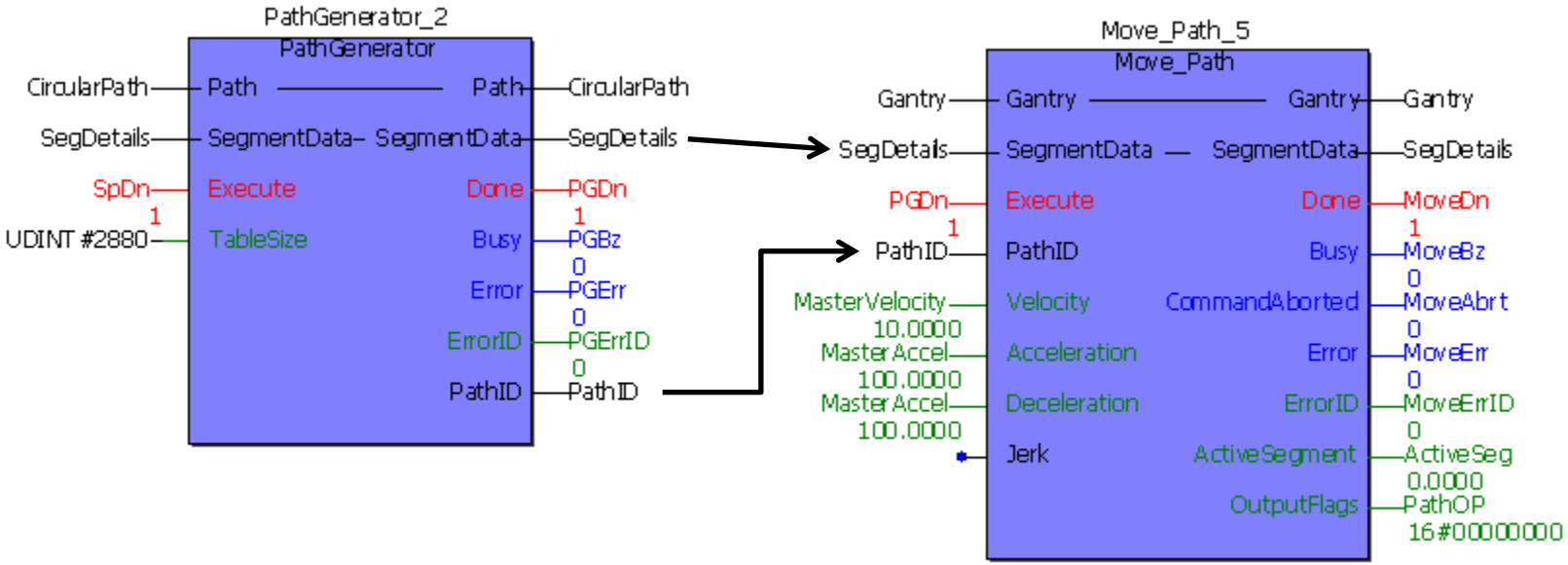




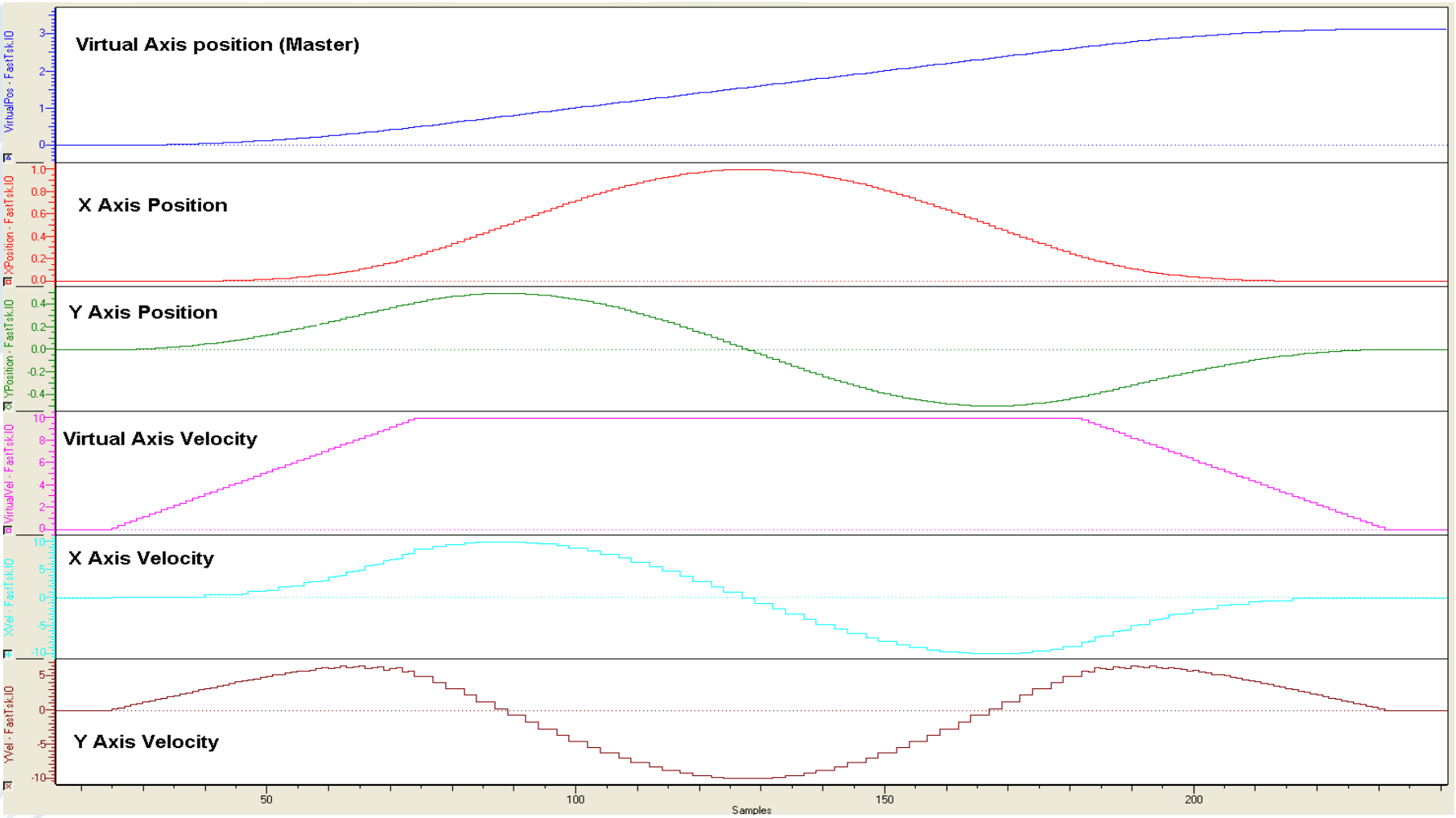
Example 2

```
                (* Circular path*)
                (*==============*)
             2  CircularPath.Data[1].SegmentType:=TB_PatternType#Arc;
           0.5  CircularPath.Data[1].Radius:=LREAL#0.5;
         180.0  CircularPath.Data[1].StartAngle:=LREAL#180.0;
      -360.0000 CircularPath.Data[1].TraversedAngle:=LREAL#-360.0;
        0.0500  CircularPath.Data[1].Resolution:=REAL#0.05;

             1  CircularPath.Segments := INT#1;
```



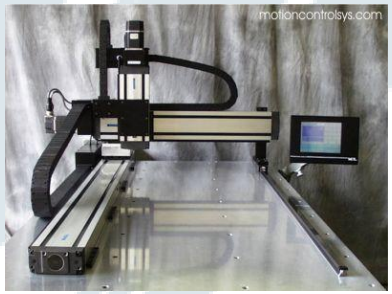


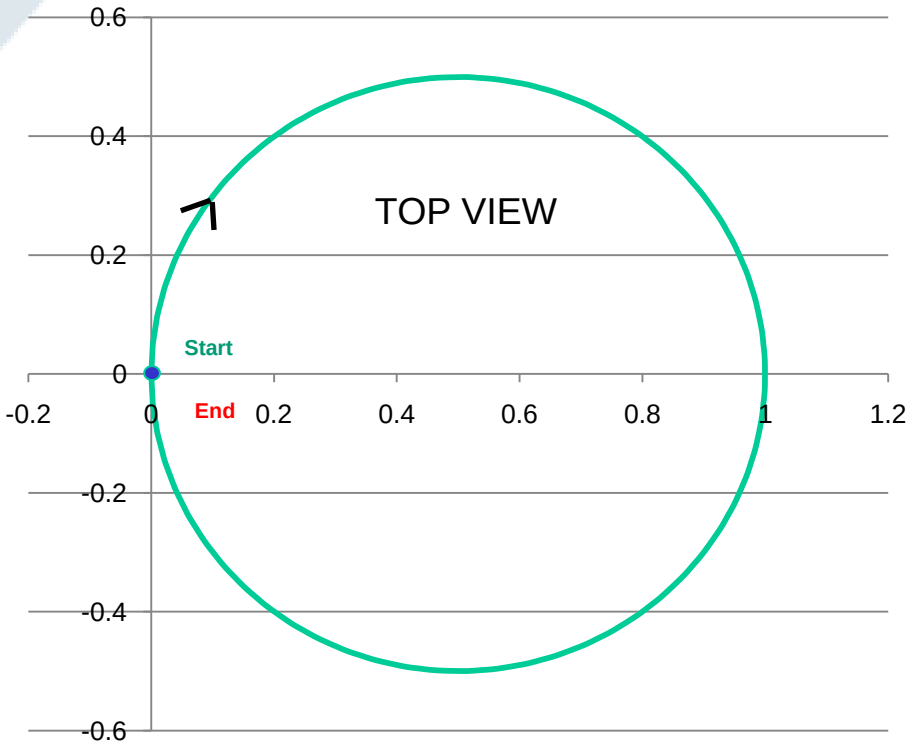
Commanded Profiles



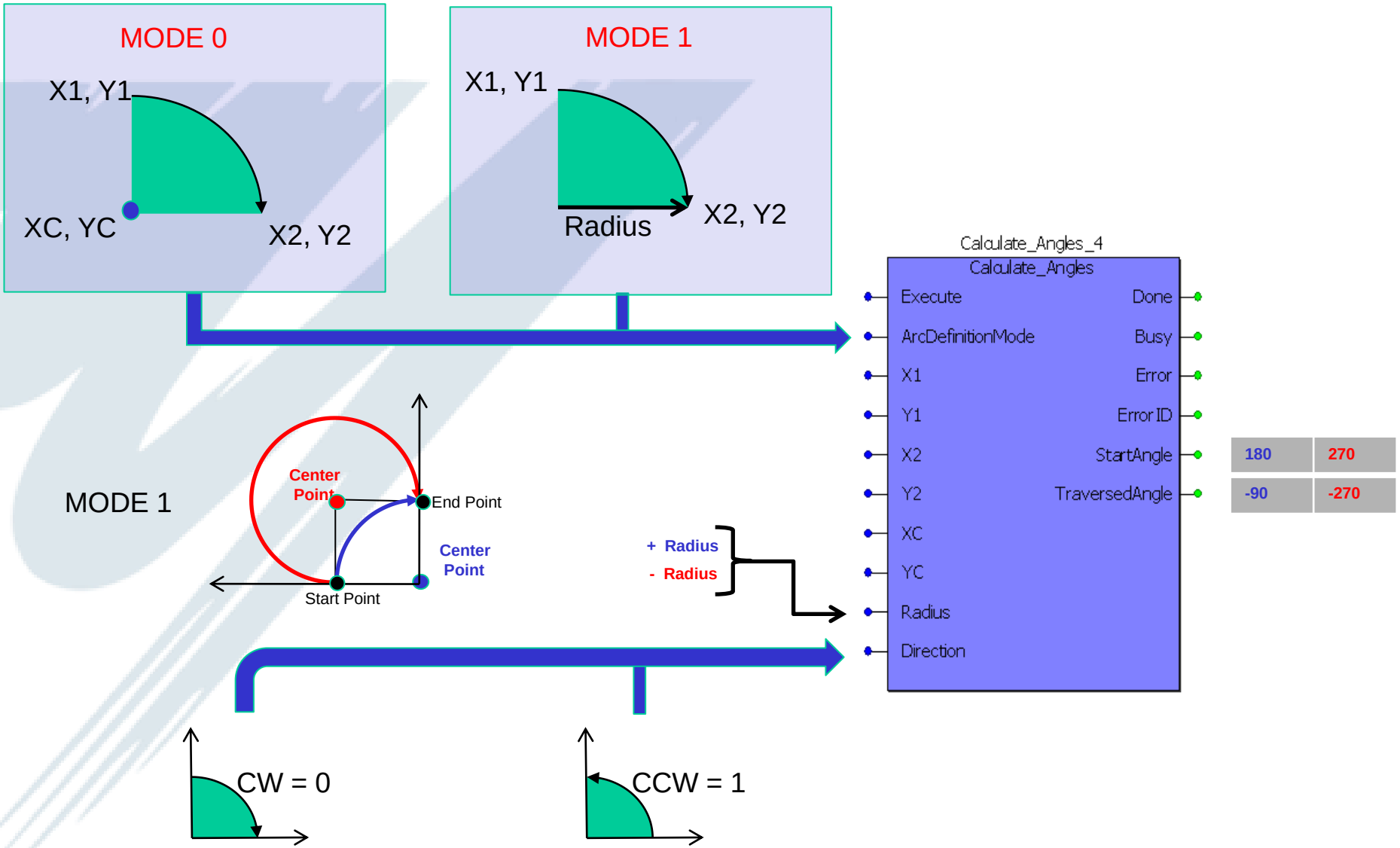
Actual Profile

```
        (* Circular path*)
        (*===============*)
      2 CircularPath.Data[1].SegmentType:=TB_PatternType#Arc;
 0.5000 CircularPath.Data[1].Radius:=LREAL#0.5;
180.0000 CircularPath.Data[1].StartAngle:=LREAL#180.0;
-360.0000 CircularPath.Data[1].TraversedAngle:=LREAL#-360.0;
 0.0500 CircularPath.Data[1].Resolution:=REAL#0.05;

      1 CircularPath.Segments := INT#1;
```

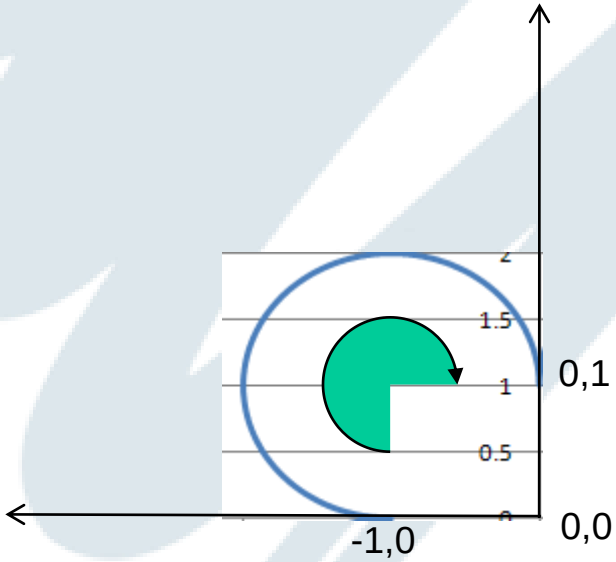


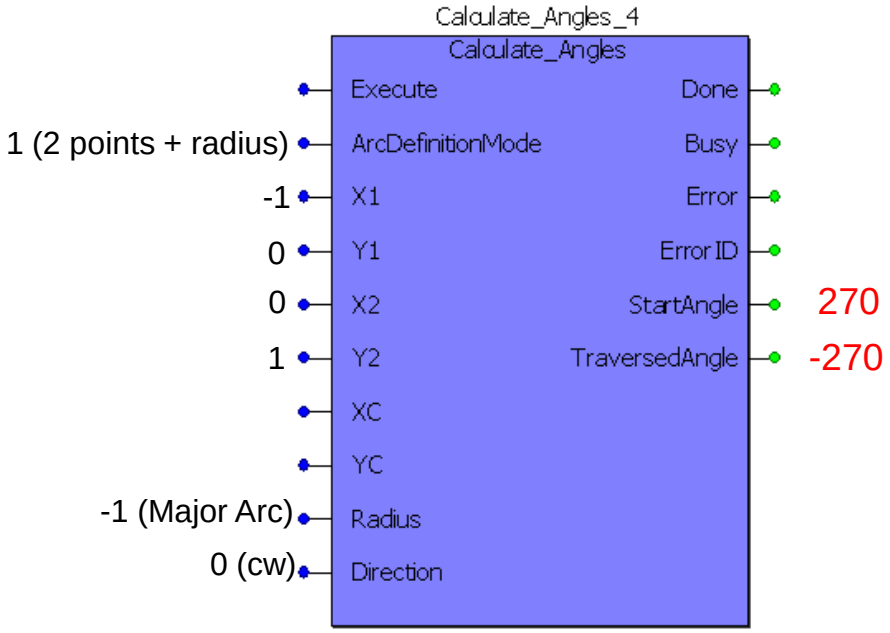


CalculateAngles



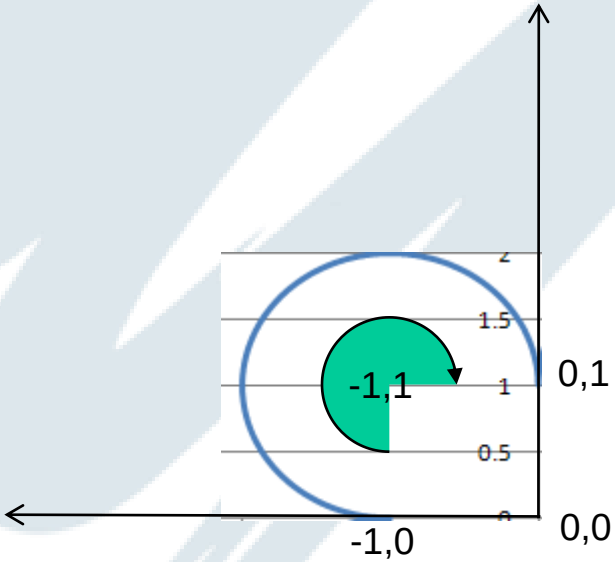
2 Co-ordinates + Radius Mode

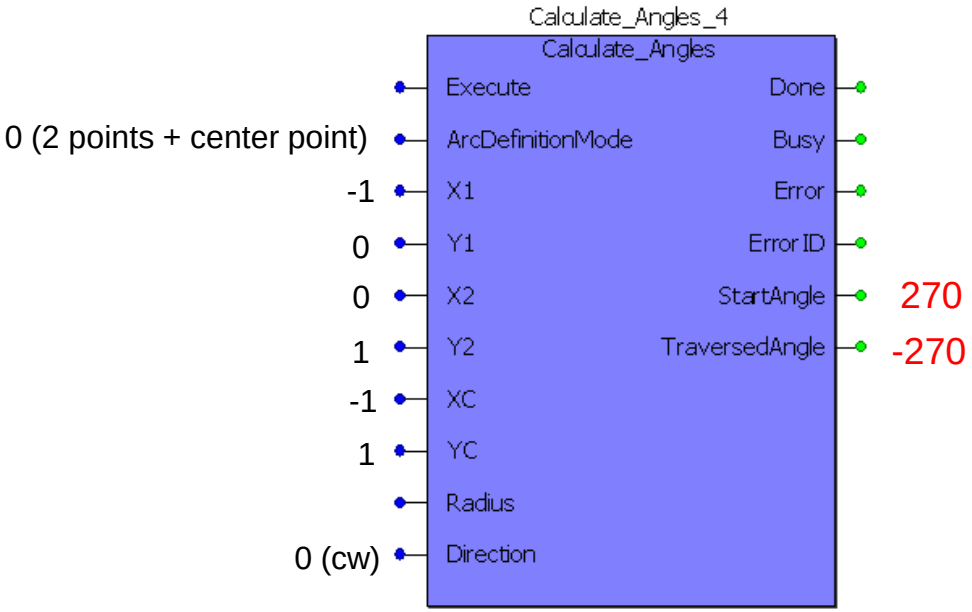




 |

2 Co-ordinates + Center point Mode

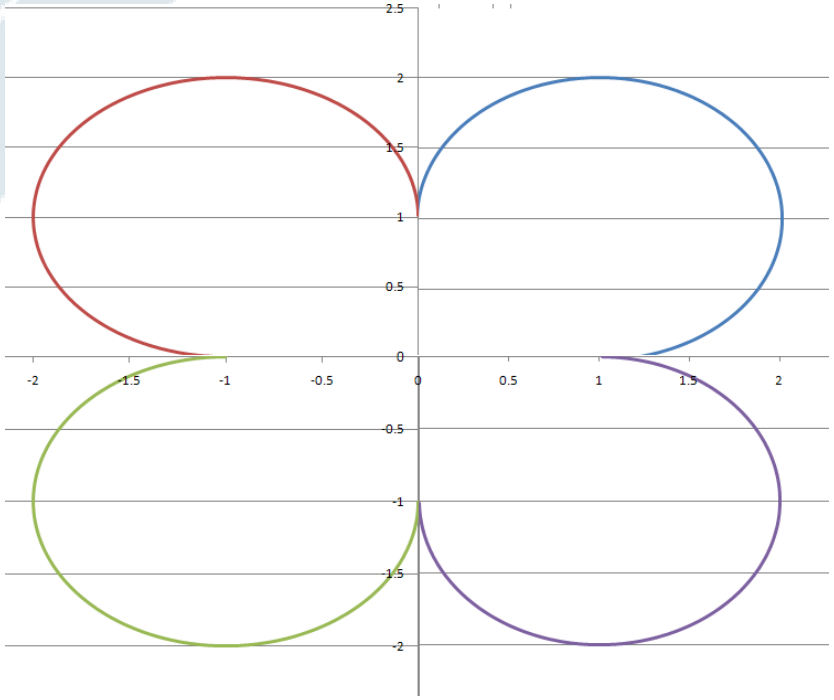




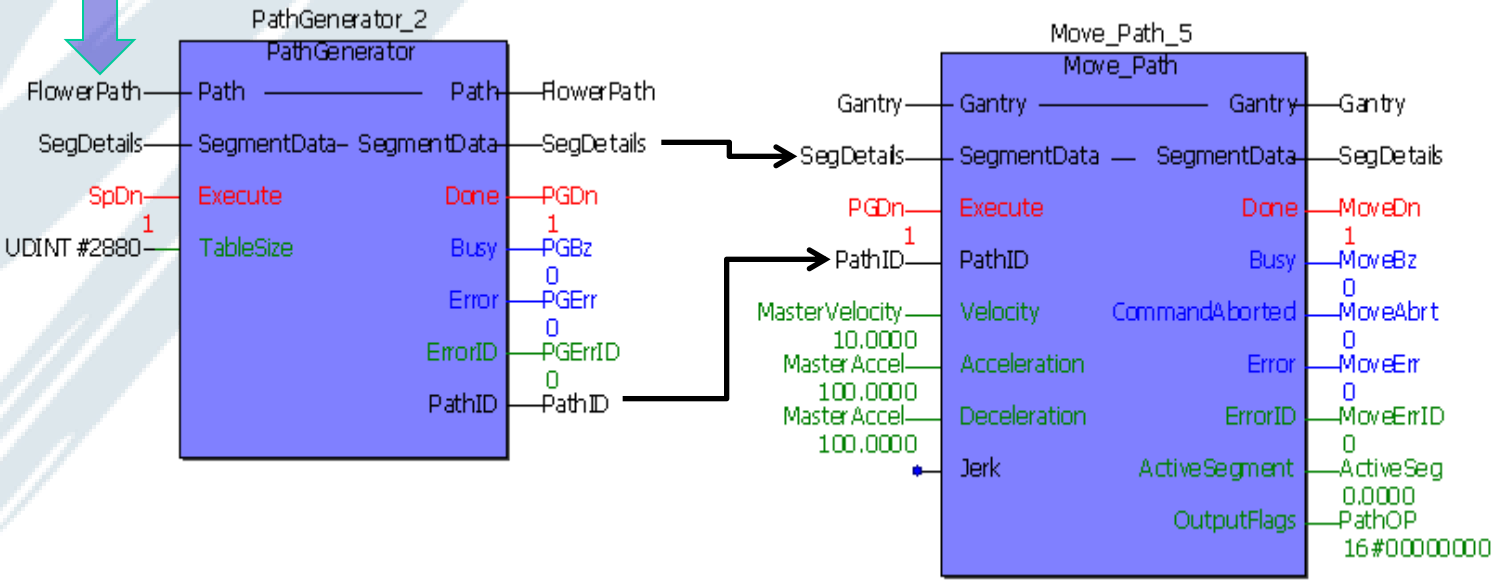
 |

Contour Example

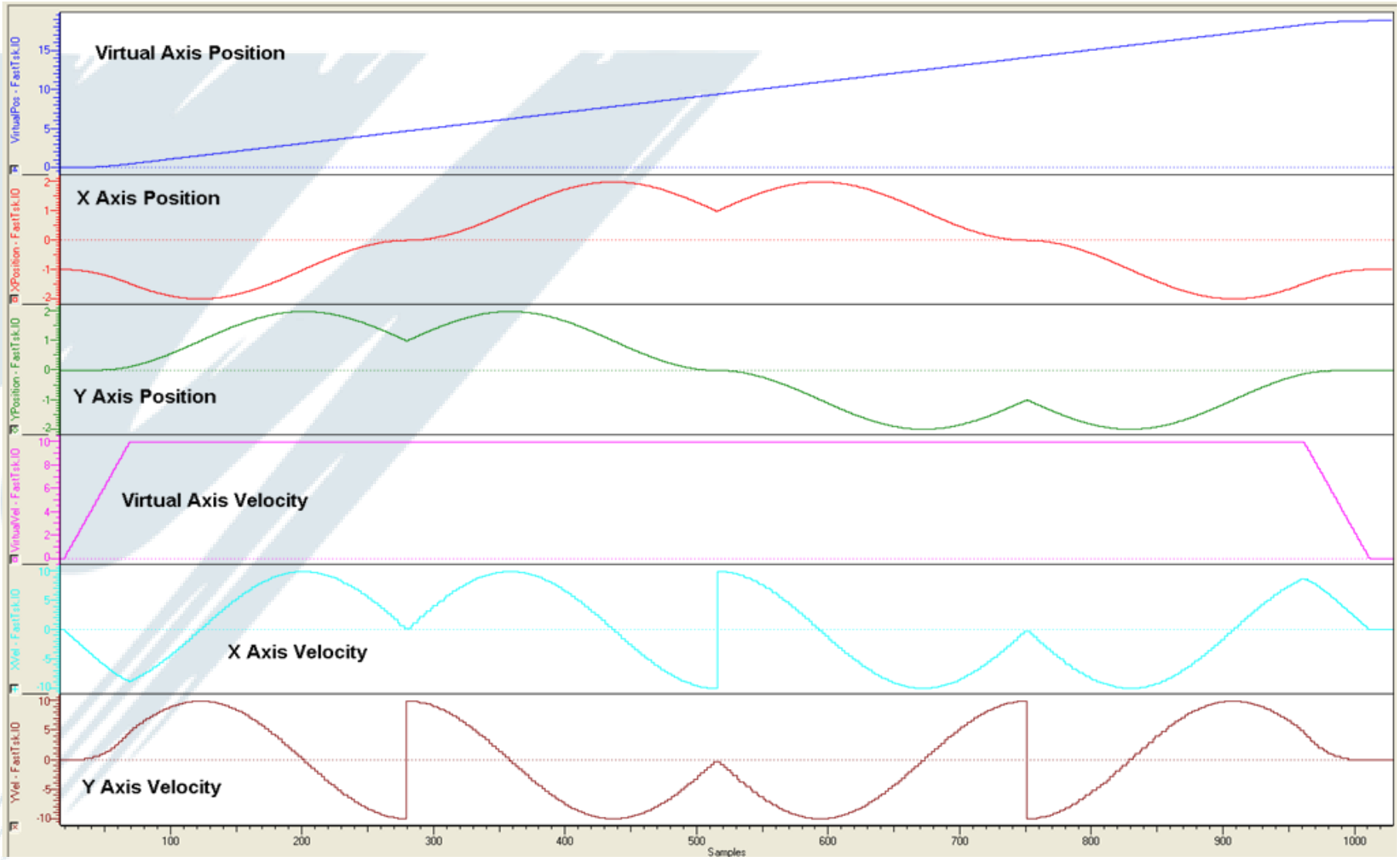
```
Calculate_Angles_1(Execute:=TRUE, ArcDefinitionMode := INT#1, X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_2(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#0.0,X2:=LREAL#1.0,Y1:=LREAL#1.0,Y2:=LREAL#0.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_3(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#-1.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_4(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#0.0,X2:=LREAL#-1.0,Y1:=LREAL#-1.0,Y2:=LREAL#0.0,Radius:=LREAL#-1.0,Direction:=FALSE);
```



```
FlowerPath.Data[1].SegmentType := TB_PatternType#Arc;
FlowerPath.Data[1].Radius:=LREAL#1.0;
Calculate_Angles_1(Execute:=TRUE,ArcDefinitionMode := INT#1, X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);
FlowerPath.Data[1].StartAngle :=Calculate_Angles_1.StartAngle;
FlowerPath.Data[1].TraversedAngle:=Calculate_Angles_1.TraversedAngle;
FlowerPath.Data[1].Resolution:=REAL#0.05;
```



Commanded Profiles



Actual Profile