APPLICATION NOTE YASKAWA

Title: Using GCodeComm.DLL for G Code Streaming	
Product(s): MP3000iec, MotionWorks IEC	**Doc. No. AN.MPIEC.24**

TABLE OF CONTENTS

Title: Using GCodeComm.DLL for G Code Streaming	
Product(s): MP3000iec, MotionWorks IEC	**Doc. No. AN.MPIEC.24**

Application Overview

This document explains the features available in the GCodeComm DLL for use when sending G Code data to an MPiec controller. The G Code data is received in the controller by the Read_GCode_Stream function block from the Group Toolbox.

Required Equipment

Item	Product	Note
Hardware	Any MP3000iec Series controller	MP3000iec supports PLCopen Part 4
Firmware	3.4.0 minimum	To support Part Coordinate System (PCS) and path look ahead.
Software	MotionWorks IEC 3.4.0	For MPiec project development
Software	Comm GUI	For testing and evaluating the DLL, proof of concept for a G Code streaming solution.
Toolbox	Group Toolbox v340	User library which supports UDP packet version 20170103. Note, the Group Toolbox relies on several other supporting libraries, see the Toolbox documentation.
DLL	GCodeComm	Provides the communication link between a user interface and the MPiec controller specifically for streaming G Code data and receiving status feedback.

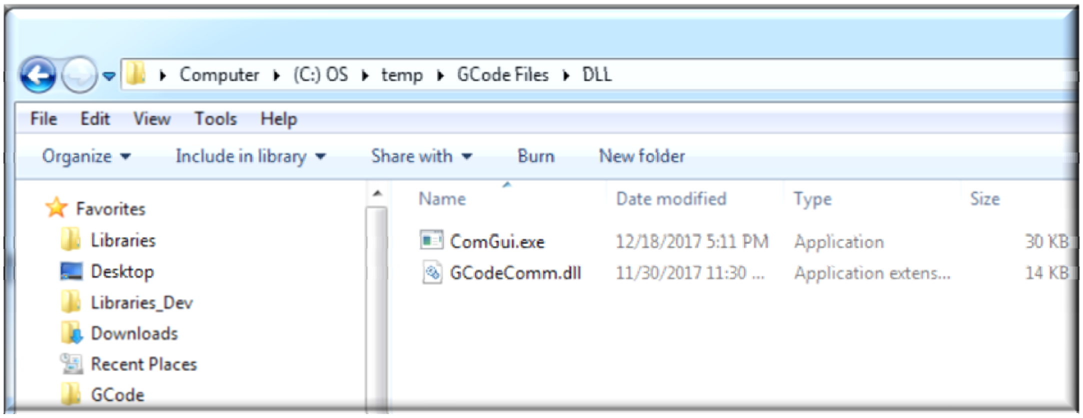
Compatibility / Revision History

Each row listed represents cross compatibility of the three items listed.

	version		
Release Date	DLL	UDP Packet	Group Toolbox
January 2018	3.4.1.0	20180103	v340

Installation

For a basic installation, the DLL can simply be placed in the same directory as the executable which will access it.



Name	Date modified	Type	Size
ComGui.exe	12/18/2017 5:11 PM	Application	30 KB
GCodeComm.dll	11/30/2017 11:30 ...	Application extens...	14 KB

In some installations, such as when using the DLL with Indusoft, it may be required to register the DLL as follows:

"C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe" "C:\MyPath\GCodeComm.dll" /codebase

Replace the highlighted text with the actual path to the DLL on your system.

Firewall Considerations

This DLL provides outbound TCP data and listens for inbound UDP data sent back the Read_GCode_Stream function block to the IP address which initiated the connection. The Read_GCode_Stream function block will increment the port number of the TCP connection initiated by GCodeComm, and transmit UDP status data on the “TCP Port + 1.” If a firewall blocks this inbound traffic, the DLL will not receive updates from the MPiec controller, and the G Code streaming features will not work successfully. Ensure that any firewall in your system is configured to allow TCP and UDP traffic on the selected ports of your choosing.

Methods

The following methods are available within GCodeComm.Stream

GCodeComm.Stream client_ = new GCodeComm.Stream();

Method	Description
Connect(IPAddress ip, int tcpPort)	One of two styles of connection methods available. Initiates a TCP connection with the Read_Gcode_Stream function block executing on the MPiec Controller. Set Read_GCode_Stream.Enable:=TRUE prior to executing the Connect Method, otherwise an error will result from the Connect method. Also be sure the function block is reporting the Valid output with no Errors. Upon successful connection, Read_GCode_Stream.HostConnected will be set to TRUE.
Connect(IPAddress ip, int tcpPort, int udpPort)	Second of two styles of connection methods available. Both do the same thing
Disconnect()	Terminates the connection. On the MPiec Controller, Read_GCode_Stream.HostConnected will be set to FALSE.
SendData(string stringData)	Option #1 for sending G Code. The application opens a file into a String, and passes the entire contents of the G Code data.
SendFile(string fileLocation, int byteOffset)	Option#2 for sending G Code. byteOffset is normally zero unless trying to resume a path in progress after a fault. This non zero byte offset would likely come from the last known MotionByteOffset provided by the MPiec controller, and indicates the file location which corresponds to the last executed motion instruction.
Pause()	Stops the DLL from sending data to the MPiec controller until Resume is invoked. On the MPiec Controller, the data already received will continue to be processed, so the mechanism may continue to operate for a short time. If the motion must be paused (Interrupted) immediately with expectation of resuming, the MC_GroupInterrupt function block on the controller must be executed.
Resume()	If Paused, the DLL will resume sending G Code data to the MPiec controller.
Cancel()	Stops the DLL from sending any more data to the MPiec controller. Once cancel is invoked, resume is not possible.

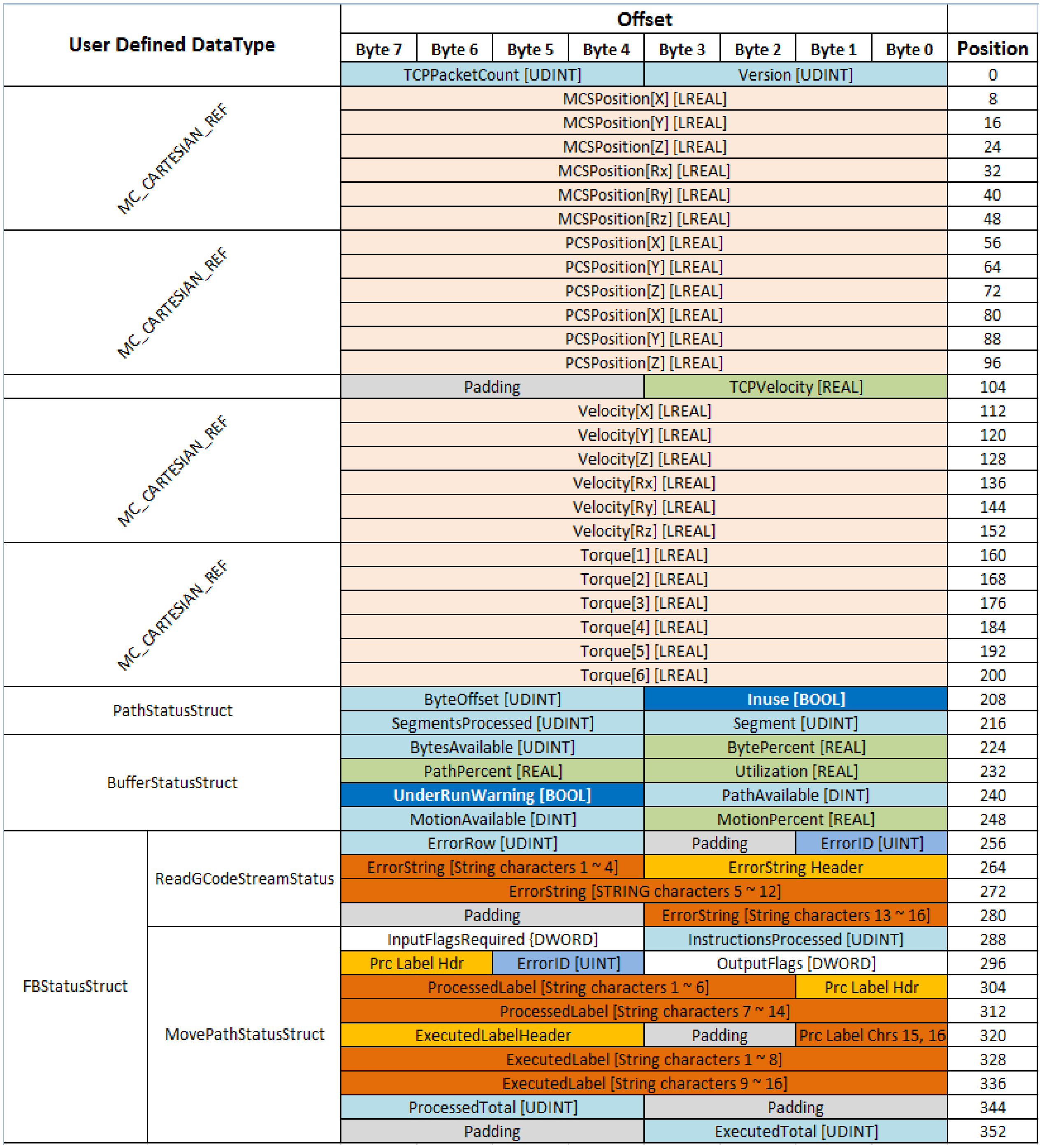
Properties

Properties	DataType	Description
Status	BOOL	Reports if there is data being streamed.
Connected	BOOL	Reports if the DLL is connected to the MPiec / Read_GCode_Stream function block
Progress	Double (64 bit)	Reports the fraction of data that has been sent to the controller.
Paused	BOOL	Reports if the data stream is currently paused.
MotionByteOffset	UINT (32 bit)	Reports the byte offset from the time the connection was initiated that relates to the first character of the G Code block which is currently providing motion. If there was a fault or error on the MPiec controller, this value refers to the last G Code block which was attempted.
RemainingBytes	UINT (32 bit)	Reports the number of bytes of data that are waiting to be sent to the MPiec controller.
UDP (Class)	StreamStruct	Reports the structured data received from the MPiec controller over UDP.
TCP (Class)		

Events

Events	Description
NewDataEvent	Raised each time a UDP packet is received from the MPiec Controller.
VersionMismatchEvent	Raised if the UDP packet version ID does not match the version the DLL is expecting. Streaming is not possible if there is a packet version mismatch. See the Compatibility/Revision history chart in this document.

UDP Packet version 20180103



User Defined DataType		Offset								
		Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Position
		TCPPacketCount [UDINT]				Version [UDINT]				0
MC_CARTESIAN_REF		MCSPosition[X] [LREAL]								8
		MCSPosition[Y] [LREAL]								16
		MCSPosition[Z] [LREAL]								24
		MCSPosition[Rx] [LREAL]								32
		MCSPosition[Ry] [LREAL]								40
		MCSPosition[Rz] [LREAL]								48
MC_CARTESIAN_REF		PCSPosition[X] [LREAL]								56
		PCSPosition[Y] [LREAL]								64
		PCSPosition[Z] [LREAL]								72
		PCSPosition[X] [LREAL]								80
		PCSPosition[Y] [LREAL]								88
		PCSPosition[Z] [LREAL]								96
		Padding				TCPVelocity [REAL]				104
MC_CARTESIAN_REF		Velocity[X] [LREAL]								112
		Velocity[Y] [LREAL]								120
		Velocity[Z] [LREAL]								128
		Velocity[Rx] [LREAL]								136
		Velocity[Ry] [LREAL]								144
		Velocity[Rz] [LREAL]								152
MC_CARTESIAN_REF		Torque[1] [LREAL]								160
		Torque[2] [LREAL]								168
		Torque[3] [LREAL]								176
		Torque[4] [LREAL]								184
		Torque[5] [LREAL]								192
		Torque[6] [LREAL]								200
PathStatusStruct		ByteOffset [UDINT]				**Inuse [BOOL]**				208
		SegmentsProcessed [UDINT]				Segment [UDINT]				216
BufferStatusStruct		BytesAvailable [UDINT]				BytePercent [REAL]				224
		PathPercent [REAL]				Utilization [REAL]				232
		UnderRunWarning [BOOL]				PathAvailable [DINT]				240
		MotionAvailable [DINT]				MotionPercent [REAL]				248
FBStatusStruct	ReadGCodeStreamStatus	ErrorRow [UDINT]				Padding		ErrorID [UINT]		256
		ErrorString [String characters 1 ~ 4]				ErrorString Header				264
		ErrorString [STRING characters 5 ~ 12]								272
		Padding				ErrorString [String characters 13 ~ 16]				280
	MovePathStatusStruct	InputFlagsRequired {DWORD]				InstructionsProcessed [UDINT]				288
		Prc Label Hdr		ErrorID [UINT]		OutputFlags [DWORD]				296
		ProcessedLabel [String characters 1 ~ 6]						Prc Label Hdr		304
		ProcessedLabel [String characters 7 ~ 14]								312
		ExecutedLabelHeader				Padding		Prc Label Chrs 15, 16		320
		ExecutedLabel [String characters 1 ~ 8]								328
		ExecutedLabel [String characters 9 ~ 16]								336
		ProcessedTotal [UDINT]				Padding				344
		Padding				ExecutedTotal [UDINT]				352

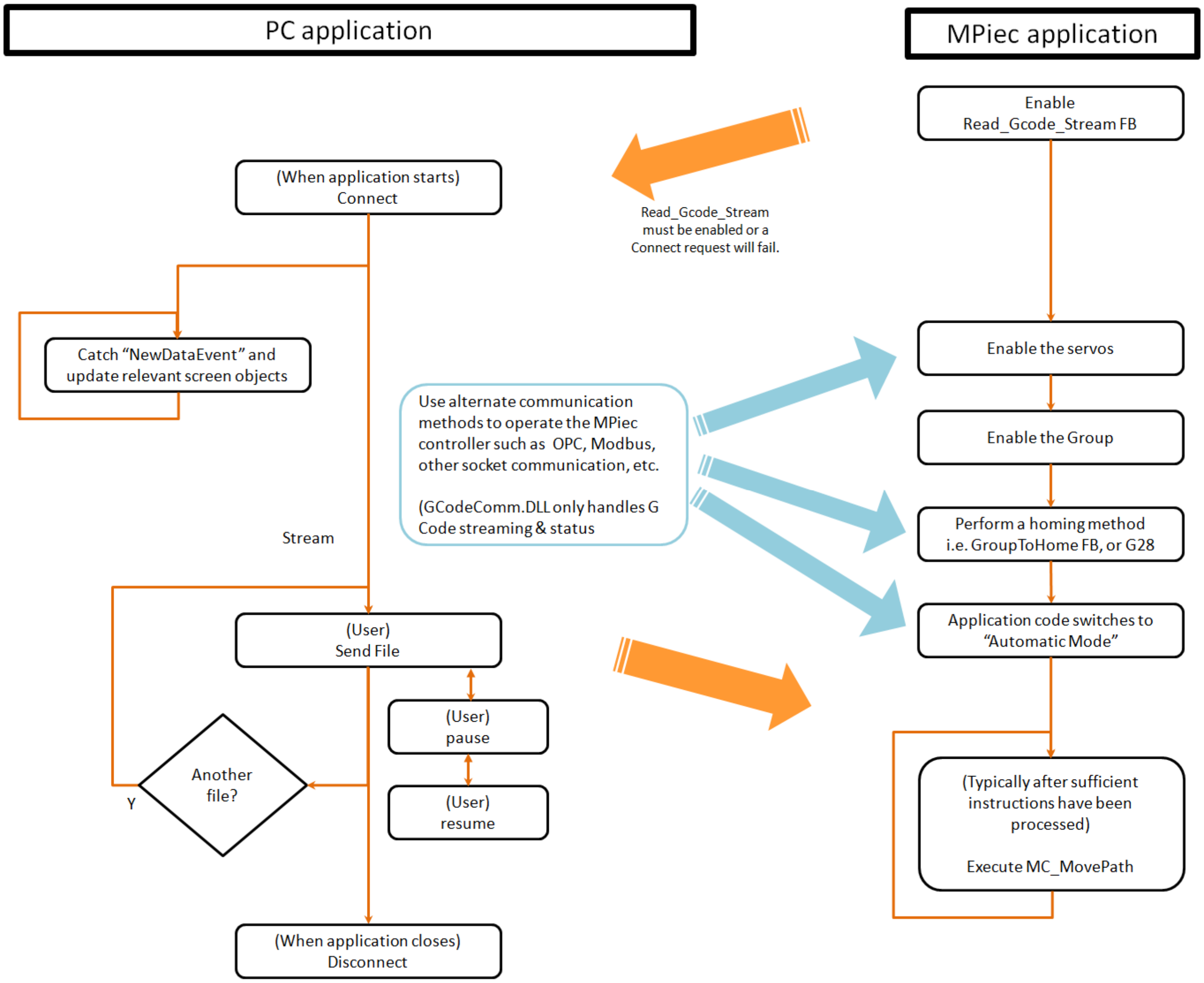
Class definition in Visual Studio

```
[StructLayout(LayoutKind.Sequential, Pack = 8)]
    public class PathStatusStruct
    {
        public bool InUse;
        public UInt32 ByteOffset;
        public UInt32 PathSegment;
        public UInt32 PathSegmentsProcessed;
    }

    [StructLayout(LayoutKind.Sequential, Pack = 8)]
    public class BufferStatusStruct
    {
        public float BytePercent;
        public Int32 BytesAvailable;
        public float BytesUtilization;
        public float PathPercent;
        public Int32 PathAvailable;
        public bool UnderRunWarning;
        public float MotionPercent;
        public Int32 MotionAvailable;
    }

    [StructLayout(LayoutKind.Sequential, Pack = 8)]
    public class ReadGCodeStreamStatus
    {
        public UInt16 ErrorID;
        public UInt32 ErrorRow;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 4)]
        private string ErrorStringHeader;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 18)]
        public string ErrorString;
        public UInt32 InstructionsProcessed;
    }

    [StructLayout(LayoutKind.Sequential, Pack = 8)]
    public class MovePathStatusStruct
    {
        public Int32 InputFlagsRequired;
        public Int32 OutputFlags;
        public UInt16 ErrorID;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 4)]
        private string ProcessedStringHeader;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 18)]
        public string ProcessedLabel;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 4)]
        private string ExecutedStringHeader;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 18)]
        public string ExecutedLabel;
        public UInt32 ProcessedTotal;
        public UInt32 ExecutedTotal;
    }

    [StructLayout(LayoutKind.Sequential, Pack = 8)]
    public class FBStatusStruct
    {
        public ReadGCodeStreamStatus ReadStream = new ReadGCodeStreamStatus();
        public MovePathStatusStruct MovePath = new MovePathStatusStruct();
    }
```

```
[StructLayout(LayoutKind.Sequential, Pack = 8)]
    public class MC_CARTESIAN_REF
    {
        public double X;
        public double Y;
        public double Z;
        public double Rx;
        public double Ry;
        public double Rz;
    }

    [StructLayout(LayoutKind.Sequential, Pack = 8)]
    public class StreamStruct
    {
        public UInt32 Version;      //  4 BYTE: To uniquely identify the structure definition
        public UInt32 TCPPacketCount;
        public MC_CARTESIAN_REF MCSPosition = new MC_CARTESIAN_REF();
        public MC_CARTESIAN_REF PCSPosition = new MC_CARTESIAN_REF();
        public float TCPVelocity;
        public MC_CARTESIAN_REF Velocity = new MC_CARTESIAN_REF();
        public MC_CARTESIAN_REF Torque = new MC_CARTESIAN_REF();
        public PathStatusStruct PathStatus = new PathStatusStruct();
        public BufferStatusStruct Buffer = new BufferStatusStruct();
        public FBStatusStruct FBStatus = new FBStatusStruct();
    }
}
```

Operational Process

An application using the GCodeComm DLL is expected to provide the following sequence of operation:


PC application
MPiec application
Enable
Read_Gcode_Stream FB
(When application starts)
Connect
Read_Gcode_Stream
must be enabled or a
Connect request will fail.
Catch "NewDataEvent" and
update relevant screen objects
Enable the servos
Use alternate communication
methods to operate the MPiec
controller such as OPC, Modbus,
other socket communication, etc.
(GCodeComm.DLL only handles G
Code streaming & status
Enable the Group
Perform a homing method
i.e. GroupToHome FB, or G28
Stream
Application code switches to
"Automatic Mode"
(User)
Send File
(User)
pause
Another
file?
Y
(User)
resume
(Typically after sufficient
instructions have been
processed)
Execute MC_MovePath
(When application closes)
Disconnect